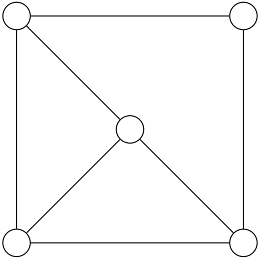
When Maximum Stable Set can be solved in FPT time

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

ISAAC 2019, Shanghai, December 11th

Independent Set

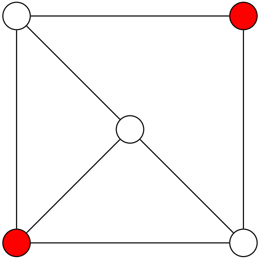
Problem: Given a graph



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

Independent Set

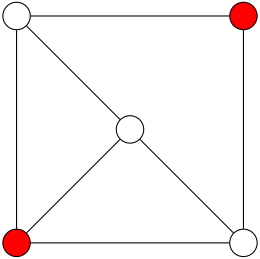
Problem: Given a graph



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

Independent Set

Problem: Given a graph

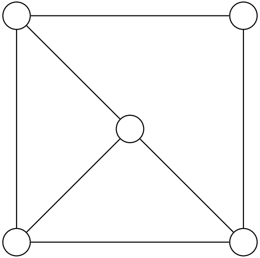


and an integer k: Is there an independent set of size at least k?

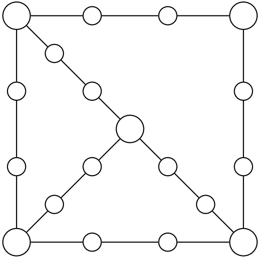
NP-complete even in subcubic graphs

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

NP-hard cases [Alekseev '82, Poljak '73]

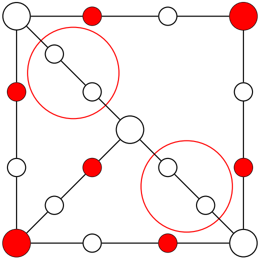


NP-hard cases [Alekseev '82, Poljak '73]



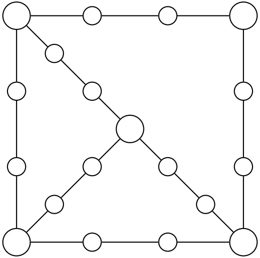
Subdivide every edge twice

NP-hard cases [Alekseev '82, Poljak '73]



Subdivide every edge twice

NP-hard cases [Alekseev '82, Poljak '73]



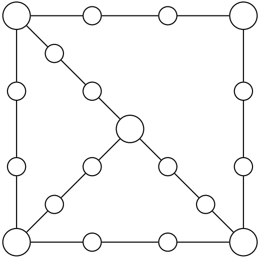
Subdivide every edge any even number of times

This reduction + NP-hardness on subcubic graphs ⇒

NP-hardness for subcubic graphs, with arbitrarily large

- girth, and
- distance between two vertices with degree at least 3.

NP-hard cases [Alekseev '82, Poljak '73]



Subdivide every edge any even number of times

This reduction + NP-hardness on subcubic graphs ⇒

NP-hardness for subcubic graphs, with arbitrarily large

- girth, and
- distance between two vertices with degree at least 3.

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

P/NP-hard dichotomy

For H connected:

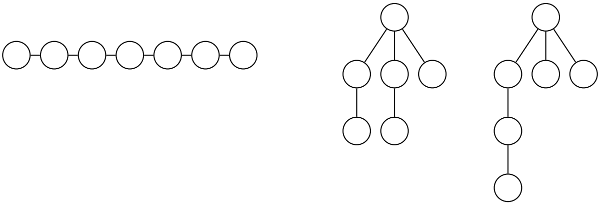
- NP-hard, if H is not a path or a subdivided claw ($K_{1,3}$)
- in P, if H is a path on up to 6 vertices
- in P, if H is a claw with one edge subdivided once
- For other H, the problem is open

P/NP-hard dichotomy

For H connected:

- NP-hard, 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

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

Ultimate goal: Dichotomy classification

For every H,

- if *easy(H)* then Independent Set is FPT on H-free graphs,
- otherwise it is W[1]-hard.

Ultimate goal: Dichotomy classification

For every H,

- if *easy(H)* then Independent Set is FPT on H-free graphs,
- otherwise it is W[1]-hard.

For the P/NP-hard dichotomy, we have at least a natural candidate for the criterion *easy(H)*...

