A subexponential parameterized algorithm for Proper Interval Completion

Ivan Bliznets[1] Fedor V. Fomin[2]
Marcin Pilipczuk[2] Michał Pilipczuk[2]

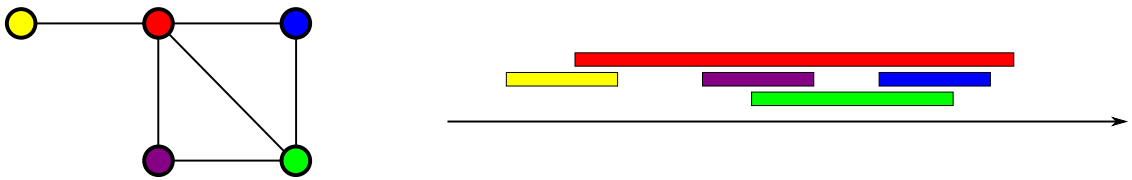
[1]St. Petersburg Academic University of the Russian Academy of Sciences, Russia

[2]Department of Informatics, University of Bergen, Norway

ESA'14, Wrocław,
September 9th, 2014

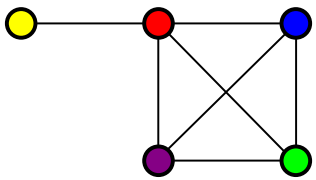
(Proper) interval graphs

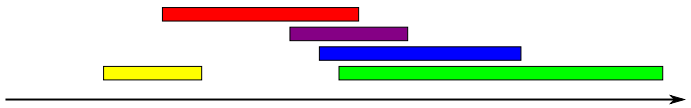
- **Interval graphs**: graphs admitting an intersection model of intervals on a line.



(Proper) interval graphs

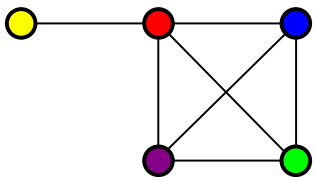
- **Interval graphs**: graphs admitting an intersection model of intervals on a line.
- **Proper** interval graphs: graphs admitting an intersection model of intervals on a line s.t. no interval contains any other interval.

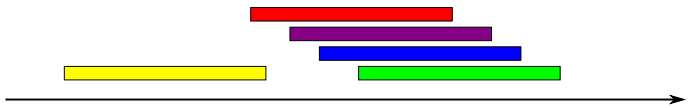




(Proper) interval graphs

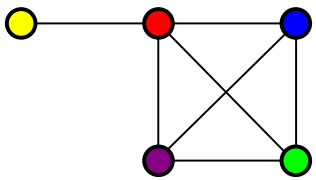
- **Interval graphs**: graphs admitting an intersection model of intervals on a line.
- **Proper** interval graphs: graphs admitting an intersection model of intervals on a line s.t. no interval contains any other interval.
- **Unit** interval graphs: graphs admitting an intersection model of unit intervals on a line.

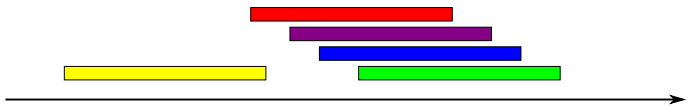




(Proper) interval graphs

- **Interval graphs**: graphs admitting an intersection model of intervals on a line.
- **Proper** interval graphs: graphs admitting an intersection model of intervals on a line s.t. no interval contains any other interval.
- **Unit** interval graphs: graphs admitting an intersection model of unit intervals on a line.
- PIG = UIG.





The problem

(Proper) Interval Completion

Input: A graph G and an integer k

Parameter: k

Question: Can one turn G into a (proper) interval graph by adding at most k edges?

The problem

(Proper) Interval Completion

Input: A graph G and an integer k

Parameter: k

Question: Can one turn G into a (proper) interval graph by adding at most k edges?

- **This talk**: FPT algorithms for PIC (time $f(k) \cdot n^{\mathcal{O}(1)}$)

The problem

(Proper) Interval Completion

Input: A graph G and an integer k

Parameter: k

Question: Can one turn G into a (proper) interval graph by adding at most k edges?

