Efficient Algorithms on almost special Graph Classes or How to create new problems for algorithmic research?

Venkatesh Raman

(Retd) IMSc Chennai, currently honorary professor at IIT Palakkad

Talk at Pre-conference School of CALDAM 2025,
February 10, 2025

Graph theorists study special classes of graphs like

- Trees, Forests
- Chordal Graphs
- Interval Graphs
- Circular Arc Graphs
- Perfect Graphs
- Bipartite Graphs
- Proper Interval Graphs
- Unit Interval Graphs
- Planar Graphs
- Cliques, Cluster Graphs (where each connected component is a clique)
- ....

and study (optimization) parameters like

- Maximum Clique
- Maximum Independent Set
- Minimum Dominating Set
- Chromatic Number
- Longest Path
- Longest Cycle
- Minimum Vertex Cover
- ....

- These parameters and the graph classes naturally occur in real world problems.

Motivation

- These parameters and the graph classes naturally occur in real world problems.
- Most of these graph classes are recognizable in polynomial time (most of them even linear time).

Motivation

- These parameters and the graph classes naturally occur in real world problems.
- Most of these graph classes are recognizable in polynomial time (most of them even linear time).
  Given a graph, it can be tested whether it is planar, bipartite or chordal, ... in linear time.

- These parameters and the graph classes naturally occur in real world problems.
- Most of these graph classes are recognizable in polynomial time (most of them even linear time).
  Given a graph, it can be tested whether it is planar, bipartite or chordal, ... in linear time.
- Some of these optimization parameters/problems are solvable in polynomial time in these special classes of graphs (though *NP*-hard in general graphs).

- These parameters and the graph classes naturally occur in real world problems.
- Most of these graph classes are recognizable in polynomial time (most of them even linear time).
  Given a graph, it can be tested whether it is planar, bipartite or chordal, ... in linear time.
- Some of these optimization parameters/problems are solvable in polynomial time in these special classes of graphs (though *NP*-hard in general graphs).
  For example,
  - in the class of Chordal Graphs, we can determine a maximum clique, or independent set or chromatic number in polynomial time.
  - In Bipartite graphs, we can find a minimum vertex cover in polynomial time.
  - In Trees, we can find a minimum dominating set or a vertex cover in polynomial time.

How efficiently can we solve these (recognition and optimization) problems in graph classes that are almost tree or chordal or planar or ...?

How efficiently can we solve these (recognition and optimization) problems in graph classes that are almost tree or chordal or planar or ...?

for appropriate definition of almost and appropriate definition of efficiency ....

Meaning of Almost ....

Meaning of Almost ....

- A small number of vertices/edges whose deletion makes the graph special....

Meaning of Almost ....

- A small number of vertices/edges whose deletion makes the graph special....
- In case of edge modifications, we can also consider edge contractions and deletions.

Meaning of Almost ....

- A small number of vertices/edges whose deletion makes the graph special....
- In case of edge modifications, we can also consider edge contractions and deletions.
- For the most part of this talk, we assume that a given graph has at most k vertices whose removal makes it special.

Meaning of Almost ....

- A small number of vertices/edges whose deletion makes the graph special....
- In case of edge modifications, we can also consider edge contractions and deletions.
- For the most part of this talk, we assume that a given graph has at most k vertices whose removal makes it special.
  Let us call such a graph almost special.

Meaning of Almost ....

- A small number of vertices/edges whose deletion makes the graph special....
- In case of edge modifications, we can also consider edge contractions and deletions.
- For the most part of this talk, we assume that a given graph has at most k vertices whose removal makes it special.
  Let us call such a graph almost special. Here *special* could be chordal or bipartite or forest ...

We address two algorithmic questions:

We address two algorithmic questions:

- Recognition: Is the given graph almost special?
- Optimization: In the given almost special graph, how fast can we solve .... — optimization problem like VERTEX COVER or CLIQUE or CHROMATIC NUMBER?

- Π is a non-trivial and hereditary class of graphs
  I.e. Π and $\bar{\Pi}$ are infinite and Π is closed under induced subgraphs.

- Π is a non-trivial and hereditary class of graphs
  I.e. Π and $\bar{\Pi}$ are infinite and Π is closed under induced subgraphs.
- $F(\Pi) = \{G : G \notin \Pi,\ \textit{subgraphs of } G \textit{ are in } \Pi\}$
  Minimal graphs not in Π.