Known results before 2018

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

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

Known results before 2018

Why is INDEPENDENT SET 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

Known results before 2018

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

Every K_r-free graphs has either:

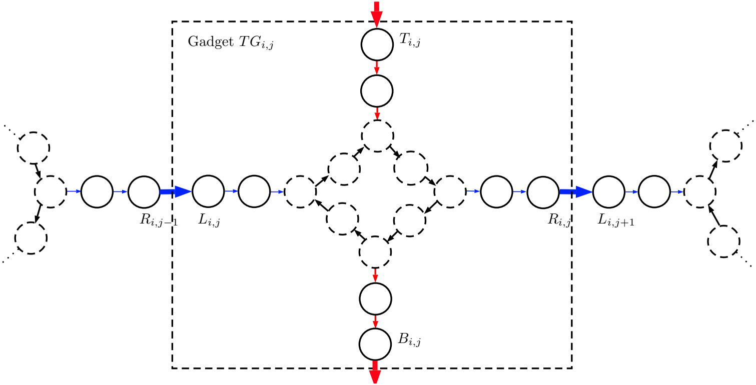
- at most Ramsey(k,r) $\approx k^{r-1}$ vertices $\rightarrow$ brute-force is FPT
- an independent set of size k $\rightarrow$ answer YES

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

---

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

BBCTW '18: W[1]-hardness reduction



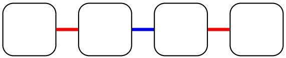
Simultaneously avoiding as induced subgraph:

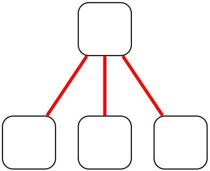
- $C_4, C_5, \ldots, C_s$
- $K_{1,4}$
- any tree with two degree-3+ vertices at distance at most s

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*

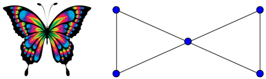


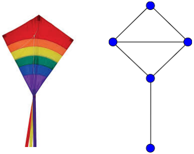


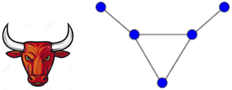
bipartite complete except possibly one edge

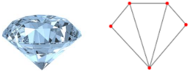
half-graph

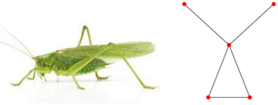
Candidates on 5 vertices

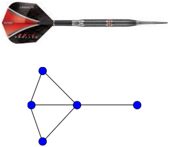




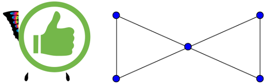


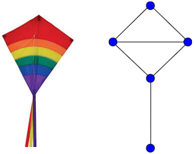


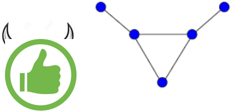


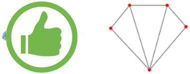


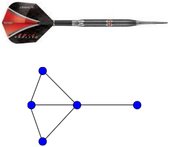
Candidates on 5 vertices



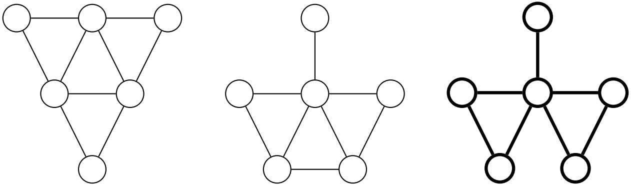




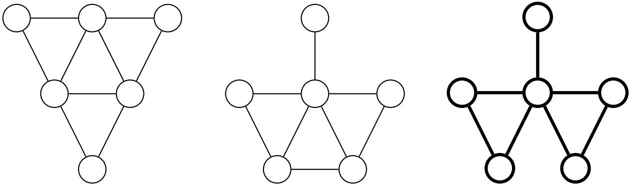




Other W[1]-hard cases due to a variant of the reduction

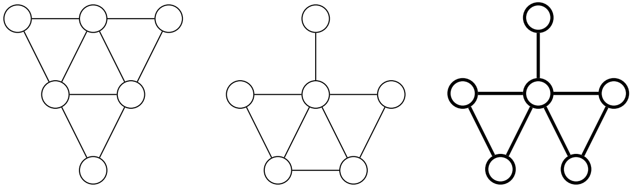


Other W[1]-hard cases due to a variant of the reduction



Mainly left with "path of cliques"

Other W[1]-hard cases due to a variant of the reduction



Mainly left with "path of cliques"

$P(a_1, a_2, \ldots, a_s) =$ graph obtained from P_s by replacing the i-th vertex by a clique of size a_i.

Ambitious conjecture

Conjecture: Independent Set is FPT in $P(t, t, \ldots, t)$-free.

Ambitious conjecture

Conjecture: INDEPENDENT SET is FPT in $P(t, t, \ldots, t)$-free.

