Turbocharging Treewidth Heuristics

Serge Gaspers[1,2], Joachim Gudmundsson[3], Mitchell Jones[3], Julián Mestre[3], and Stefan Rümmele[3,1,2]

[1]UNSW, Australia
[2]Data61, CSIRO, Australia
[3]University of Sydney, Australia

Presented at IPEC 2016

Overview

Tree Decompositions

- Tree decomposition of $\mathcal{G} = (V, E)$

Tree Decompositions

- Tree decomposition of $\mathcal{G} = (V, E)$
- A pair $\mathcal{T} = (T, \chi)$, tree T.

Tree Decompositions

- Tree decomposition of $\mathcal{G} = (V, E)$
- A pair $\mathcal{T} = (T, \chi)$, tree T.
- Each $t \in T$ has a bag $\chi(t) \subseteq V$.

Tree Decompositions

- Tree decomposition of $\mathcal{G} = (V, E)$
- A pair $\mathcal{T} = (T, \chi)$, tree T.
- Each $t \in T$ has a bag $\chi(t) \subseteq V$.
- Properties:

Tree Decompositions

- Tree decomposition of $\mathcal{G} = (V, E)$
- A pair $\mathcal{T} = (T, \chi)$, tree T.
- Each $t \in T$ has a bag $\chi(t) \subseteq V$.
- Properties:
  1. For each $v \in V$, there is a $t \in T$ such that $v \in \chi(t)$.

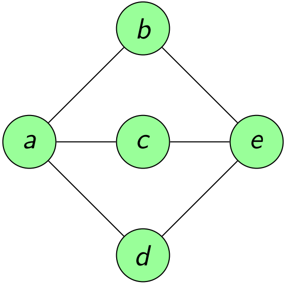
Tree Decompositions

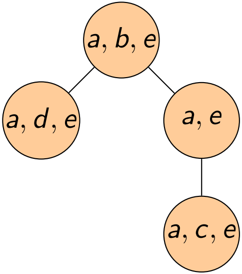
- Tree decomposition of $\mathcal{G} = (V, E)$
- A pair $\mathcal{T} = (T, \chi)$, tree T.
- Each $t \in T$ has a bag $\chi(t) \subseteq V$.
- Properties:
  1. For each $v \in V$, there is a $t \in T$ such that $v \in \chi(t)$.
  2. For each $\{ u, v \} \in E$, there is a $t \in T$ such that $\{ u, v \} \subseteq \chi(t)$.

Tree Decompositions

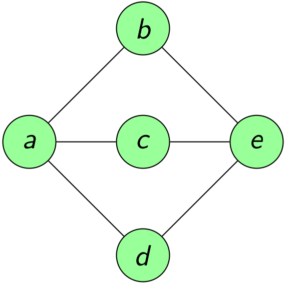
- Tree decomposition of $\mathcal{G} = (V, E)$
- A pair $\mathcal{T} = (T, \chi)$, tree T.
- Each $t \in T$ has a bag $\chi(t) \subseteq V$.
- Properties:
  1. For each $v \in V$, there is a $t \in T$ such that $v \in \chi(t)$.
  2. For each $\{ u, v \} \in E$, there is a $t \in T$ such that $\{ u, v \} \subseteq \chi(t)$.
  3. For each $v \in V$, the set $\{ t \in T \mid v \in \chi(t) \}$ forms a connected subtree of T.

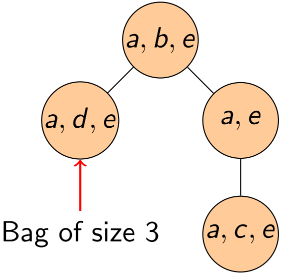
Example





Example





Tree Decompositions

1. The width of $\mathcal{T}$: $\max_{t \in T}(|\chi(t)| - 1)$.

Tree Decompositions

1. The width of $\mathcal{T}$: $\max_{t \in T}(|\chi(t)| - 1)$.
2. The treewidth of $\mathcal{G}$: min width over all tree decompositions.

Tree Decompositions

1. The width of $\mathcal{T}$: $\max_{t \in T}(|\chi(t)| - 1)$.
2. The treewidth of $\mathcal{G}$: min width over all tree decompositions.

Treewidth

Instance: Graph $\mathcal{G}$ and integer k.
Problem: Decide whether $\mathrm{tw}(\mathcal{G}) \leq k$ holds.

Motivation

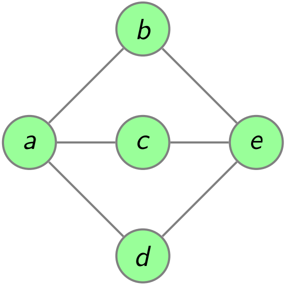
- Many problems can be solved easily on trees (independent set).
- Find graphs that are "tree"-like.
- Many problems solved efficiently on graphs of bounded treewidth.

Elimination Orders

- Construct tree given elimination ordering π.

