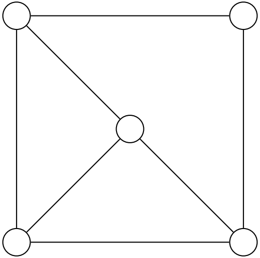
Maximum Independent Set in H-Free Graphs

Édouard Bonnet, Nicolas Bousquet, Pierre Charbit, Stéphan Thomassé, and Rémi Watrigant

Séminaire Graphes et Optimisation, LaBRI, Bordeaux, September 21st, 2018

Independent Set

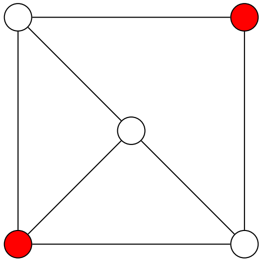
Problem: Given a graph



and an integer k: Is there an independent set of size at least k?

Independent Set

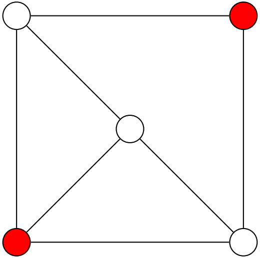
Problem: Given a graph



and an integer k: Is there an independent set of size at least k?

Independent Set

Problem: Given a graph

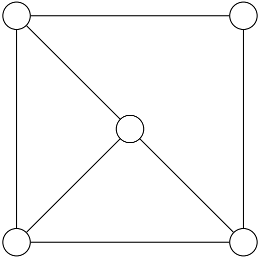


and an integer k: Is there an independent set of size at least k?

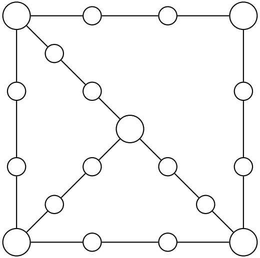
NP-complete even in graphs with maximum degree 3.

What about on graphs excluding an induced subgraph H? (called H-free graphs)

NP-hardness cases [Alekseev '82]

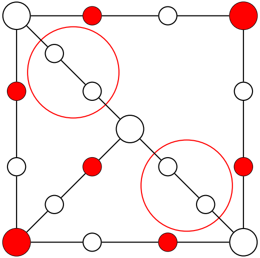


NP-hardness cases [Alekseev '82]



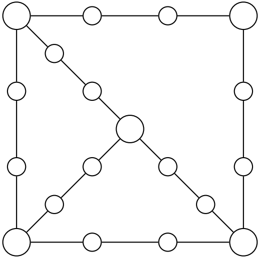
Subdivide every edge twice

NP-hardness cases [Alekseev '82]



Subdivide every edge twice

NP-hardness cases [Alekseev '82]

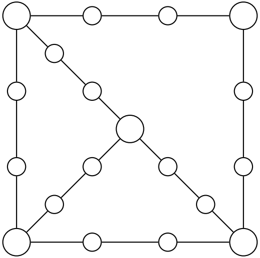


Subdivide every edge any fixed even number of times

This reduction + NP-hardness on graphs of degree 3 ⇒

NP-hardness for graphs of degree 3, with arbitrarily large girth and distance between two vertices with degree 3 (*branching vertices*).

NP-hardness cases [Alekseev '82]



This reduction + NP-hardness on graphs of degree 3 ⇒

NP-hardness for graphs of degree 3, with arbitrarily large girth and distance between two vertices with degree 3 (*branching vertices*).

The constructed graph is H-free except if H is...