- **This talk**: FPT algorithms for PIC (time $f(k) \cdot n^{\mathcal{O}(1)}$)
- **Related paper**: the same about IC

History

- Kaplan et al., 1994: $16^k \cdot n^{\mathcal{O}(1)}$ algorithms for completion to chordal and proper interval graphs.

History

- Kaplan et al., 1994: $16^k \cdot n^{\mathcal{O}(1)}$ algorithms for completion to chordal and proper interval graphs.
  - Approach via deleting *forbidden induced subgraphs*.

History

- Kaplan et al., 1994: $16^k \cdot n^{\mathcal{O}(1)}$ algorithms for completion to chordal and proper interval graphs.
  - Approach via deleting *forbidden induced subgraphs*.
  - For PIC, the running time was later reduced to $4^k \cdot n^{\mathcal{O}(1)}$.

History

- Kaplan et al., 1994: $16^k \cdot n^{\mathcal{O}(1)}$ algorithms for completion to chordal and proper interval graphs.
  - Approach via deleting *forbidden induced subgraphs*.
  - For PIC, the running time was later reduced to $4^k \cdot n^{\mathcal{O}(1)}$.
- Villanger et al., 2007: a $k^{2k} \cdot n^{\mathcal{O}(1)}$ algorithm for IC.

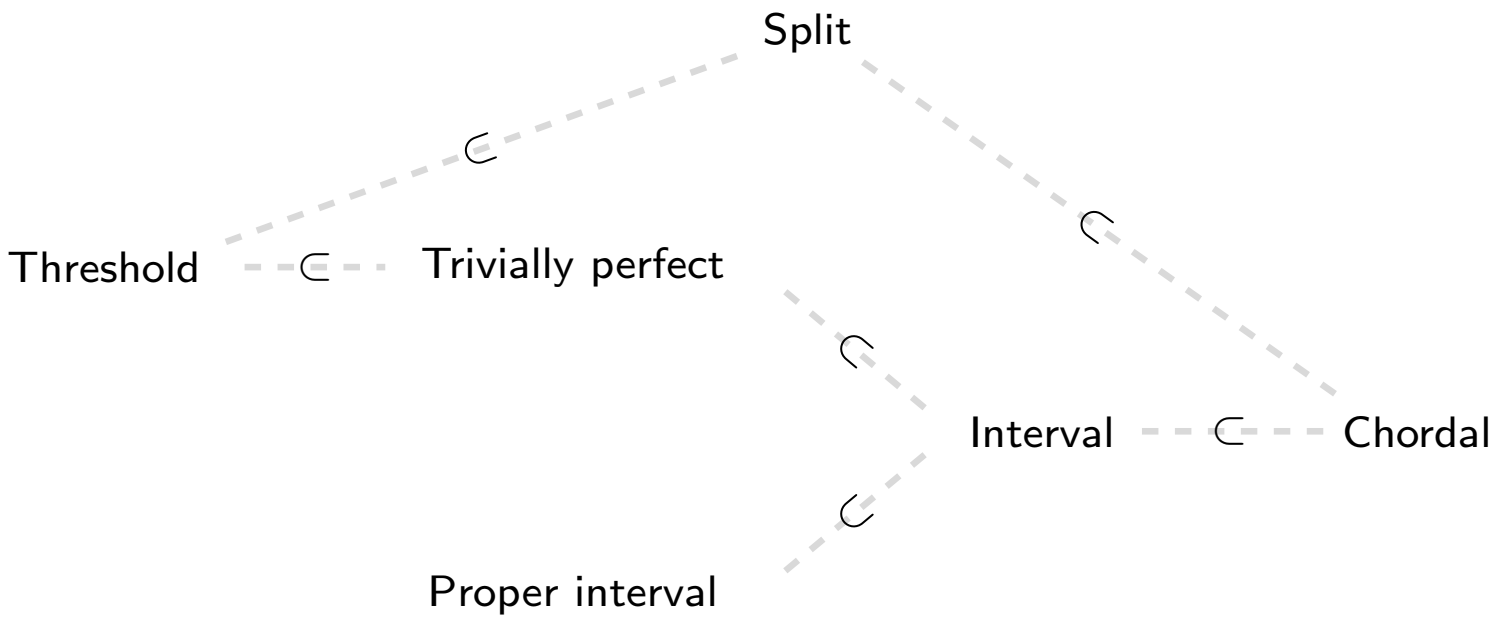
History

- Kaplan et al., 1994: $16^k \cdot n^{\mathcal{O}(1)}$ algorithms for completion to chordal and proper interval graphs.
  - Approach via deleting *forbidden induced subgraphs*.
  - For PIC, the running time was later reduced to $4^k \cdot n^{\mathcal{O}(1)}$.
- Villanger et al., 2007: a $k^{2k} \cdot n^{\mathcal{O}(1)}$ algorithm for IC.
- Fomin and Villanger, 2012: a $k^{\mathcal{O}(k^{1/2})} \cdot n^{\mathcal{O}(1)}$ algorithm for Chordal Completion.

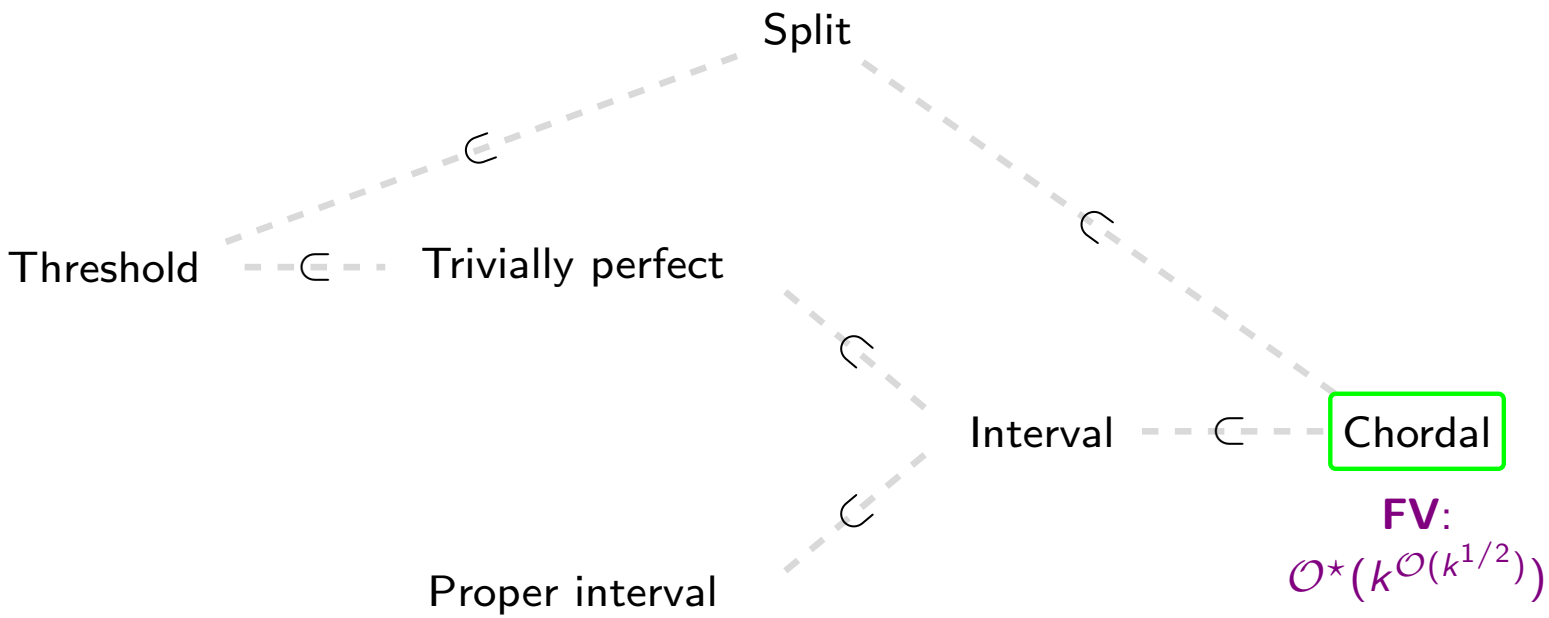
History

- Kaplan et al., 1994: $16^k \cdot n^{\mathcal{O}(1)}$ algorithms for completion to chordal and proper interval graphs.
  - Approach via deleting *forbidden induced subgraphs*.
  - For PIC, the running time was later reduced to $4^k \cdot n^{\mathcal{O}(1)}$.
- Villanger et al., 2007: a $k^{2k} \cdot n^{\mathcal{O}(1)}$ algorithm for IC.
- Fomin and Villanger, 2012: a $k^{\mathcal{O}(k^{1/2})} \cdot n^{\mathcal{O}(1)}$ algorithm for Chordal Completion.
- Do other completion problems admit subexponential parameterized algorithms?