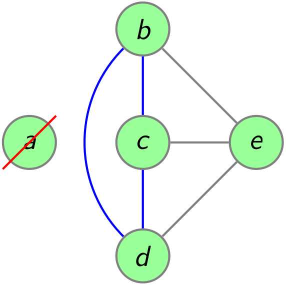
Elimination Orders

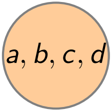
- Construct tree given elimination ordering π.
- To eliminate v:
  1. Create t_v, $\chi(t_v) = \{ v \} \cup N(v)$.
  2. Form a clique out of $N(v)$.



- Elimination order $\pi = (a, e, b, c, d)$.

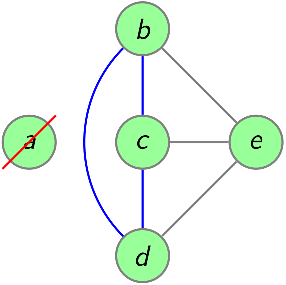
1. Eliminate a.





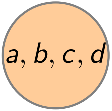
- Elimination order $\pi = (a, e, b, c, d)$.

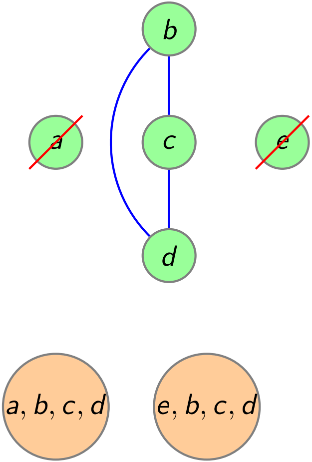
1. Eliminate a.
2. Create bag $\{\, a, b, c, d \,\}$.



- Elimination order $\pi = (a, e, b, c, d)$.

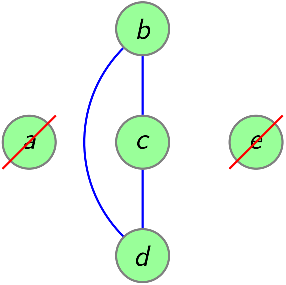
1. Eliminate a.
2. Create bag $\{ a, b, c, d \}$.
3. Eliminate e.





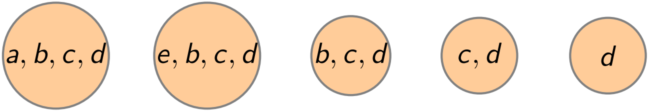
- Elimination order $\pi = (a, e, b, c, d)$.

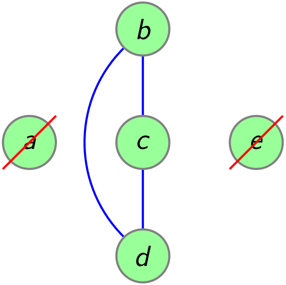
1. Eliminate a.
2. Create bag $\{ a, b, c, d \}$.
3. Eliminate e.
4. Create bag $\{ e, b, c, d \}$.



- Elimination order $\pi = (a, e, b, c, d)$.

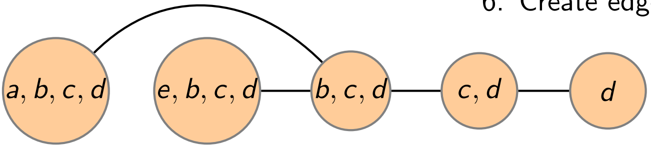
1. Eliminate a.
2. Create bag $\{\, a, b, c, d \,\}$.
3. Eliminate e.
4. Create bag $\{\, e, b, c, d \,\}$.
5. And so on..





- Elimination order $\pi = (a, e, b, c, d)$.

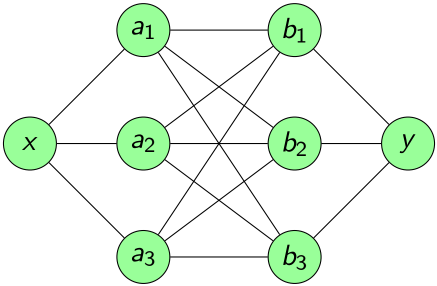
1. Eliminate a.
2. Create bag $\{ a, b, c, d \}$.
3. Eliminate e.
4. Create bag $\{ e, b, c, d \}$.
5. And so on..
6. Create edges.



- Width of π: max degree of any vertex *during* elimination.
- $\mathcal{G}$ has treewidth $\leq k \iff \pi$ has width $\leq k$ (e.g., Bodlaender, Koster 2010).

Greedy Algorithms

- GreedyDegree : select next vertex with smallest degree.
- GreedyFillIn : select next vertex whose elimination results in the fewest new edges.



Overview

Parameterized complexity review

- Given input x and parameter k, language L.

Parameterized complexity review

- Given input x and parameter k, language L.
- Decide $(x, k) \in L$ in $f(k) \cdot |x|^{O(1)}$.