P/NP-complete status of MIS on H-free graphs

For H connected:

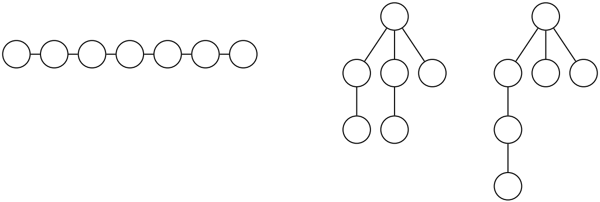
- NP-complete, if H is not a path or a subdivided claw ($K_{1,3}$)
- in P, if H is a path on up to 6 vertices
- in P, if H is a claw with one edge subdivided once
- For other H, the problem is open

P/NP-complete status of MIS on H-free graphs

For H connected:

- NP-complete, if H is not a path or a subdivided claw ($K_{1,3}$)
- in P, if H is a path on up to 6 vertices
- in P, if H is a claw with one edge subdivided once
- For other H, the problem is open

Minimal open cases:



Other dichotomies

The polynomial algorithms for P_5-free and then P_6-free graphs use tools that cannot generalize to P_8-free graphs and beyond.

Understanding P_t-free graphs is a challenge

Other dichotomies

The polynomial algorithms for P_5-free and then P_6-free graphs use tools that cannot generalize to P_8-free graphs and beyond.

Understanding P_t-free graphs is a challenge

there are other goodies/baddies partition:

- PTAS/APX-hard
- SUBEXP/ETH-hard
- **FPT/W[1]-hard**

Parameterized complexity

Fixed-Parameter Tractable (FPT) algorithm:
in time $f(k)n^{O(1)}$ with

- n, the size of the instance,
- k, a parameter such as the solution size, and
- f, any computable function.

Parameterized complexity

Fixed-Parameter Tractable (FPT) algorithm:
in time $f(k)n^{O(1)}$ with

- n, the size of the instance,
- k, a parameter such as the solution size, and
- f, any computable function.

Example:

- Vertex Cover has a simple $2^k n^{O(1)}$-algorithm
- Independent Set is W[1]-hard (hence unlikely FPT)

Convenient definition of W[1]-hard for our purpose:
As hard as Independent Set for FPT reductions

FPT reductions

Reduction from (Π, k) to (Π', k') taking FPT time and such that $\mathbf{k' = g(k)}$ **for a computable function g**.

FPT reductions

Reduction from (Π, k) to (Π', k') taking FPT time and such that $\mathbf{k' = g(k)}$ **for a computable function g**.

Are the following FPT-reductions?

- The "subdividing the edges twice" trick that we saw?
- Complementing the graph, from MIS to CLIQUE?
- $(G, k) \mapsto (G, n-k)$, from MIS to VERTEX COVER?

FPT reductions

Reduction from (Π, k) to (Π', k') taking FPT time and such that $\mathbf{k' = g(k)}$ **for a computable function g**.

Are the following FPT-reductions?

- The "subdividing the edges twice" trick that we saw? No
- Complementing the graph, from MIS to CLIQUE?
- $(G, k) \mapsto (G, n - k)$, from MIS to VERTEX COVER?

FPT reductions

Reduction from (Π, k) to (Π', k') taking FPT time and such that $\mathbf{k' = g(k)}$ **for a computable function g**.

Are the following FPT-reductions?

- The "subdividing the edges twice" trick that we saw? No
- Complementing the graph, from MIS to CLIQUE? Yes
- $(G, k) \mapsto (G, n - k)$, from MIS to VERTEX COVER?

FPT reductions

Reduction from (Π, k) to (Π', k') taking FPT time and such that $\mathbf{k' = g(k)}$ **for a computable function g**.

Are the following FPT-reductions?

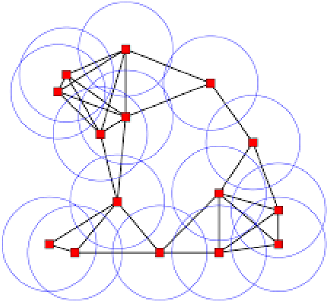
- The "subdividing the edges twice" trick that we saw? No
- Complementing the graph, from MIS to Clique? Yes
- $(G, k) \mapsto (G, n - k)$, from MIS to Vertex Cover? No

Why MIS and why forbidden induced subgraphs?

- some hard problems like Dominating Set are almost indifferent to forbidding induced subgraphs
- for subgraphs or minors, the dichotomy would be trivial
- can shed light on other hereditary classes

Why MIS and why forbidden induced subgraphs?