- Π is a non-trivial and hereditary class of graphs
  I.e. Π and $\bar{\Pi}$ are infinite and Π is closed under induced subgraphs.
- $F(\Pi) = \{G : G \notin \Pi, \textit{ subgraphs of } G \textit{ are in } \Pi\}$
  Minimal graphs not in Π.
- For example, if Π is the class of bipartite graphs, $F(\Pi)$ is

- Π is a non-trivial and hereditary class of graphs
  I.e. Π and $\bar{\Pi}$ are infinite and Π is closed under induced subgraphs.
- $F(\Pi) = \{G : G \notin \Pi, \textit{ subgraphs of } G \textit{ are in } \Pi\}$
  Minimal graphs not in Π.
- For example, if Π is the class of bipartite graphs, $F(\Pi)$ is the set of all odd cycles.

- Π is a non-trivial and hereditary class of graphs
  I.e. Π and $\bar{\Pi}$ are infinite and Π is closed under induced subgraphs.
- $F(\Pi) = \{G : G \notin \Pi, \ \textit{subgraphs of } G \textit{ are in } \Pi\}$
  Minimal graphs not in Π.
- For example, if Π is the class of bipartite graphs, $F(\Pi)$ is the set of all odd cycles.
- Every such Π is characterized by an $F(\Pi)$.

- Π is a non-trivial and hereditary class of graphs
  I.e. Π and $\bar{\Pi}$ are infinite and Π is closed under induced subgraphs.
- $F(\Pi) = \{G : G \notin \Pi, \textit{ subgraphs of } G \textit{ are in } \Pi\}$
  Minimal graphs not in Π.
- For example, if Π is the class of bipartite graphs, $F(\Pi)$ is the set of all odd cycles.
- Every such Π is characterized by an $F(\Pi)$.
  I.e. G is not in Π if and only if some graph H in $F(\Pi)$ is an induced subgraph of G.

- Π is a non-trivial and hereditary class of graphs
  I.e. Π and $\bar{\Pi}$ are infinite and Π is closed under induced subgraphs.
- $F(\Pi) = \{G : G \notin \Pi, \textit{ subgraphs of } G \textit{ are in } \Pi\}$
  Minimal graphs not in Π.
- For example, if Π is the class of bipartite graphs, $F(\Pi)$ is the set of all odd cycles.
- Every such Π is characterized by an $F(\Pi)$.
  I.e. G is not in Π if and only if some graph H in $F(\Pi)$ is an induced subgraph of G.
- $\Pi + k = \{G(V, E) : \exists S \subseteq V, |S| \leq k, G \setminus S \in \Pi\}$

Definitions

- Π is a non-trivial and hereditary class of graphs
  I.e. Π and $\bar{\Pi}$ are infinite and Π is closed under induced subgraphs.
- $F(\Pi) = \{G : G \notin \Pi,\ \textit{subgraphs of } G \textit{ are in } \Pi\}$
  Minimal graphs not in Π.
- For example, if Π is the class of bipartite graphs, $F(\Pi)$ is the set of all odd cycles.
- Every such Π is characterized by an $F(\Pi)$.
  I.e. G is not in Π if and only if some graph H in $F(\Pi)$ is an induced subgraph of G.
- $\Pi + k = \{G(V, E) : \exists S \subseteq V, |S| \leq k, G \setminus S \in \Pi\}$
- $P : G \to N$ – a graph parameter

Some Algorithmic assumptions

- Determining whether $G(V, E)$ is in Π (Recognition)

- Determining whether $G(V, E)$ is in Π (Recognition) or finding $P(G)$ (Optimization)

- Determining whether $G(V, E)$ is in Π (Recognition) or finding $P(G)$ (Optimization) can be done in $O(|V|^c)$ time for some constant c.

- Determining whether $G(V, E)$ is in Π (Recognition) or finding $P(G)$ (Optimization) can be done in $O(|V|^c)$ time for some constant c.
  I.e. We look at Π and $P(G)$ for which this assumption is true.

- Determining whether $G(V, E)$ is in Π (Recognition) or finding $P(G)$ (Optimization) can be done in $O(|V|^c)$ time for some constant c.
  I.e. We look at Π and $P(G)$ for which this assumption is true.

This talk: Recognition and Optimization in $\Pi + k$ graphs.

Recognizing $\Pi + k$ graphs.

Meaning of Efficiency – Polynomial (in n and k)?