Parameterized complexity review

- Given input x and parameter k, language L.
- Decide $(x, k) \in L$ in $f(k) \cdot |x|^{O(1)}$.
- Then L is fixed-parameter tractable (FPT).

Parameterized complexity review

- Given input x and parameter k, language L.
- Decide $(x, k) \in L$ in $f(k) \cdot |x|^{O(1)}$.
- Then L is fixed-parameter tractable (FPT).
- Vertex Cover, parameterized by size k: $O(2^k kn)$ time.

Parameterized complexity review

- Given input x and parameter k, language L.
- Decide $(x, k) \in L$ in $f(k) \cdot |x|^{O(1)}$.
- Then L is fixed-parameter tractable (FPT).
- Vertex Cover, parameterized by size k: $O(2^k kn)$ time.
- Independent Set, parameterized by size k: unknown.

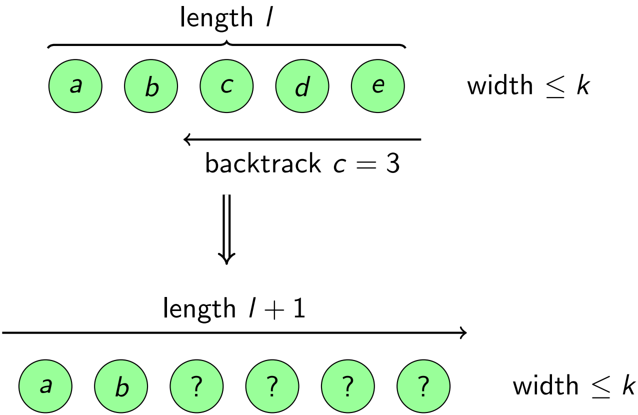
Parameterized complexity review

- Given input x and parameter k, language L.
- Decide $(x, k) \in L$ in $f(k) \cdot |x|^{O(1)}$.
- Then L is fixed-parameter tractable (FPT).
- Vertex Cover, parameterized by size k: $O(2^k kn)$ time.
- Independent Set, parameterized by size k: unknown.
- Independent Set is in the class W[1] $\supseteq$ FPT.

Parameterized complexity review

- Given input x and parameter k, language L.
- Decide $(x, k) \in L$ in $f(k) \cdot |x|^{O(1)}$.
- Then L is fixed-parameter tractable (FPT).
- Vertex Cover, parameterized by size k: $O(2^k kn)$ time.
- Independent Set, parameterized by size k: unknown.
- Independent Set is in the class W[1] $\supseteq$ FPT.
- Problems in W[1] are 'harder' than problems in FPT.

Incremental Conservative Treewidth



Incremental Conservative Treewidth

IC-Treewidth

Instance: Graph $\mathcal{G}$, integers k and c, partial elimination order π of length l and width $\leq k$.
Problem: Does there exist a partial elimination order π' of length $l + 1$ and width $\leq k$ such that π and π' are identical on the first $l - c$ positions.

Hardness of IC-Treewidth

Theorem

IC-Treewidth *is W[1]-hard when parameterized by l.*

Hardness of IC-Treewidth

Theorem

IC-Treewidth *is W[1]-hard when parameterized by l.*

- Independent Set : Given $\mathcal{G} = (V, E)$, does there exist an independent set of size t?

Hardness of IC-Treewidth

Theorem
IC-Treewidth *is W[1]-hard when parameterized by l.*

- Independent Set : Given $\mathcal{G} = (V, E)$, does there exist an independent set of size t?
- Independent Set is W[1]-complete with parameter t.

Hardness of IC-Treewidth

Theorem

IC-Treewidth *is W[1]-hard when parameterized by l.*

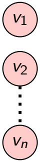
- Independent Set: Given $\mathcal{G} = (V, E)$, does there exist an independent set of size t?
- Independent Set is W[1]-complete with parameter t.
- Independent Set $\implies$ IC-Treewidth.

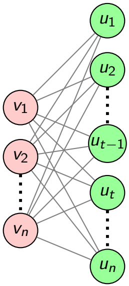
Hardness of IC-Treewidth

Theorem

IC-Treewidth *is W[1]-hard when parameterized by l.*

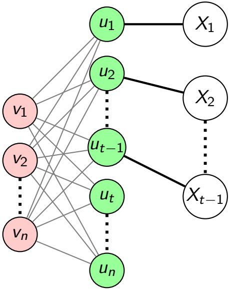
- Independent Set : Given $\mathcal{G} = (V, E)$, does there exist an independent set of size t?
- Independent Set is W[1]-complete with parameter t.
- Independent Set $\implies$ IC-Treewidth .
- $V = \{ v_1, \ldots, v_n \}$, $d = \max_{v \in V} \deg_{\mathcal{G}}(v)$.