- some hard problems like DOMINATING SET are almost indifferent to forbidding induced subgraphs
- for subgraphs or minors, the dichotomy would be trivial
- can shed light on other hereditary classes



Known results

- FPT for H on at most 4 vertices but C_4 [Dabrowski et al. '12]
- MIS is W[1]-hard in $K_{1,4}$-free graphs [Hermelin et al. '14]

Why is MIS FPT in K_r-free graphs?[1]

[1] This is why the question is not interesting for subgraphs and minors

Known results

- FPT for H on at most 4 vertices but C_4 [Dabrowski et al. '12]
- MIS is W[1]-hard in $K_{1,4}$-free graphs [Hermelin et al. '14]

Why is MIS FPT in K_r-free graphs?[1]

Every K_r-free graphs has either:

- at most Ramsey(k,r) $\approx k^{r-1}$ vertices $\rightarrow$ brute-force is FPT
- an independent set of size k $\rightarrow$ answer YES

[1] This is why the question is not interesting for subgraphs and minors

Our current goal

Let's try to remove the C_4s with an FPT reduction

First thoughts

k-Multicolored Independent Set is W[1]-hard

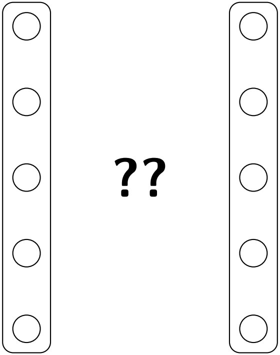
Instances whose vertex-set is partitioned into k cliques

First thoughts

k-Multicolored Independent Set is W[1]-hard

Instances whose vertex-set is partitioned into k cliques

What should we avoid between the cliques?

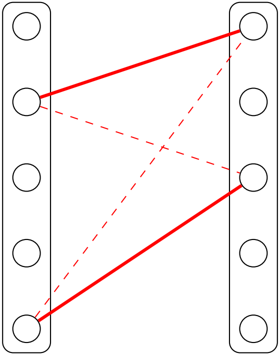


First thoughts

k-Multicolored Independent Set is W[1]-hard

Instances whose vertex-set is partitioned into k cliques

What should we avoid between the cliques? $2K_2$

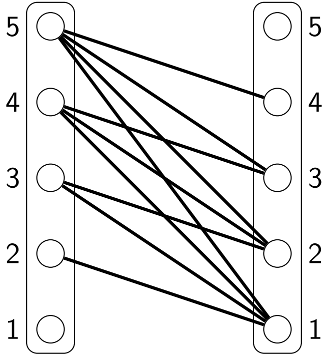


First thoughts

k-Multicolored Independent Set is W[1]-hard

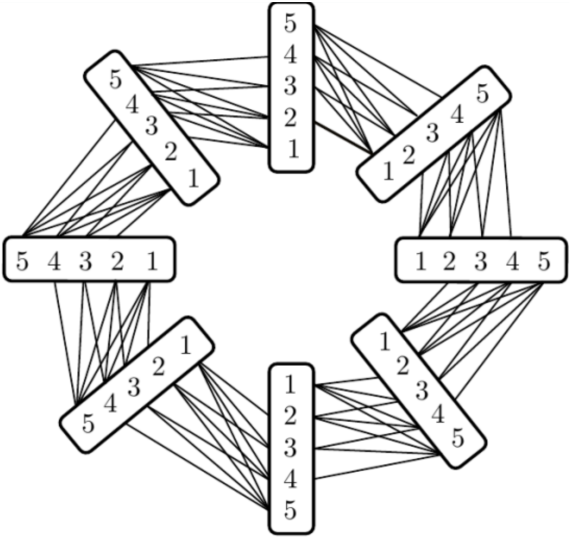
Instances whose vertex-set is partitioned into k cliques