- We showed it for $P(t, t, t)$ in the 2018 paper.
- No *easy* argument for $P(1, 1, 1, 1, 1)$ and P_7 is open.

Ambitious conjecture

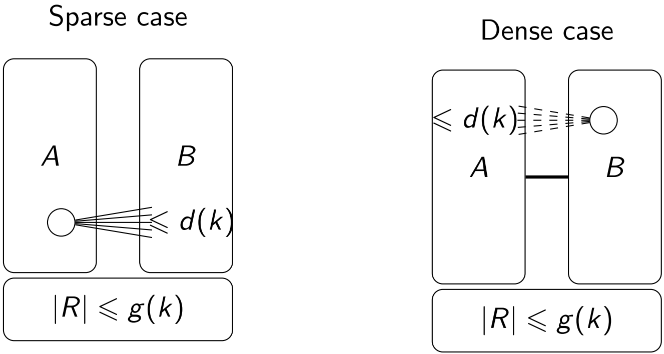
Conjecture: INDEPENDENT SET is FPT in $P(t, t, \ldots, t)$-free.

- We showed it for $P(t, t, t)$ in the 2018 paper.
- No *easy* argument for $P(1, 1, 1, 1, 1)$ and P_7 is open.

Our main contribution:
An FPT algorithm for INDEPENDENT SET in $P(1, t, t, t)$-free.

Our main new ingredient: introducing *co-graphs with parameterized noise*, and associated FPT subroutines

Co-graphs with parameterized noise

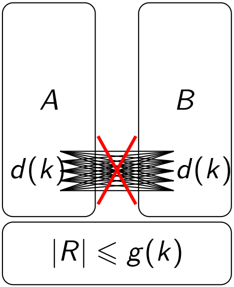


Tripartition (A, B, R) of the graph, where R is small, and:

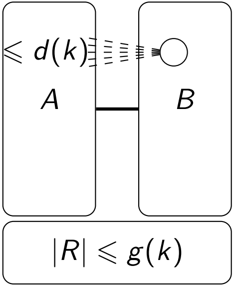
- Sparse case: the degree to the other side is small
- Dense case: the co-degree to the other side is small

Co-graphs with parameterized noise

Sparse case



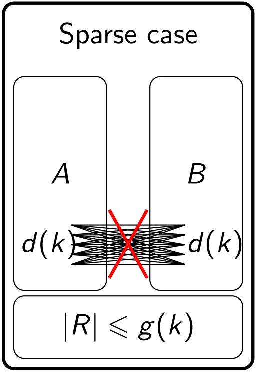
Dense case

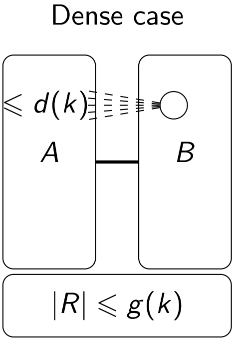


Tripartition (A, B, R), where R is small, and:

- Sparse case: no large transversal biclique
- Dense case: the co-degree to the other side is small

Co-graphs with parameterized noise



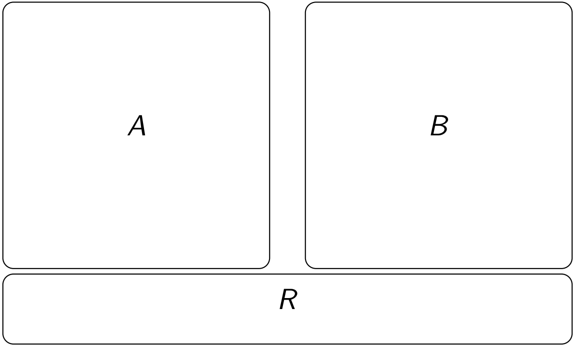


Tripartition (A, B, R), where R is small, and:

- Sparse case: no large transversal biclique
- Dense case: the co-degree to the other side is small

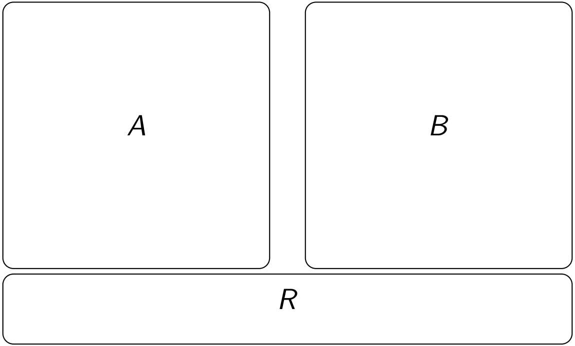
An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A,B]$