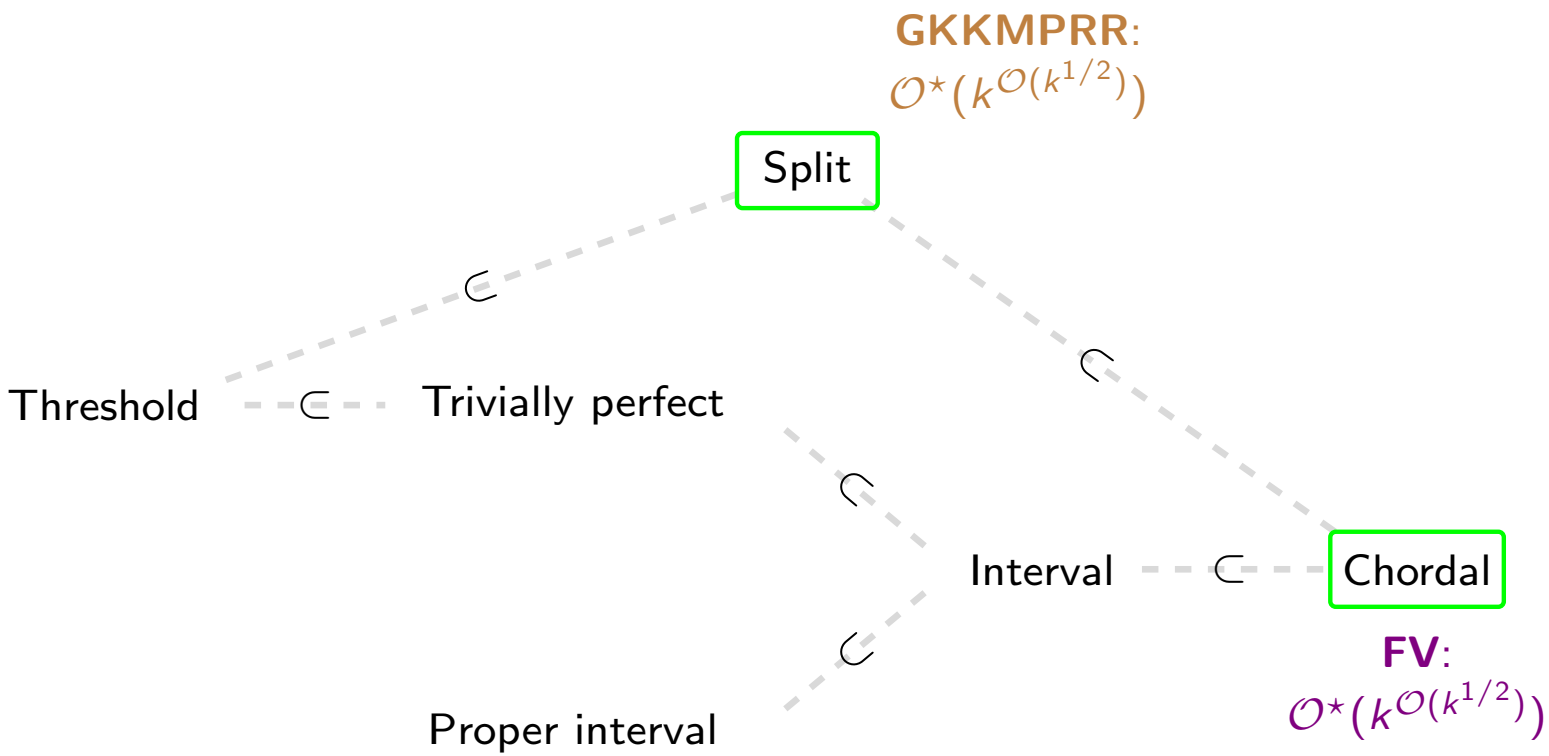
Subexponential algorithms



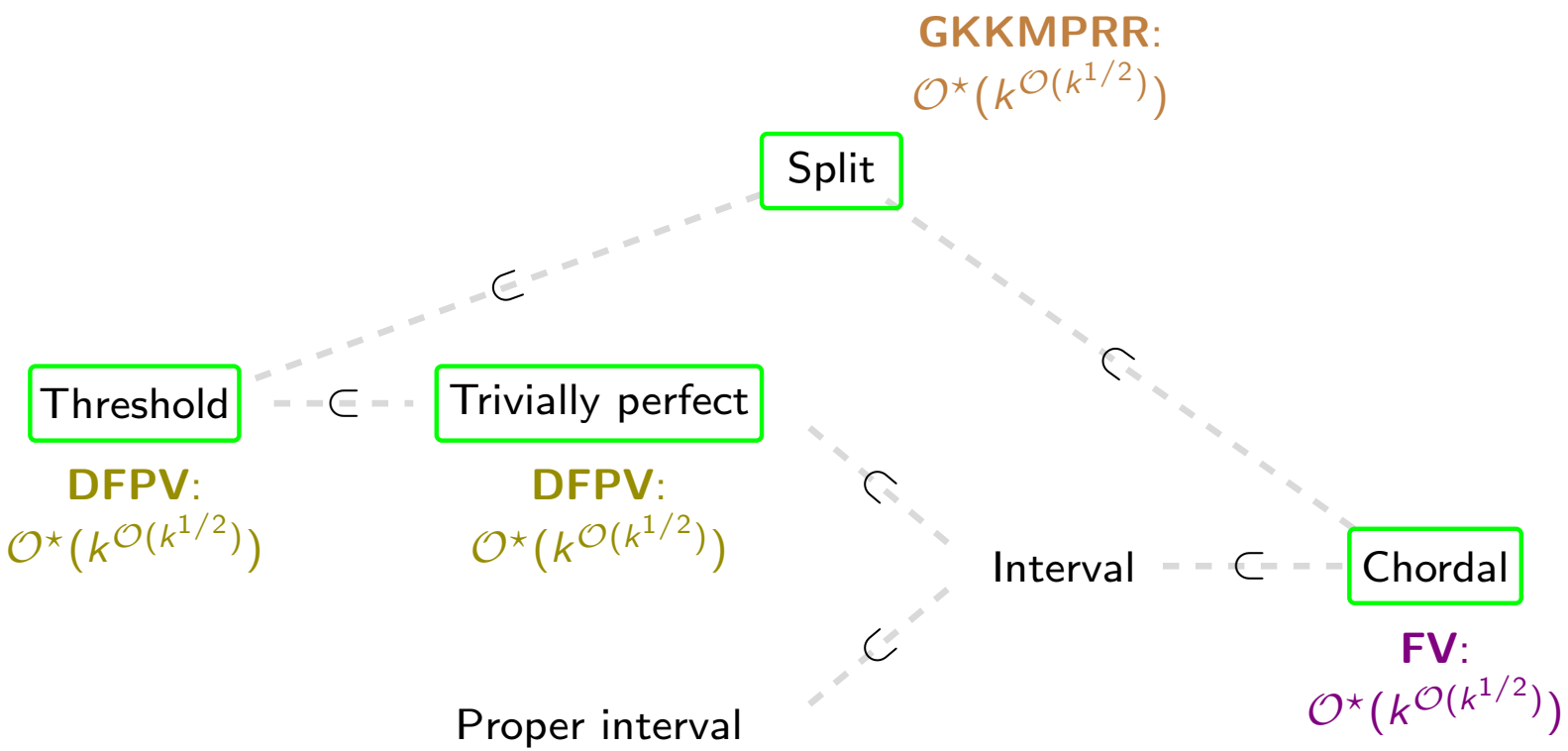
Subexponential algorithms



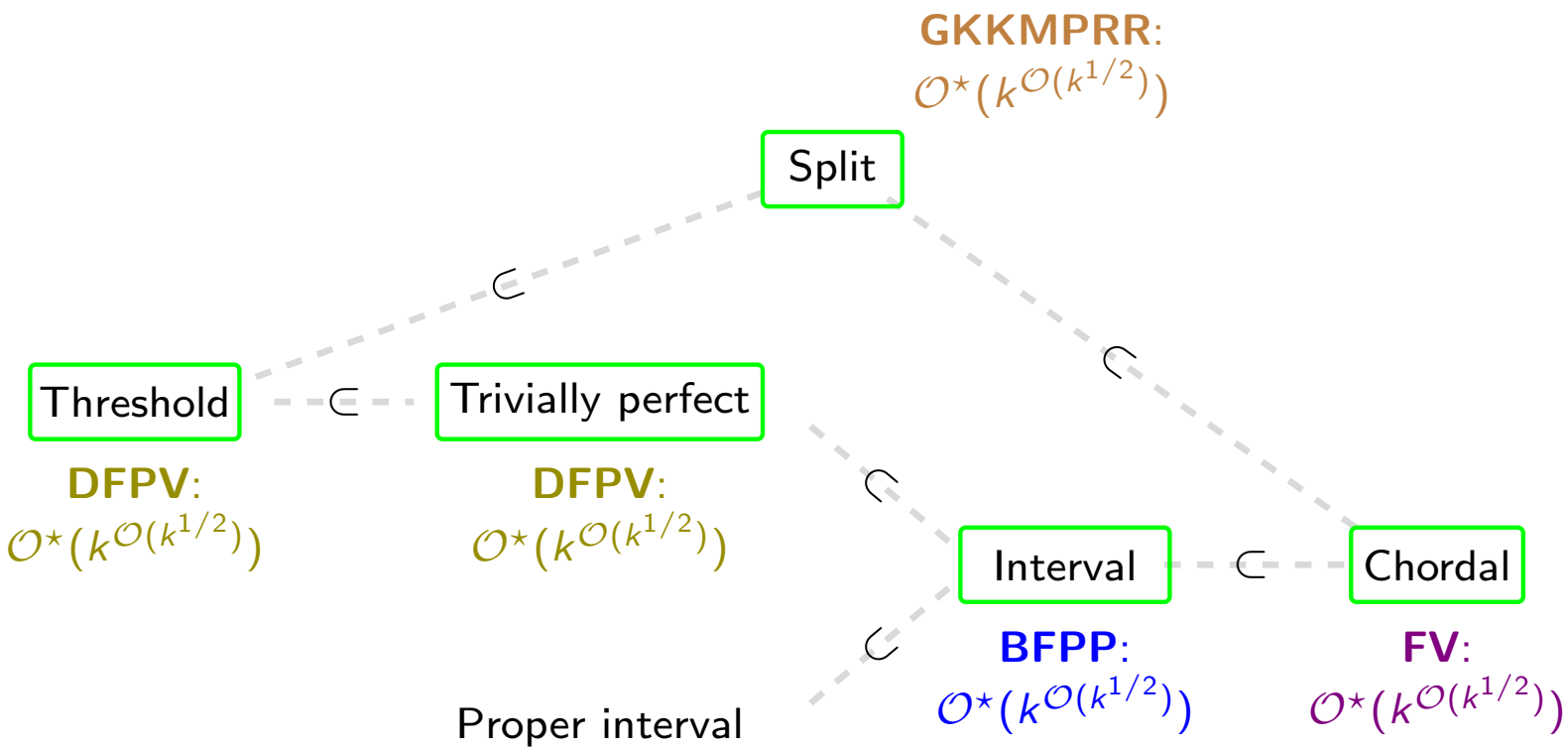
Subexponential algorithms