What should we avoid between the cliques? $2K_2$



Half-graphs

"Cycle" of half graphs



Two inequalities enforce the equality

Grid Tiling

Input: $k \times k$ grid of cells containing pairs over $[n]^2$

(1,1) (3,1) (2,4)	(5,1) (1,4) (5,3)	(1,1) (2,4) (3,3)
(2,2) (1,4)	(3,1) (1,2)	(2,2) (2,3)
(1,3) (2,3) (3,3)	(1,1) (1,3)	(2,3) (5,3)

⇒

(1,1) (3,1) (2,4)	(5,1) (1,4) (5,3)	(1,1) (2,4) (3,3)
(2,2) (1,4)	(3,1) (1,2)	(2,2) (2,3)
(1,3) (2,3) (3,3)	(1,1) (1,3)	(2,3) (5,3)

Example with $k = 3$ cliques/color classes and $n = 5$

Grid Tiling

Input: $k \times k$ grid of cells containing pairs over $[n]^2$

(1,1) (3,1) (2,4)	(5,1) (1,4) (5,3)	(1,1) (2,4) (3,3)
(2,2) (1,4)	(3,1) (1,2)	(2,2) (2,3)
(1,3) (2,3) (3,3)	(1,1) (1,3)	(2,3) (5,3)

(1,1) (3,1) (2,4)	(5,1) (1,4) (5,3)	(1,1) (2,4) (3,3)
(2,2) (1,4)	(3,1) (1,2)	(2,2) (2,3)
(1,3) (2,3) (3,3)	(1,1) (1,3)	(2,3) (5,3)

Example with $k = 3$ cliques/color classes and $n = 5$

Output: select one pair per cell so that

- columns agree on the first coordinate
- rows agree on the second coordinate

Grid Tiling w.r.t the number of cells is W[1]-hard

			$(v_j, \cdot)$	
$(\cdot, v_i)$	(v_i, v_i)	$(\cdot, v_i)$	(v_j, v_i)	$(\cdot, v_i)$
			$(v_j, \cdot)$	
			(v_j, v_j)	
			$(v_j, \cdot)$	

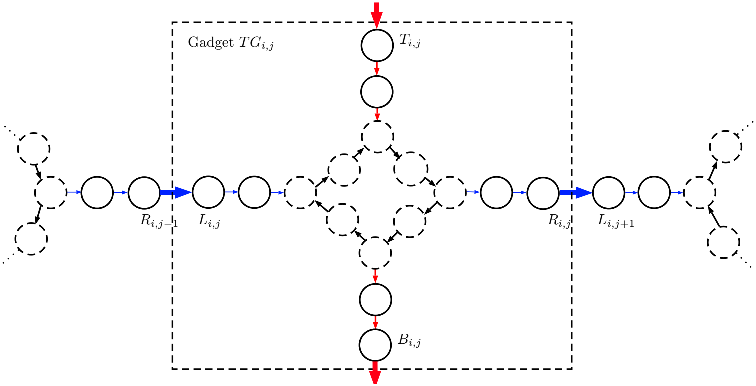
Grid Tiling w.r.t the number of cells is W[1]-hard

			$(v_j, \cdot)$	
$(\cdot, v_i)$	(v_i, v_i)	$(\cdot, v_i)$	(v_j, v_i)	$(\cdot, v_i)$
			$(v_j, \cdot)$	
			(v_j, v_j)	
			$(v_j, \cdot)$	

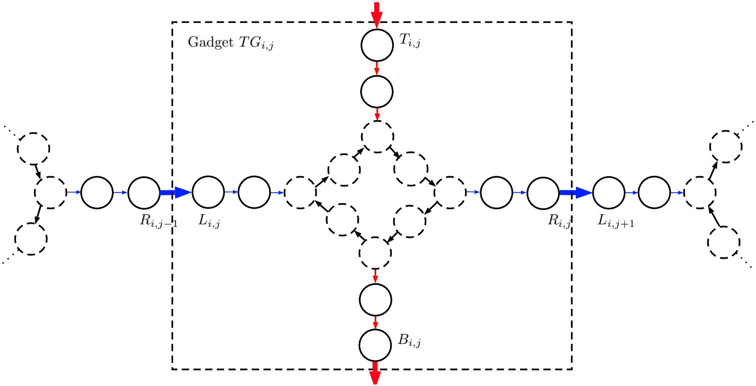
The same with inequalities has the same lower bound