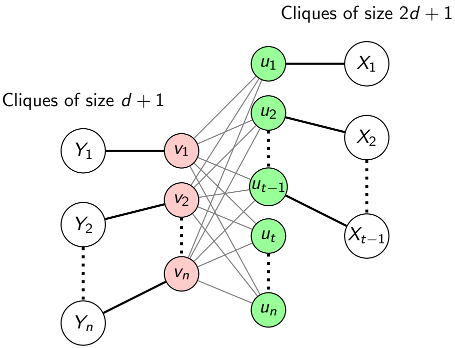

v_1
v_2
v_n



u_1
u_2
v_1
u_{t-1}
v_2
u_t
v_n
u_n

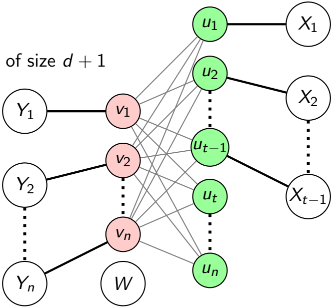
Cliques of size $2d + 1$



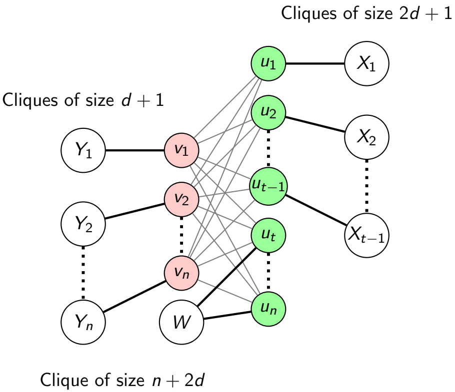

Cliques of size $2d+1$
Cliques of size $d+1$
Y_1
Y_2
Y_n
v_1
v_2
v_n
u_1
u_2
u_{t-1}
u_t
u_n
X_1
X_2
X_{t-1}

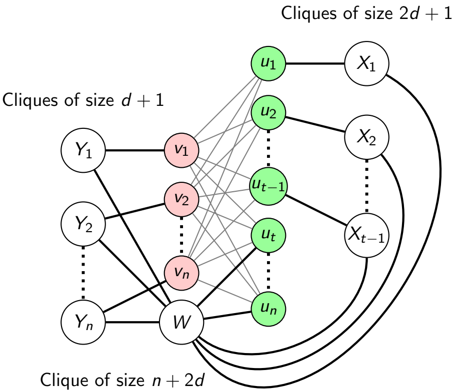
Cliques of size $2d+1$

Cliques of size $d+1$

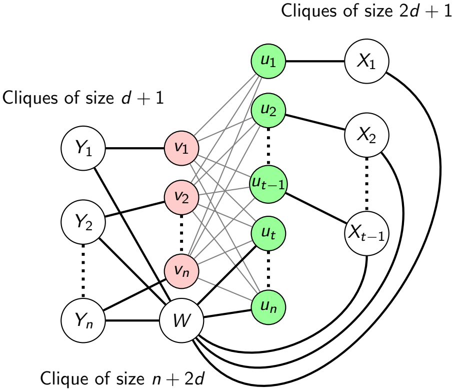


Clique of size $n+2d$


Cliques of size $2d+1$
Cliques of size $d+1$
u_1
X_1
u_2
X_2
Y_1
v_1
u_{t-1}
v_2
Y_2
u_t
X_{t-1}
v_n
u_n
Y_n
W
Clique of size $n+2d$

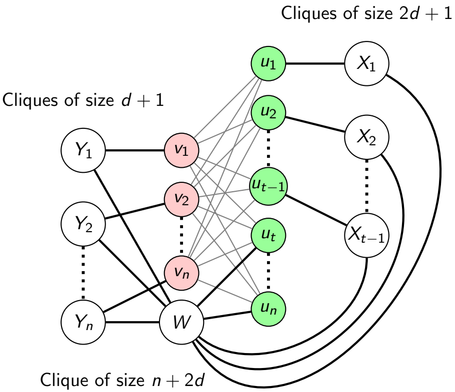


Cliques of size $2d+1$
Cliques of size $d+1$
Clique of size $n+2d$
u_1
X_1
u_2
X_2
u_{t-1}
X_{t-1}
u_t
u_n
Y_1
v_1
Y_2
v_2
v_n
Y_n
W



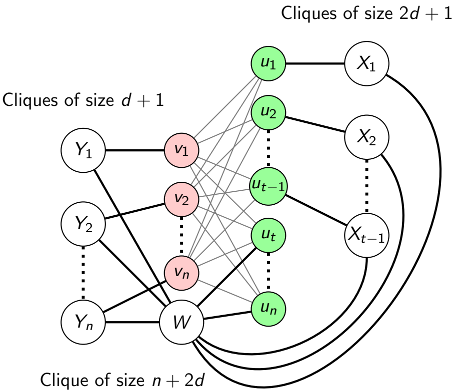
$\Longrightarrow$

- Find an ordering of width $k \leq n + 2d + 1$.