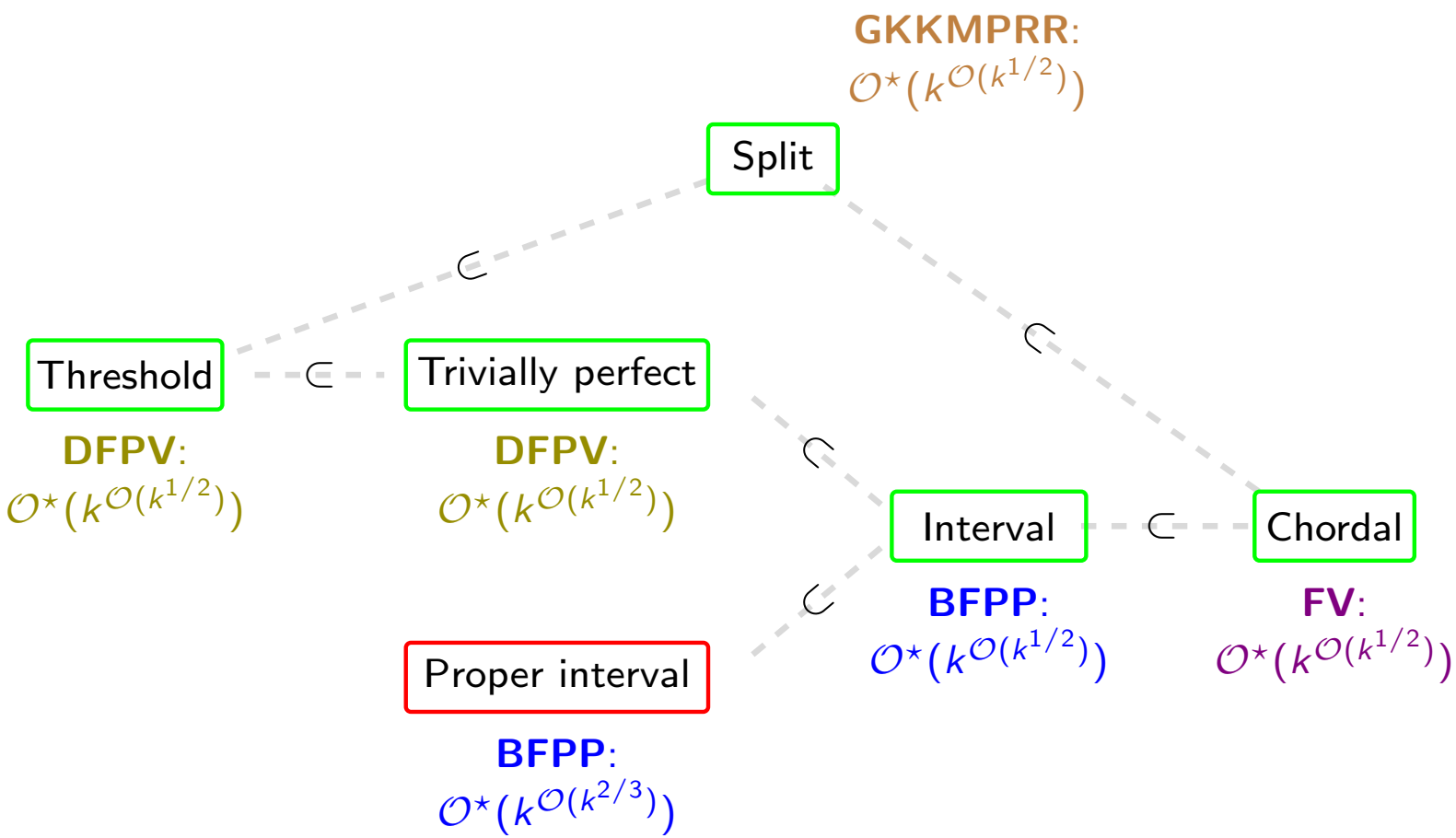
Subexponential algorithms



Subexponential algorithms



Subexponential algorithms



SUBEPT and ETH

- For many other edge modification problems, under ETH one can exclude existence of a $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ algorithm.

SUBEPT and ETH

- For many other edge modification problems, under ETH one can exclude existence of a $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ algorithm.
- Examples: C_4-free Deletion, C_4-free Completion, Trivially Perfect Deletion, Cograph Completion... (DFPV).