Useful for geometric problems such as Packing Unit Disks

Avoiding C_4 with half graphs everywhere



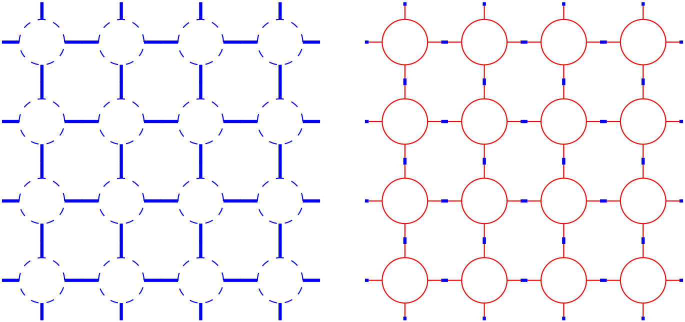
Avoiding C_4 with half graphs everywhere



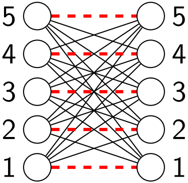
Simultaneously:

- no $C_4, C_5, \ldots, C_s$
- no $K_{1,4}$
- no tree with two branching vertices

Two variants of the reduction



Variants with half-graphs in blue and antimatchings in red

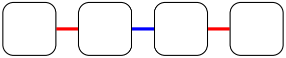


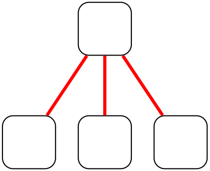
antimatching

FPT candidates

H should be chordal and

- either *a path of cliques* with *simple connections between adjacent cliques*
- or *a subdivided claw of cliques* with *very simple connections between adjacent cliques*





bipartite complete except possibly one edge

half-graph

What about algorithms now?

Modular FPT reduction which traps many hard cases.

Generic algorithmic technique for the remaining cases?

What about algorithms now?

Modular FPT reduction which traps many hard cases.

Generic algorithmic technique for the remaining cases?

So far, we did not get something very unified.

- Many H-specific arguments
- A handful of transversal tricks/ideas

What about algorithms now?

Modular FPT reduction which traps many hard cases. 

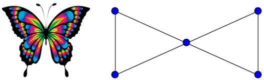
Generic algorithmic technique for the remaining cases?

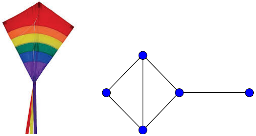
So far, we did not get something very unified.

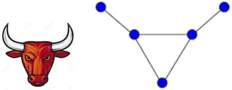
- Many H-specific arguments
- A handful of transversal tricks/ideas

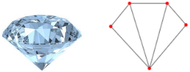
Maybe not so surprising:
notably open for P_t-free graphs
entire papers have been dedicated to -free graphs

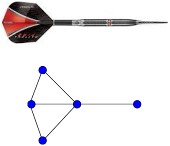
Some candidates on 5 vertices



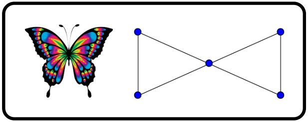


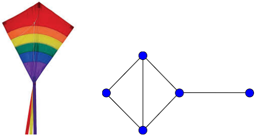


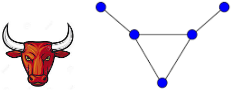


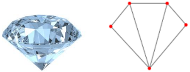


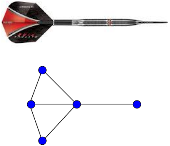
Some candidates on 5 vertices











Two tricks to catch the butterfly

Trick 1: we can guess the solution on any subset of $f(k)$ vertices

Two tricks to catch the butterfly

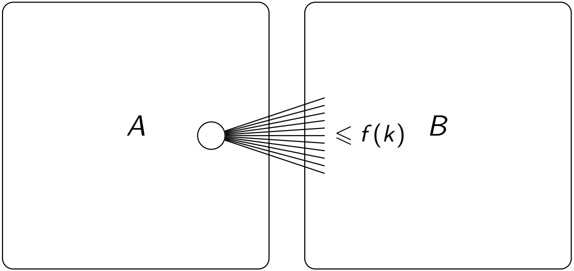
Trick 1: we can guess the solution on any subset of $f(k)$ vertices