$\Longrightarrow$

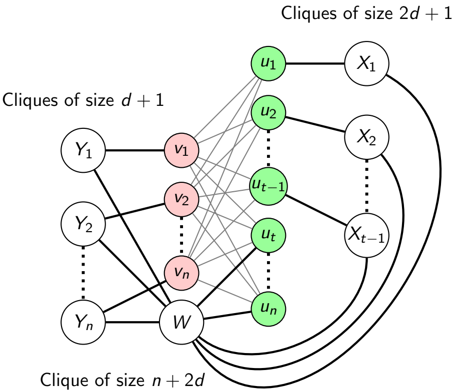
- Find an ordering of width $k \leq n + 2d + 1$.
- Each vertex $v \in V$ has $\deg_{\mathcal{G}'}(v) \leq n + 2d + 1$.



$\Longrightarrow$

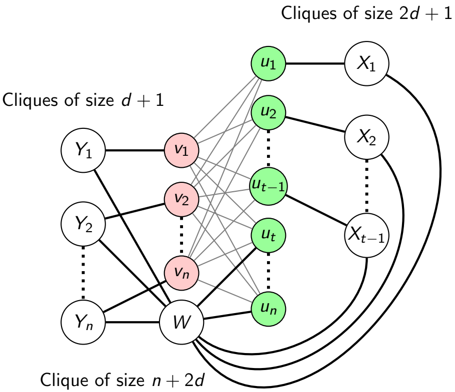
- Find an ordering of width $k \leq n + 2d + 1$.
- Each vertex $v \in V$ has $\deg_{\mathcal{G}'}(v) \leq n + 2d + 1$.
- Given independent set S on V, π has width $\leq n + 2d + 1$.

$\Longleftarrow$



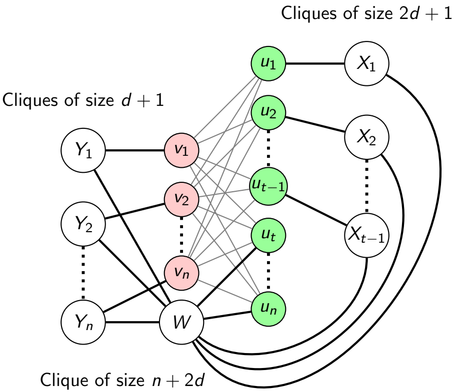
- Given $\pi = (u_1, u_2, \ldots, u_{t-1})$ and $c = t-1$.

$\Longleftarrow$

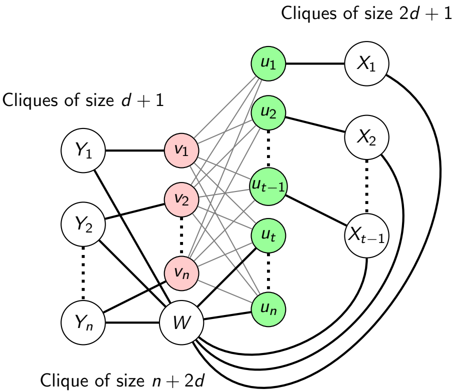


- Given $\pi = (u_1, u_2, \ldots, u_{t-1})$ and $c = t - 1$.
- π cannot contain any from X, Y, W or u_j for $t \leq j \leq n$.

$\Longleftarrow$



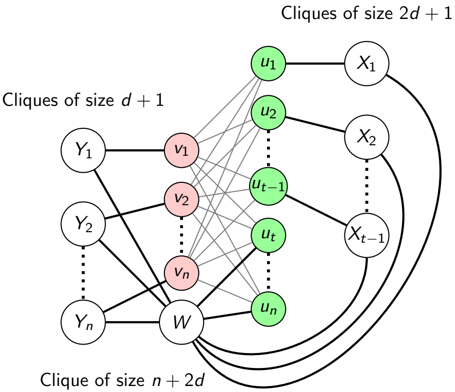
- Given $\pi = (u_1, u_2, \ldots, u_{t-1})$ and $c = t - 1$.
- π cannot contain any from X, Y, W or u_j for $t \leq j \leq n$.
- Eliminating $u_j \implies$ cannot add $v_i \in V$.



$\Longleftarrow$

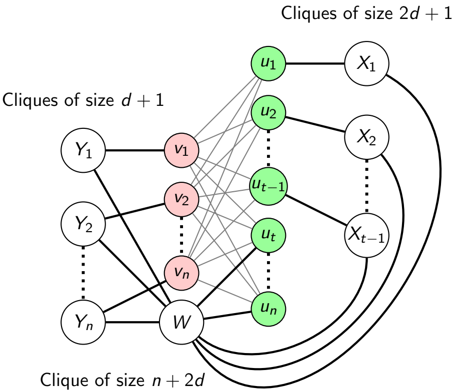
- Given $\pi = (u_1, u_2, \ldots, u_{t-1})$ and $c = t-1$.
- π cannot contain any from X, Y, W or u_j for $t \leq j \leq n$.
- Eliminating $u_j \implies$ cannot add $v_i \in V$.
- Eliminating $v_i \implies$ cannot add u_j.