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



An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A, B]$

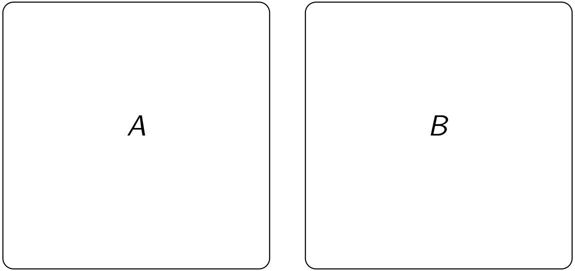
Trick 1: we can guess the solution on any subset of $f(k)$ vertices
We just try all the $2^{f(k)}$ possibilities



An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A, B]$

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

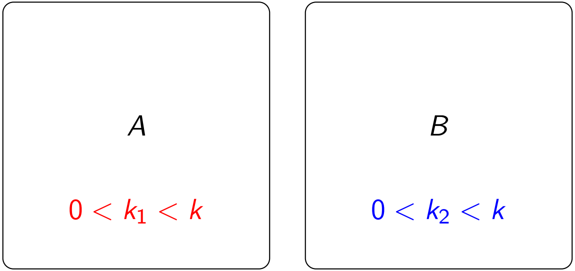
Trick 2: Excavating a sequence of solutions



An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A, B]$

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: Excavating a sequence of solutions

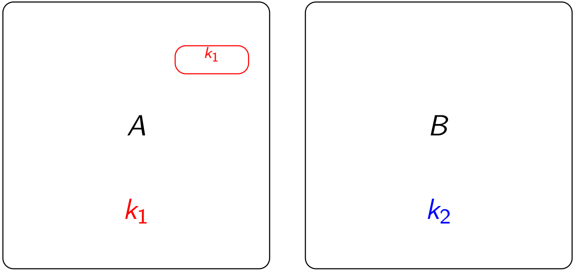


We guess how many vertices a solution contains in A and B

An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A,B]$

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: Excavating a sequence of solutions

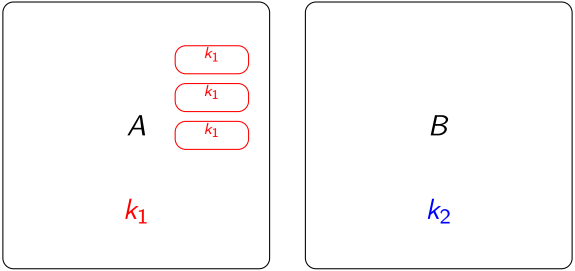


We extract independent sets of size k_1 in $G[A]$

An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A,B]$

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: Excavating a sequence of solutions

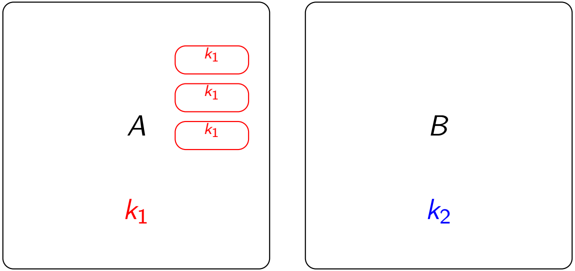


We extract independent sets of size k_1 in $G[A]$

An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A, B]$

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: Excavating a sequence of solutions

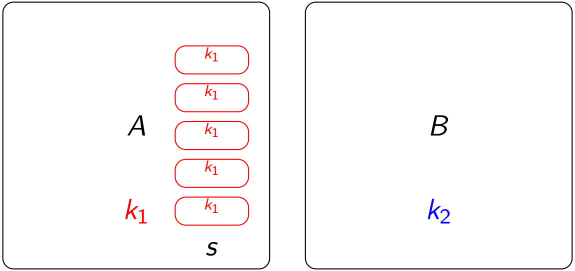


If this process stops quickly, use Trick 1

An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A, B]$

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: Excavating a sequence of solutions

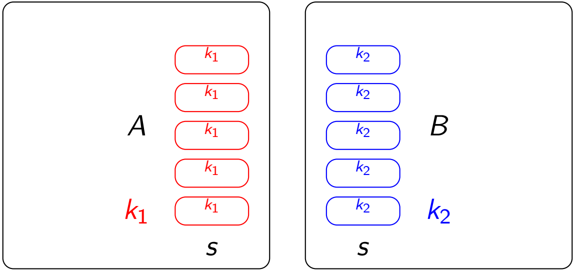


If it goes on, we stop after $s \gg k, d$ steps

An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A, B]$

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: Excavating a sequence of solutions