We just try all the $2^{f(k)}$ possibilities

Two tricks to catch the butterfly

Trick 1: we can guess the solution on any subset of $f(k)$ vertices
We just try all the $2^{f(k)}$ possibilities

Trick 2: We can progress if we have the following



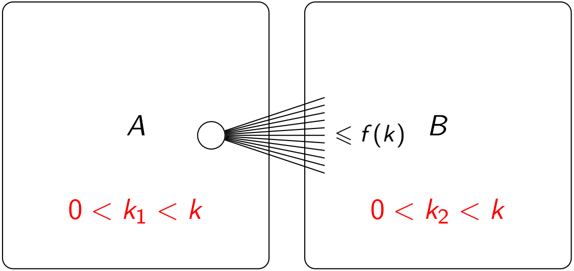
with

- A and B intersecting the solution
- all the vertices in A have at most $f(k)$ neighbors in B

Two tricks to catch the butterfly

Trick 1: we can guess the solution on any subset of $f(k)$ vertices
We just try all the $2^{f(k)}$ possibilities

Trick 2: We can progress if we have the following

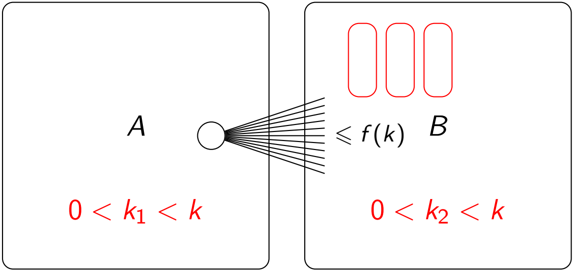


We guess how many vertices a solution contains in A and B

Two tricks to catch the butterfly

Trick 1: we can guess the solution on any subset of $f(k)$ vertices
We just try all the $2^{f(k)}$ possibilities

Trick 2: We can progress if we have the following

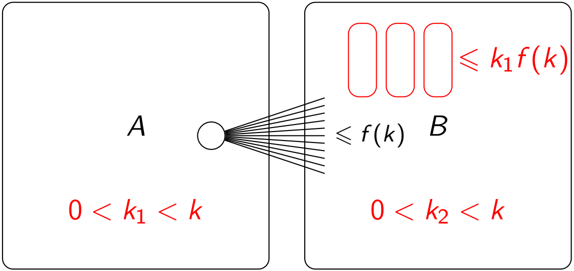


We extract independent sets of size k_2 in $G[B]$

Two tricks to catch the butterfly

Trick 1: we can guess the solution on any subset of $f(k)$ vertices
We just try all the $2^{f(k)}$ possibilities

Trick 2: We can progress if we have the following

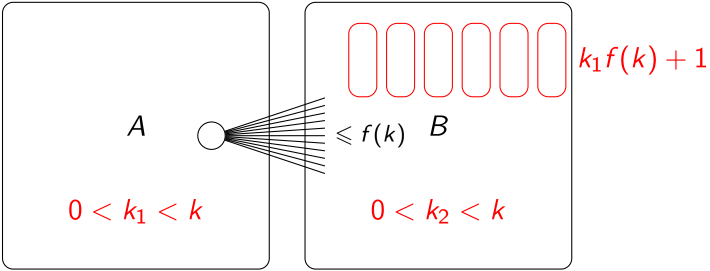


If this stops before $k_1 f(k) + 1$ are extracted, use Trick 1

Two tricks to catch the butterfly

Trick 1: we can guess the solution on any subset of $f(k)$ vertices
We just try all the $2^{f(k)}$ possibilities