$\Longleftarrow$



- Given $\pi = (u_1, u_2, \ldots, u_{t-1})$ and $c = t - 1$.
- π cannot contain any from X, Y, W or u_j for $t \leq j \leq n$.
- Eliminating $u_j \implies$ cannot add $v_i \in V$.
- Eliminating $v_i \implies$ cannot add u_j.
- Only $t-1$ suitable vertices of U.

$\Longleftarrow$



- Given $\pi = (u_1, u_2, \ldots, u_{t-1})$ and $c = t - 1$.
- π cannot contain any from X, Y, W or u_j for $t \leq j \leq n$.
- Eliminating $u_j \implies$ cannot add $v_i \in V$.
- Eliminating $v_i \implies$ cannot add u_j.
- Only $t-1$ suitable vertices of U.
- π must only contain V.

Length l Partial Elimination Order

Length-l-Partial-Elimination-Order

Instance: Graph $\mathcal{G}$, integers l and k.
Problem: Does there exist a partial elimination order of $\mathcal{G}$ of length l and width $\leq k$?

LENGTH l PARTIAL ELIMINATION ORDER

Theorem

LENGTH-l-PARTIAL-ELIMINATION-ORDER *is FPT when parameterized by l and k.*

Proof Sketch

- Let $S = \{ v \in V \mid \deg(v) \leq k \}$.

Proof Sketch

- Let $S = \{\, v \in V \mid \deg(v) \le k \,\}$.
- Run greedy algorithm for Independent Set on $\mathcal{G}[S]$.

Proof Sketch

- Let $S = \{ v \in V \mid \deg(v) \leq k \}$.
- Run greedy algorithm for Independent Set on $\mathcal{G}[S]$.
- Find independent set $\mathcal{I}$ of size l.

Proof Sketch

- Let $S = \{ v \in V \mid \deg(v) \leq k \}$.
- Run greedy algorithm for Independent Set on $\mathcal{G}[S]$.
- Find independent set $\mathcal{I}$ of size l.
- ✓ $\mathcal{I}$ has degree $\leq k \implies$ ordering has width $\leq k$.

Proof Sketch

- Let $S = \{ v \in V \mid \deg(v) \leq k \}$.
- Run greedy algorithm for Independent Set on $\mathcal{G}[S]$.
- Find independent set $\mathcal{I}$ of size l.
- $\checkmark$ $\mathcal{I}$ has degree $\leq k \implies$ ordering has width $\leq k$.
- $\times$ Then $|S| \leq (l-1)(k+1)$.

Proof Sketch

- Let $S = \{ v \in V \mid \deg(v) \leq k \}$.
- Run greedy algorithm for Independent Set on $\mathcal{G}[S]$.
- Find independent set $\mathcal{I}$ of size l.
- ✓ $\mathcal{I}$ has degree $\leq k \implies$ ordering has width $\leq k$.
- $\times$ Then $|S| \leq (l-1)(k+1)$.
- Branch for every node $v \in S$: Add it to π, eliminate v from $\mathcal{G}$ solve for $l - 1$.

Proof Sketch

- Let $S = \{ v \in V \mid \deg(v) \leq k \}$.
- Run greedy algorithm for Independent Set on $\mathcal{G}[S]$.
- Find independent set $\mathcal{I}$ of size l.
- ✓ $\mathcal{I}$ has degree $\leq k \implies$ ordering has width $\leq k$.
- $\times$ Then $|S| \leq (l-1)(k+1)$.
- Branch for every node $v \in S$: Add it to π, eliminate v from $\mathcal{G}$ solve for $l-1$.
- Number of branches is

$$\prod_{i=1}^{l} (l-i)(k+1) \implies O^*((l-1)!(k+1)^l).$$

Taking a step back

- Use Length-l-Partial-Elimination-Order to backtrack c vertices and extend π again by $c + 1$ vertices.

Theorem
IC-Treewidth *is fixed-parameter tractable when parameterized by* c *and* k*.*

What about k alone?

Theorem

Length-l-Partial-Elimination-Order *is NP-hard even when* $k = 5$.

What about k alone?

Theorem

Length-l-Partial-Elimination-Order *is NP-hard even when* $k = 5$.

- Reduce from Independent Set on cubic graphs (every node has degree 3).

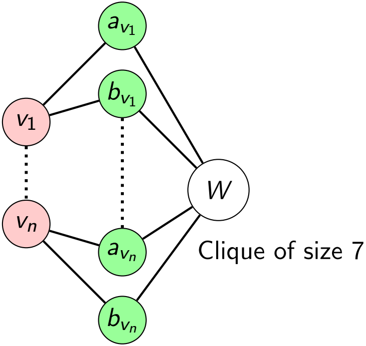
WHAT ABOUT k ALONE?

Theorem

LENGTH-l-PARTIAL-ELIMINATION-ORDER *is NP-hard even when* $k = 5$.