Meaning of Efficiency – Polynomial (in n and k)?

- An Easy Observation
  - Determining whether a given graph G is in $\Pi + k$ can be performed in $O(|V|^{k+c})$ for some constant c.

Meaning of Efficiency – Polynomial (in n and k)?

- An Easy Observation
  - Determining whether a given graph G is in $\Pi + k$ can be performed in $O(|V|^{k+c})$ for some constant c.
    Recall our assumption that determining whether G is in Π can be performed in $O(|V|^c)$ time.

Meaning of Efficiency – Polynomial (in n and k)?

- An Easy Observation
  - Determining whether a given graph G is in $\Pi + k$ can be performed in $O(|V|^{k+c})$ for some constant c.
    Recall our assumption that determining whether G is in Π can be performed in $O(|V|^c)$ time.
- Covers problems like *vertex cover (k away from edgelss)*, *feedback vertex set (k away from a forest)*, *odd cycle transversal (k away from a bipartite graph)*, *split vertex deletion (k away from a split graph)*, *cluster vertex deletion set (k away from a cluster graph)*

Meaning of Efficiency – Polynomial (in n and k)?

- An Easy Observation
  - Determining whether a given graph G is in $\Pi + k$ can be performed in $O(|V|^{k+c})$ for some constant c.
    Recall our assumption that determining whether G is in Π can be performed in $O(|V|^{c})$ time.
- Covers problems like *vertex cover (k away from edgelss)*, *feedback vertex set (k away from a forest)*, *odd cycle transversal (k away from a bipartite graph)*, *split vertex deletion (k away from a split graph)*, *cluster vertex deletion set (k away from a cluster graph)*
- A classical Result (Lewis and Yannakakis [1980])
  Determining whether a given graph G is in $\Pi + k$ is NP-complete.

Meaning of Efficiency – Polynomial (in n and k)?

- An Easy Observation
  - Determining whether a given graph G is in $\Pi + k$ can be performed in $O(|V|^{k+c})$ for some constant c.
    Recall our assumption that determining whether G is in Π can be performed in $O(|V|^c)$ time.
- Covers problems like *vertex cover (k away from edgelss), feedback vertex set (k away from a forest), odd cycle transversal (k away from a bipartite graph), split vertex deletion (k away from a split graph), cluster vertex deletion set (k away from a cluster graph)*
- A classical Result (Lewis and Yannakakis [1980])
  Determining whether a given graph G is in $\Pi + k$ is NP-complete.
  I.e. determining whether a given graph has at most k vertices whose deletion results in Π is NP-complete

Meaning of Efficiency – Polynomial (in n and k)?

- An Easy Observation
  - Determining whether a given graph G is in $\Pi + k$ can be performed in $O(|V|^{k+c})$ for some constant c.
    Recall our assumption that determining whether G is in Π can be performed in $O(|V|^c)$ time.
- Covers problems like *vertex cover (k away from edgelss), feedback vertex set (k away from a forest), odd cycle transversal (k away from a bipartite graph), split vertex deletion (k away from a split graph), cluster vertex deletion set (k away from a cluster graph)*
- A classical Result (Lewis and Yannakakis [1980])
  Determining whether a given graph G is in $\Pi + k$ is NP-complete.
  I.e. determining whether a given graph has at most k vertices whose deletion results in Π is NP-complete
- So an algorithm with time Polynomial($|V(G)|, k$) is unlikely.

So what should be the measure of ffficiency? – Fixed-parameter tractability (FPT)

So what should be the measure of ffficiency? – Fixed-parameter tractability (FPT)

- **Question:** Is there are an algorithm for recognizing graphs in $\Pi + k$, with complexity $f(k)|V|^c$? I.e. is the problem fixed-parameter tractable (FPT)?

So what should be the measure of ffficiency? – Fixed-parameter tractability (FPT)

- **Question:** Is there are an algorithm for recognizing graphs in $\Pi + k$, with complexity $f(k)|V|^c$? I.e. is the problem fixed-parameter tractable (FPT)?
  $|V|^{k+c}$ algorithm does NOT make it fixed-parameter tractable, but something like $2^k|V|^c$ does make it FPT.

So what should be the measure of ffficiency? – Fixed-parameter tractability (FPT)

- **Question:** Is there are an algorithm for recognizing graphs in $\Pi + k$, with complexity $f(k)|V|^c$? I.e. is the problem fixed-parameter tractable (FPT)?
  $|V|^{k+c}$ algorithm does NOT make it fixed-parameter tractable, but something like $2^k|V|^c$ does make it FPT.
- A popular notion to deal with NP-hard problems.