Trick 2: We can progress if we have the following

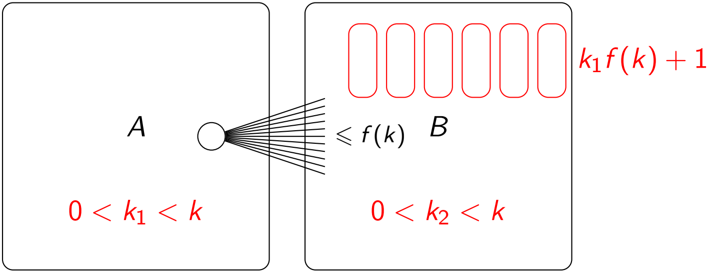


If we can extract $k_1 f(k) + 1$ of them, we stop there

Two tricks to catch the butterfly

Trick 1: we can guess the solution on any subset of $f(k)$ vertices
We just try all the $2^{f(k)}$ possibilities

Trick 2: We can progress if we have the following

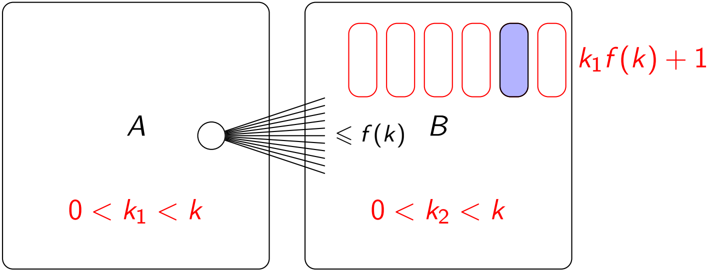


What does this achieve?

Two tricks to catch the butterfly

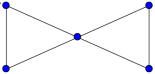
Trick 1: we can guess the solution on any subset of $f(k)$ vertices
We just try all the $2^{f(k)}$ possibilities

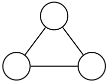
Trick 2: We can progress if we have the following



Any independent set of size k_1 in $G[A]$ can be completed

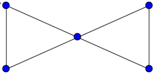
FPT algorithm in butterfly-free graph

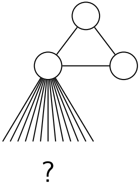




Let us consider a triangle and its neighbors

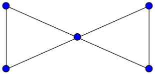
FPT algorithm in butterfly-free graph





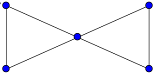
Can there be very many vertices attached to a single vertex?

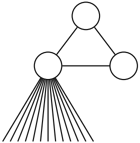
FPT algorithm in butterfly-free graph



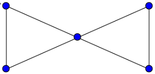
Less than k. Otherwise: easy solution or butterfly

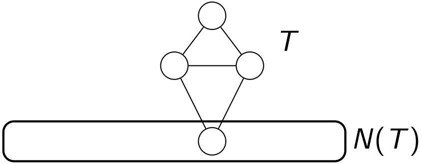
FPT algorithm in butterfly-free graph





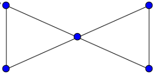
We use Trick 1 to get rid of those particular neighbors

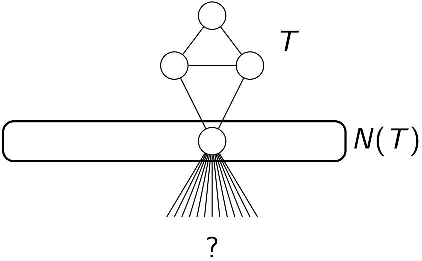
FPT algorithm in butterfly-free graph  



Now, all the vertices in $N(T)$ have at least two neighbors in T

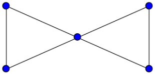
FPT algorithm in butterfly-free graph

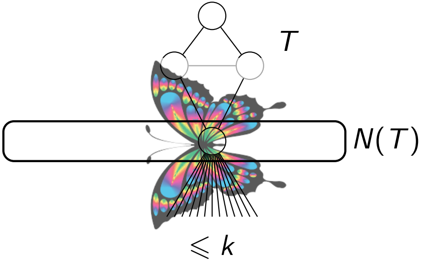




Can they have many neighbors in the rest of the graph?