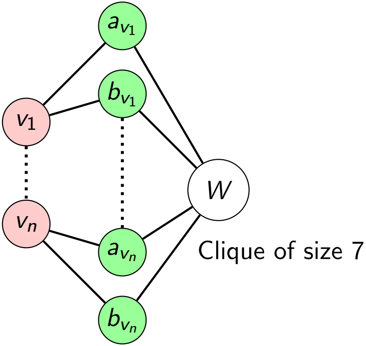
- Reduce from INDEPENDENT SET on cubic graphs (every node has degree 3).
- INDEPENDENT SET on cubic graphs $\implies$ LENGTH-l-PARTIAL-ELIMINATION-ORDER.

Proof



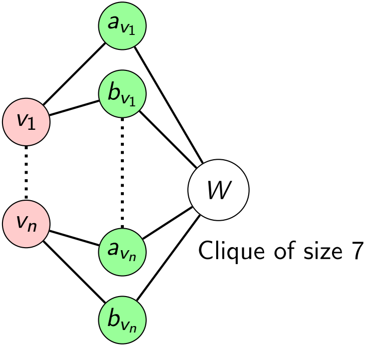
- $l = t, k = 5.$

Proof



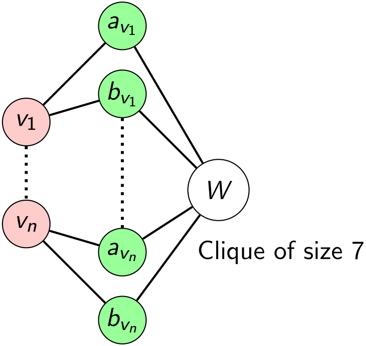
- $l = t, k = 5$.
- π doesn't contain any W or $N(W)$, since width ≤ 5.

Proof



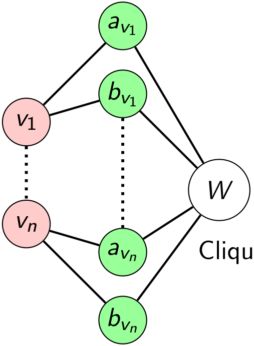
- $l = t, k = 5$.
- π doesn't contain any W or $N(W)$, since width ≤ 5.
- π must contain V.

Proof



- $l = t, k = 5$.
- π doesn't contain any W or $N(W)$, since width ≤ 5.
- π must contain V.
- $(v, u) \in E$, eliminating v increases deg of u.

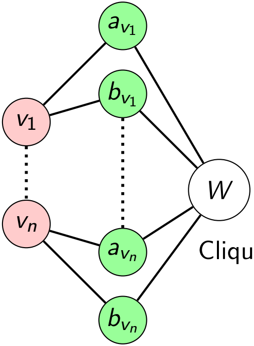
Proof



Clique of size 7

- $l = t, k = 5$.
- π doesn't contain any W or $N(W)$, since width ≤ 5.
- π must contain V.
- $(v, u) \in E$, eliminating v increases deg of u.
- π must form an independent set.

Proof



- $l = t, k = 5$.
- π doesn't contain any W or $N(W)$, since width ≤ 5.
- π must contain V.
- $(v, u) \in E$, eliminating v increases deg of u.
- π must form an independent set.
- Independent set of size $t \iff$ a partial elimination order of width 5 and length $l = t$.

Theorem

The function version of IC-TREEWIDTH *is NP-hard under Turing reductions even when $k = 5$.*

Theorem

The function version of IC-TREEWIDTH *is NP-hard under Turing reductions even when $k = 5$.*

Proof.

- Reduce from LENGTH-l-PARTIAL-ELIMINATION-ORDER when $k = 5$.

$\square$

Theorem

The function version of IC-TREEWIDTH *is NP-hard under Turing reductions even when* $k = 5$*.*

Proof.

- Reduce from LENGTH-l-PARTIAL-ELIMINATION-ORDER when $k = 5$.
- Iteratively solve IC-TREEWIDTH .

□

Theorem

The function version of $\textsc{IC-Treewidth}$ *is NP-hard under Turing reductions even when* $k = 5$*.*

Proof.

- Reduce from $\textsc{Length-}l\textsc{-Partial-Elimination-Order}$ when $k = 5$.
- Iteratively solve $\textsc{IC-Treewidth}$.
- Start with $|\pi| = 0$ and finish with $|\pi| = l - 1$.

$\square$

Summary

IC-Treewidth

Instance: Graph $\mathcal{G}$, integers k and c, partial elimination order π of length l and width $\leq k$.
Problem: Does there exist a partial elimination order π' of length $l+1$ and width $\leq k$ such that π and π' are identical on the first $l-c$ positions.

Parameter	Complexity
c & k	FPT
l	W[1]-hard
k	NP-hard even for $k=5$

Overview

Algorithm

1. Run standard heuristic (e.g. GreedyDegree or GreedyFillIn).

Algorithm

1. Run standard heuristic (e.g. GreedyDegree or GreedyFillIn).
2. Stop if the current vertex has degree $> k$.

Algorithm

1. Run standard heuristic (e.g. GreedyDegree or GreedyFillIn).
2. Stop if the current vertex has degree $> k$.
3. Use FPT algorithm to extend π without increasing k.