So what should be the measure of ffficiency? – Fixed-parameter tractability (FPT)

- **Question:** Is there are an algorithm for recognizing graphs in $\Pi + k$, with complexity $f(k)|V|^c$? I.e. is the problem fixed-parameter tractable (FPT)?
  $|V|^{k+c}$ algorithm does NOT make it fixed-parameter tractable, but something like $2^k|V|^c$ does make it FPT.
- A popular notion to deal with NP-hard problems.
  There is a notion of (W-)hardness in this paradigm as well.

Recognizing $\Pi + k$ graphs in FPT time

- Recall
  - $F(\Pi) = \{G : G \notin \Pi,\ \textit{subgraphs of } G \textit{ are in } \Pi\}$ (i.e. minimal graphs not in Π)

- Recall
  - $F(\Pi) = \{G : G \notin \Pi, \textit{ subgraphs of } G \textit{ are in } \Pi\}$ (i.e. minimal graphs not in Π)
  - For example, if Π is the class of bipartite graphs, $F(\Pi)$ is

Recognizing $\Pi + k$ graphs in FPT time

- Recall
  - $F(\Pi) = \{G : G \notin \Pi,\ \textit{subgraphs of } G \textit{ are in } \Pi\}$ (i.e. minimal graphs not in Π)
  - For example, if Π is the class of bipartite graphs, $F(\Pi)$ is the set of all odd cycles.

- Recall
  - $F(\Pi) = \{G : G \notin \Pi,\ \textit{subgraphs of } G \textit{ are in } \Pi\}$ (i.e. minimal graphs not in Π)
  - For example, if Π is the class of bipartite graphs, $F(\Pi)$ is the set of all odd cycles.
- **Theorem (Cai 1991)** Recognizing graphs in $\Pi + k$ is *FPT* if $F(\Pi)$ is finite.

Recognizing $\Pi + k$ graphs in FPT time

- Recall
  - $F(\Pi) = \{G : G \notin \Pi, \textit{ subgraphs of } G \textit{ are in } \Pi\}$ (i.e. minimal graphs not in Π)
  - For example, if Π is the class of bipartite graphs, $F(\Pi)$ is the set of all odd cycles.
- **Theorem (Cai 1991)** Recognizing graphs in $\Pi + k$ is *FPT* if $F(\Pi)$ is finite.
- **Corollary** Recognizing graphs in $\Pi + k$ is *FPT* if Π is edgeless or Split or Cluster

Recognizing $\Pi + k$ graphs in FPT time

- Recall
  - $F(\Pi) = \{G : G \notin \Pi,\ \textit{subgraphs of } G \textit{ are in } \Pi\}$ (i.e. minimal graphs not in Π)
  - For example, if Π is the class of bipartite graphs, $F(\Pi)$ is the set of all odd cycles.
- **Theorem (Cai 1991)** Recognizing graphs in $\Pi + k$ is *FPT* if $F(\Pi)$ is finite.
- **Corollary** Recognizing graphs in $\Pi + k$ is *FPT* if Π is edgeless or Split or Cluster (as they have finite forbidden sets).

Recognizing $\Pi + k$ graphs when $F(\Pi)$ is infinite

- If Π is the class of forests, or bipartite graphs or Chordal graphs or Interval graphs, then FPT
  (Problem specific algorithms over a series of papers).

Recognizing $\Pi + k$ graphs when $F(\Pi)$ is infinite

- If Π is the class of forests, or bipartite graphs or Chordal graphs or Interval graphs, then *FPT*
(Problem specific algorithms over a series of papers).
- If Π is the class of Perfect graphs or Wheel-free graphs, the problem is W-hard.

Recognizing $\Pi + k$ graphs when $F(\Pi)$ is infinite

- If Π is the class of forests, or bipartite graphs or Chordal graphs or Interval graphs, then *FPT*
  (Problem specific algorithms over a series of papers).
- If Π is the class of Perfect graphs or Wheel-free graphs, the problem is W-hard.
- **Open** Dichotomy? Is there a property of Π that characterizes when it is FPT and when W-hard?