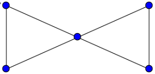
FPT algorithm in butterfly-free graph

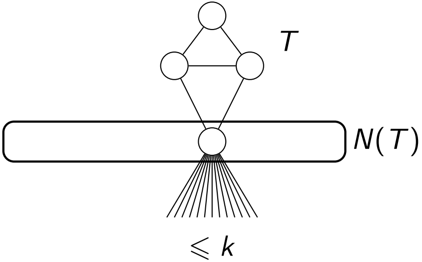




No, less than k; otherwise easy solution or butterfly

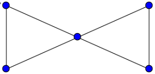
FPT algorithm in butterfly-free graph

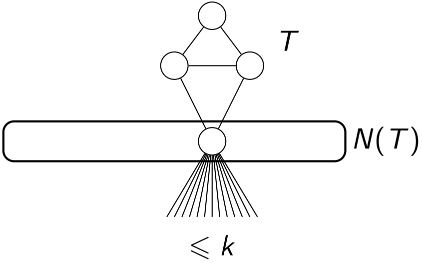




A solution intersects $T \cap N(T)$ (why?)

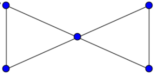
FPT algorithm in butterfly-free graph

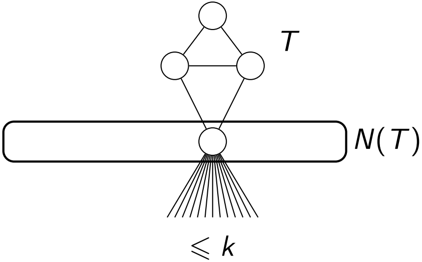




Either it also intersects $\overline{T \cap N(T)}$, and we conclude with Trick 2

FPT algorithm in butterfly-free graph





Or not. And we solve $G[T \cap N(T)]$ since it is $4K_2$-free (Alekseev)

Results and perspectives

FPT algorithms when

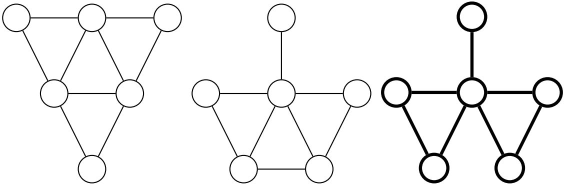
- H is a clique minus a bipartite complete graph
  (can be seen as a P_3 of cliques, generalizes the butterfly)
- H is the union of cliques (parameterized version of Alekseev)
- H is a clique minus a triangle ($K_r \setminus K_4$ contrains a $K_{1,4}$)
- H, candidate on 5 vertices: crown, gem, kite, $\overline{P}$, dart, cricket

Results and perspectives

FPT algorithms when

- H is a clique minus a bipartite complete graph
  (can be seen as a P_3 of cliques, generalizes the butterfly)
- H is the union of cliques (parameterized version of Alekseev)
- H is a clique minus a triangle ($K_r \setminus K_4$ contrains a $K_{1,4}$)
- H, candidate on 5 vertices: crown, gem, kite, $\overline{P}$, dart, cricket

W[1]-hardness cases with a third reduction:

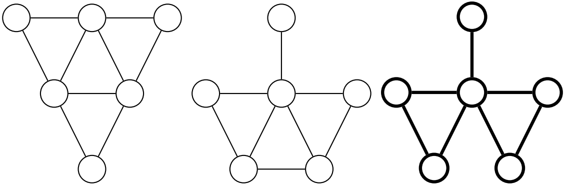


Results and perspectives

FPT algorithms when

- H is a clique minus a bipartite complete graph (can be seen as a P_3 of cliques, generalizes the butterfly)
- H is the union of cliques (parameterized version of Alekseev)
- H is a clique minus a triangle ($K_r \setminus K_4$ contrains a $K_{1,4}$)
- H, candidate on 5 vertices: crown, gem, kite, $\overline{P}$, dart, cricket

W[1]-hardness cases with a third reduction:



Mainly left with "path of cliques"