Algorithm

1. Run standard heuristic (e.g. GreedyDegree or GreedyFillIn).
2. Stop if the current vertex has degree $> k$.
3. Use FPT algorithm to extend π without increasing k.
4. Continue with heuristic (or until no ordering is found).

Algorithm

1. Run standard heuristic (e.g. GreedyDegree or GreedyFillIn).
2. Stop if the current vertex has degree $> k$.
3. Use FPT algorithm to extend π without increasing k.
4. Continue with heuristic (or until no ordering is found).

Drawback: Need to specify the value of k.

Partial k-trees

- Partial k-trees with n nodes and p percent edges randomly removed.
- Parameter $c = 8$.

			min-degree		min-fill-in		turbo-min-degree		turbo-min-fill-in	
n	k	p	quality	time	quality	time	quality	time	quality	time
250	10	0.20	10.44	0.12	11.42	0.18	10.44	0.10	10.12	0.25
250	10	0.40	10.16	0.10	11.34	0.15	10.16	0.10	10.04	0.21
250	15	0.20	15.60	0.17	16.64	0.27	15.60	0.11	15.34	0.36
250	15	0.40	15.20	0.14	16.38	0.22	15.20	0.12	15.12	0.29
250	20	0.20	20.64	0.22	21.96	0.37	20.64	0.13	20.32	0.49
250	20	0.40	20.22	0.18	21.60	0.30	20.22	0.16	20.08	0.39
500	10	0.20	10.72	0.36	11.72	0.59	10.72	0.15	10.24	0.96
500	10	0.40	10.32	0.28	11.64	0.44	10.32	0.21	10.26	0.79
500	15	0.20	15.94	0.63	16.86	1.09	15.94	0.20	15.70	1.62
500	15	0.40	15.32	0.46	17.04	0.78	15.32	0.33	15.20	1.18
500	20	0.20	20.88	0.94	22.18	1.67	20.88	0.27	20.82	2.37
500	20	0.40	20.32	0.67	22.08	1.17	20.32	0.49	20.38	1.67
1000	10	0.20	10.90	1.75	11.94	3.11	10.90	0.33	10.64	4.70
1000	10	0.40	10.56	1.29	11.98	2.18	10.56	0.64	10.20	3.65
1000	15	0.20	16.04	3.46	17.20	6.71	16.04	0.41	15.94	8.75
1000	15	0.40	15.58	2.44	17.26	4.40	15.58	1.26	15.46	6.38
1000	20	0.20	21.16	5.34	22.38	10.24	21.16	0.24	21.54	12.14
1000	20	0.40	20.50	3.76	22.56	6.90	20.50	2.01	20.34	8.94

DIMACS Graph Coloring Networks

- Parameter $c = 8$ (DSJC1000.5 and DSJC500.9 we used $c = 6$).
- Target width: Run the standard heuristics and try to improve their width by up to 6%.

id	n	m	tw	min-degree quality	min-degree time	min-fill-in quality	min-fill-in time	turbo quality	turbo time
queen7_7	49	952	35	37	0.056	37	0.075	36	0.104
queen8_8	64	1456	46	50	0.081	48	0.099	47	0.543
queen9_9	81	2112	59	64	0.100	63	0.128	62	0.266
queen11_11	121	3960	89	97	0.231	95	0.283	93	12.49
queen13_13	169	6656	125	140	0.610	137	0.808	135	36.67
queen14_14	196	8372	143	164	1.060	160	1.372	159	95.08
myciel4	23	71	10	11	0.011	11	0.016	10	4.62
le450_5b	450	5734	309	316	15.12	318	19.42	311	500.3
le450_15c	450	16680	372	376	21.35	376	26.44	372	240.6
le450_25d	450	17425	349	367	20.48	363	25.18	360	584.4
DSJC1000.5	1000	499652	977	980	642	978	705	977	5429
DSJC125.1	125	1472	64	67	0.144	66	0.170	65	54.885
DSJC250.1	250	6436	176	180	1.835	177	2.300	176	264.46
DSJC500.1	500	24916	409	413	31.086	411	43.048	410	2089.77
DSJC500.5	500	125248	479	481	41.024	482	48.481	479	19467.95
DSJC500.9	500	224874	492	493	45	493	47	492	2662

Overview

Tree Decompositions

IC-Treewidth

Turbocharged Heuristics

Conclusion
- Summary
- Open problems

Summary

- IC-Treewidth models 'local search' for treewidth heuristics.
- Use heuristics and only expensive computation when we get stuck.
- Prototype implementation shows we can improve quality over greedy heuristics with some trade-off in running time.

Open problems

- How to chose better values for the backtrack length c and width k?
- A more efficient FPT algorithm?

Implementation

Code available at github.com/mfjones/pace2016.