Recognizing $\Pi + k$ graphs when $F(\Pi)$ is infinite

- If Π is the class of forests, or bipartite graphs or Chordal graphs or Interval graphs, then *FPT*
  (Problem specific algorithms over a series of papers).
- If Π is the class of Perfect graphs or Wheel-free graphs, the problem is W-hard.
- **Open** Dichotomy? Is there a property of Π that characterizes when it is FPT and when W-hard?
  A 'dual question': Can you find an induced subgraph with property Π in a given graph? This has a dichotomy result (Khot-R (2000)).

Recognizing $\Pi + k$ graphs when $F(\Pi)$ is infinite

- If Π is the class of forests, or bipartite graphs or Chordal graphs or Interval graphs, then *FPT*
  (Problem specific algorithms over a series of papers).
- If Π is the class of Perfect graphs or Wheel-free graphs, the problem is W-hard.
- **Open** Dichotomy? Is there a property of Π that characterizes when it is FPT and when W-hard?
  A 'dual question': Can you find an induced subgraph with property Π in a given graph? This has a dichotomy result (Khot-R (2000)).
- If Π is *minor closed*, then it has a finite forbidden (minor) set (Robertson-Seymour) and hence recognition problem can be solved in polynomial time

Recognizing $\Pi + k$ graphs when $F(\Pi)$ is infinite

- If Π is the class of forests, or bipartite graphs or Chordal graphs or Interval graphs, then *FPT*
  (Problem specific algorithms over a series of papers).
- If Π is the class of Perfect graphs or Wheel-free graphs, the problem is W-hard.
- **Open** Dichotomy? Is there a property of Π that characterizes when it is FPT and when W-hard?
  A 'dual question': Can you find an induced subgraph with property Π in a given graph? This has a dichotomy result (Khot-R (2000)).
- If Π is *minor closed*, then it has a finite forbidden (minor) set (Robertson-Seymour) and hence recognition problem can be solved in polynomial time This can also be used for recognition in FPT time for some $\Pi + k$ graphs.

Some Recent Directions

- Recognizing $G \in \Pi + ke$, or $G \in \Pi - ke$ or $G \in \Pi$ with k edge contractions.

- Recognizing $G \in \Pi + ke$, or $G \in \Pi - ke$ or $G \in \Pi$ with k edge contractions.
- **A very recent direction** Let $\Pi = \{G |\ \textit{each connected component is in}\ \Pi_1\ \textit{or}\ \Pi_2\ \textit{or}\ \ldots \Pi_c\}$

- Recognizing $G \in \Pi + ke$, or $G \in \Pi - ke$ or $G \in \Pi$ with k edge contractions.
- **A very recent direction** Let $\Pi = \{G |\ \textit{each connected component is in } \Pi_1 \textit{ or } \Pi_2 \textit{ or } \ldots \Pi_c\}$ for some $\Pi_1, \Pi_2, \ldots \Pi_c$

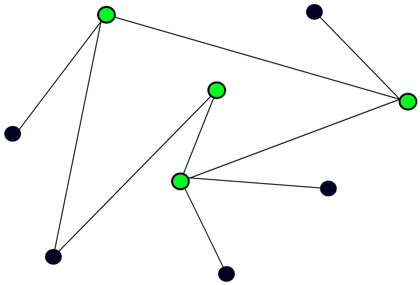
- Recognizing $G \in \Pi + ke$, or $G \in \Pi - ke$ or $G \in \Pi$ with k edge contractions.
- **A very recent direction** Let $\Pi = \{G|\ \textit{each connected component is in}\ \Pi_1\ \textit{or}\ \Pi_2\ \textit{or}\ \ldots \Pi_c\}$ for some $\Pi_1, \Pi_2, \ldots \Pi_c$ such that $F(\Pi_i)$ is finite for $i = 1$ to c.

- Recognizing $G \in \Pi + ke$, or $G \in \Pi - ke$ or $G \in \Pi$ with k edge contractions.
- **A very recent direction** Let $\Pi = \{G|$ *each connected component is in* Π_1 *or* Π_2 *or* $\ldots \Pi_c\}$ for some $\Pi_1, \Pi_2, \ldots \Pi_c$ such that $F(\Pi_i)$ is finite for $i = 1$ to c. Then recognizing G in $\Pi + k$ graphs is FPT. (Jacob et al. 2023, 2024)

- Recognizing $G \in \Pi + ke$, or $G \in \Pi - ke$ or $G \in \Pi$ with k edge contractions.
- **A very recent direction** Let
  $\Pi = \{G|$ *each connected component is in* Π_1 *or* Π_2 *or* $\ldots \Pi_c\}$
  for some $\Pi_1, \Pi_2, \ldots \Pi_c$ such that $F(\Pi_i)$ is finite for $i = 1$ to c.
  Then recognizing G in $\Pi + k$ graphs is FPT. (Jacob et al. 2023, 2024)
  Corollary Are there k vertices whose removal makes every component a clique or a split graph? (FPT).