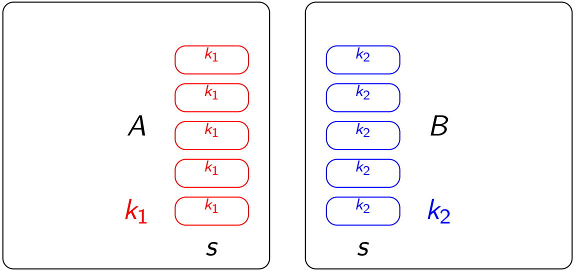


We do the same with independent sets of size k_2 in $G[B]$

An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A, B]$

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: Excavating a sequence of solutions

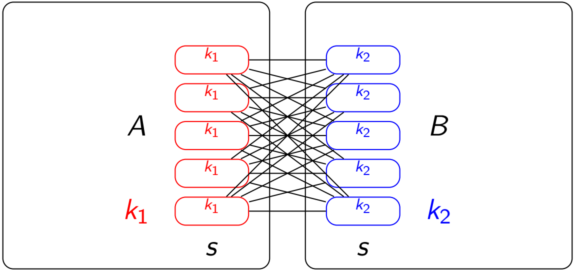


Solution! Except if there is at least one edge between each pair

An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A, B]$

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: Excavating a sequence of solutions

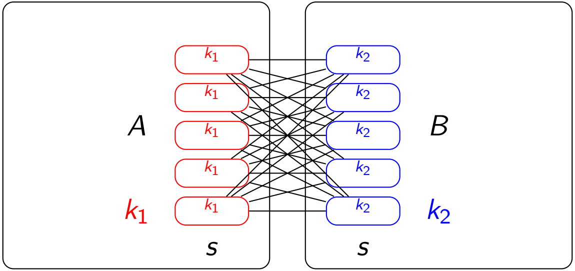


That would be s^2 edges on sk vertices

An FPT subroutine for the sparse case: no $K_{d,d}$ in $G[A,B]$

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: Excavating a sequence of solutions



By Kővari-Sós-Turán: less than $d(sk)^{2-1/d} < s^2$ edges

General roadmap for $P(1, t, t, t)$-free graphs

- Build $\mathcal{C}$: a *maximal* collection of independent cliques
- Partition the graph in classes with the same neighborhood in $\mathcal{C}$
- Show: large classes are attached to the cliques *laminarly*

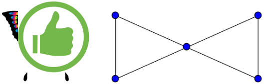
General roadmap for $P(1, t, t, t)$-free graphs

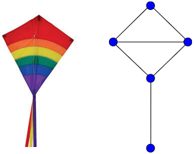
- Build $\mathcal{C}$: a *maximal* collection of independent cliques
- Partition the graph in classes with the same neighborhood in $\mathcal{C}$
- Show: large classes are attached to the cliques *laminarly*

This, the ubiquity of cliques, the $P(1, t, t, t)$-freeness imply

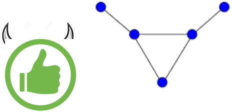
- a sparse tripartition: conclude with previous slide, or
- a dense tripartition: another lemma

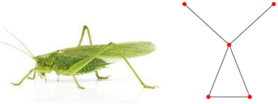
Remaining candidates on 5 vertices

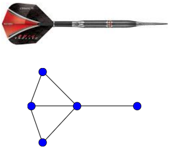




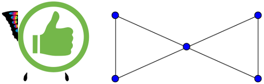
$\overline{P}$

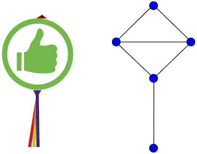


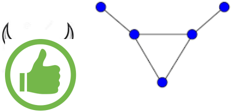


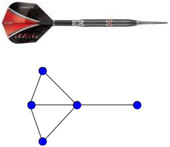


Remaining candidates on 5 vertices

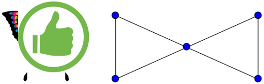


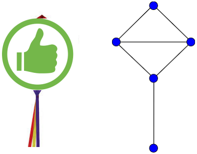


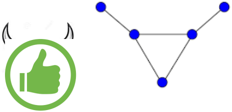


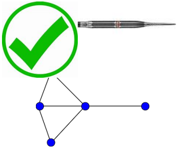


Remaining candidates on 5 vertices









Open questions

- FPT algorithm for $P(t, t, t, t)$-free graphs.
- "easy" FPT algorithm for P_5-free graphs.
- FPT algorithm for P_7-free graphs.
- derandomized algorithms for the cricket and the dart.

Open questions

- FPT algorithm for $P(t, t, t, t)$-free graphs.
- “easy” FPT algorithm for P_5-free graphs.
- FPT algorithm for P_7-free graphs.
- derandomized algorithms for the cricket and the dart.

Thank you for your attention!