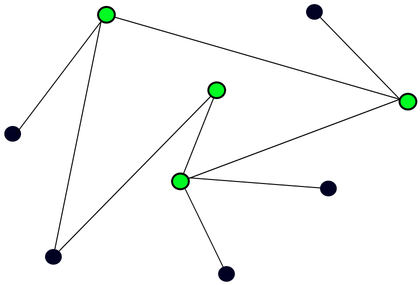
Optimization Problems in $\Pi + k$ graphs

Vertex Cover in a graph $G(V, E)$ – A subset S of vertices such that for every $(i, j) \in E$, i or j (or both) is in S.



Classical *NP*-hard problem, unlikely to have an efficient (polynomial time) algorithm in general graphs.

Vertex Cover in a graph $G(V, E)$ – A subset S of vertices such that for every $(i, j) \in E$, i or j (or both) is in S.



Classical *NP*-hard problem, unlikely to have an efficient (polynomial time) algorithm in general graphs. Has several good approximation algorithms.

Vertex Cover in Forests – Why polynomial time?

Vertex Cover in Forests – Why polynomial time?

Vertex Cover in Forests – Why polynomial time?

- Observation If x is a vertex with degree 1 with y as its unique neighbor, there exists an optimum vertex cover containing y.

Vertex Cover in Forests – Why polynomial time?

- Observation If x is a vertex with degree 1 with y as its unique neighbor, there exists an optimum vertex cover containing y.
- Degree 1 Preprocessing Rule For every pendant vertex x, include its unique neighbor in the solution and delete it. Delete all isolated vertices.

Vertex Cover in Forests – Why polynomial time?

- Observation If x is a vertex with degree 1 with y as its unique neighbor, there exists an optimum vertex cover containing y.
- Degree 1 Preprocessing Rule For every pendant vertex x, include its unique neighbor in the solution and delete it. Delete all isolated vertices.
- Question What happens if we apply the rule on a tree (connected graph with no cycles)?

- Observation Minimum vertex cover in a forest can be found in polynomial time (actually linear time).

- Observation Minimum vertex cover in a forest can be found in polynomial time (actually linear time).
- What if G is close to a forest? I.e. G is a forest plus k vertices.

- Observation Minimum vertex cover in a forest can be found in polynomial time (actually linear time).
- What if G is close to a forest? I.e. G is a forest plus k vertices.
- How efficiently can we find optimum vertex cover in such a graph?

- Observation Minimum vertex cover in a forest can be found in polynomial time (actually linear time).
- What if G is close to a forest? I.e. G is a forest plus k vertices.
- How efficiently can we find optimum vertex cover in such a graph?
- Replace forest by your favourite special graph class (Chordal, Bipartite, Planar, ...)

Vertex Cover in a Forest

- Observation Minimum vertex cover in a forest can be found in polynomial time (actually linear time).
- What if G is close to a forest? I.e. G is a forest plus k vertices.
- How efficiently can we find optimum vertex cover in such a graph?
- Replace forest by your favourite special graph class (Chordal, Bipartite, Planar, ...) and
  replace vertex cover by your favourite parameter (clique, independent set, dominating set, chromatic number)

Vertex Cover in a Forest

- Observation Minimum vertex cover in a forest can be found in polynomial time (actually linear time).
- What if G is close to a forest? I.e. G is a forest plus k vertices.
- How efficiently can we find optimum vertex cover in such a graph?
- Replace forest by your favourite special graph class (Chordal, Bipartite, Planar, ...) and
  replace vertex cover by your favourite parameter (clique, independent set, dominating set, chromatic number) which is polynomial time solvable in that class
  you have several new problems ...

Problem

- Input: A graph $G \in \Pi + k$ where Π is the class of forests.
- Question: Find the minimum vertex cover in G.

Problem

- Input: A graph $G \in \Pi + k$ where Π is the class of forests.
- Question: Find the minimum vertex cover in G.

Theorem: This problem has a $2^k n^{O(1)}$ algorithm.

Problem

- Input: A graph $G \in \Pi + k$ where Π is the class of forests.
- Question: Find the minimum vertex cover in G.

Theorem: This problem has a $2^k n^{O(1)}$ algorithm.
Proof: Let S be a subset of size k such that $G \setminus S$ is in Π. (S is a modulator).

Problem

- Input: A graph $G \in \Pi + k$ where Π is the class of forests.
- Question: Find the minimum vertex cover in G.

Theorem: This problem has a $2^k n^{O(1)}$ algorithm.
Proof: Let S be a subset of size k such that $G \setminus S$ is in Π. (S is a modulator).

- Guess the intersection Y of solution with S, delete Y.

Problem

- Input: A graph $G \in \Pi + k$ where Π is the class of forests.
- Question: Find the minimum vertex cover in G.

Theorem: This problem has a $2^k n^{O(1)}$ algorithm.
Proof: Let S be a subset of size k such that $G \setminus S$ is in Π. (S is a modulator).

- Guess the intersection Y of solution with S, delete Y. (Number of guesses is 2^k).
- Pick $N(S \setminus Y) \cap (V \setminus S)$ into the solution, delete them
- solve the (optimum) VC problem in the remaining forest.

- Note that in our algorithm, we did not really use the fact that Π contains only forests,

- Note that in our algorithm, we did not really use the fact that Π contains only forests, but that VERTEX COVER is polynomial time solvable in the class of forests.

- Note that in our algorithm, we did not really use the fact that Π contains only forests, but that VERTEX COVER is polynomial time solvable in the class of forests.
- **Theorem** If Π is a hereditary class of graphs where VERTEX COVER is solvable in polynomial time, then VERTEX COVER in $\Pi + k$ graphs has an FPT algorithm.

Dominating Set, Chromatic Number, Feedback Vertex Set in *Forest* $+ k$ graphs

Dominating Set, Chromatic Number, Feedback Vertex Set in *Forest* $+ k$ graphs

- Any graph in *Forest* $+ k$ class has treewidth at most $k + 1$, and so FVS, Dominating Set, Chromatic Number have *FPT* algorithm.
- **Open?** Are there algorithms without using treewidth machinery (like what we had for vertex cover)?

Can we extend this for chromatic number in Bipartite + k graphs?

Can we extend this for chromatic number in Bipartite + k graphs?

- **Theorem (Cai 1992)** 3-Coloring is *NP*-hard in graphs that are Bipartite + 2 vertices.

Can we extend this for chromatic number in Bipartite + k graphs?

- **Theorem (Cai 1992)** 3-Coloring is *NP*-hard in graphs that are Bipartite + 2 vertices.
- Any general result? Like

Can we extend this for chromatic number in Bipartite + k graphs?

- **Theorem (Cai 1992)** 3-Coloring is *NP*-hard in graphs that are Bipartite $+$ 2 vertices.
- Any general result? Like
  If Π and $P(G)$ satisfy some properties, then solving $P(G)$ in $\Pi + k$ is FPT or W-hard.

- Algorithms in $\Pi + ke$ graphs (edge addition, deletion or contraction)

- Algorithms in $\Pi + ke$ graphs (edge addition, deletion or contraction)
- What if S, the modulator is not given?

- Algorithms in $\Pi + ke$ graphs (edge addition, deletion or contraction)
- What if S, the modulator is not given?
  We can first find it in FPT time and then solve the problem.

- Algorithms in $\Pi + ke$ graphs (edge addition, deletion or contraction)
- What if S, the modulator is not given?
  We can first find it in FPT time and then solve the problem.
- *Solving $P(G)$ without finding the modulator* – Determine if G is not Chordal$+k$, or solve your parameter in the graph in FPT time (Jacob et al. 2022).

- Algorithms in $\Pi + ke$ graphs (edge addition, deletion or contraction)
- What if S, the modulator is not given?
  We can first find it in FPT time and then solve the problem.
- *Solving $P(G)$ without finding the modulator* – Determine if G is not Chordal$+k$, or solve your parameter in the graph in FPT time (Jacob et al. 2022).
- A lot more ...

More problems

- Take your favourite graph class that is recognizable in polynomial time.
- Pick a parameter that can be solved in that graph class in polynomial time.
- Now address the recognition and the parameter problem in graphs that are k away from the graph class in the paradigm of parameterized complexity.

Thank You