SUBEPT and ETH

- For many other edge modification problems, under ETH one can exclude existence of a $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ algorithm.
- Examples: C_4-free Deletion, C_4-free Completion, Trivially Perfect Deletion, Cograph Completion... (DFPV).
- Essentially, the presented completion problems are singular cases for which subexponential parameterized algorithms are possible.

The approach of FV

- **Standard view**: Kill all the forbidden subgraphs by adding as few edges as possible.

The approach of FV

- **Standard view**: Kill all the forbidden subgraphs by adding as few edges as possible.
- **Alternative view**: Find a shape of the decomposition of the completed graph that requires the least number of fill edges.

The approach of FV

- **Standard view**: Kill all the forbidden subgraphs by adding as few edges as possible.
- **Alternative view**: Find a shape of the decomposition of the completed graph that requires the least number of fill edges.
- Each of the considered graph classes has a decomposition: a clique tree, an interval model, etc.

The approach of FV

- **Standard view**: Kill all the forbidden subgraphs by adding as few edges as possible.
- **Alternative view**: Find a shape of the decomposition of the completed graph that requires the least number of fill edges.
- Each of the considered graph classes has a decomposition: a clique tree, an interval model, etc.
- Decomposition has building blocks, e.g., maximal cliques.

The approach of FV

- **Standard view**: Kill all the forbidden subgraphs by adding as few edges as possible.
- **Alternative view**: Find a shape of the decomposition of the completed graph that requires the least number of fill edges.
- Each of the considered graph classes has a decomposition: a clique tree, an interval model, etc.
- Decomposition has building blocks, e.g., maximal cliques.
- A chordal graph has $\leq n + 1$ maximal cliques.

The approach of FV

- **Standard view**: Kill all the forbidden subgraphs by adding as few edges as possible.
- **Alternative view**: Find a shape of the decomposition of the completed graph that requires the least number of fill edges.
- Each of the considered graph classes has a decomposition: a clique tree, an interval model, etc.
- Decomposition has building blocks, e.g., maximal cliques.
- A chordal graph has $\leq n + 1$ maximal cliques.
- **Idea**: A graph that lacks k edges to being a chordal graph, has $\leq k^{\mathcal{O}(k^{1/2})} \cdot n^{\mathcal{O}(1)}$ sets that can become a clique after completion (*potential maximal cliques*).

The approach of FV

- **Algorithm**:

The approach of FV

- **Algorithm**:
  - Enumerate potential building blocks. Hopefully there is $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ of them.

The approach of FV

- **Algorithm**:
  - Enumerate potential building blocks. Hopefully there is $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ of them.
  - Perform a dynamic programming algorithm that assembles the decomposition.

The approach of FV

- **Algorithm**:
  - Enumerate potential building blocks. Hopefully there is $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ of them.
  - Perform a dynamic programming algorithm that assembles the decomposition.
  - In the DP, keep track of the minimum possible number of fill edges.

The approach of FV

- **Algorithm**:
  - Enumerate potential building blocks. Hopefully there is $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ of them.
  - Perform a dynamic programming algorithm that assembles the decomposition.
  - In the DP, keep track of the minimum possible number of fill edges.
- **Ingredients**:

The approach of FV

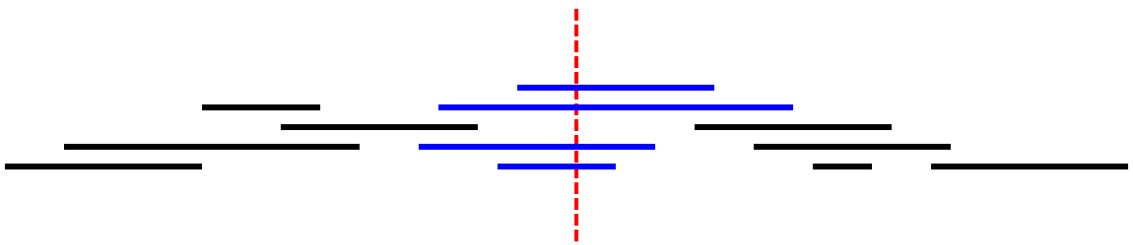
- **Algorithm**:
  - Enumerate potential building blocks. Hopefully there is $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ of them.
  - Perform a dynamic programming algorithm that assembles the decomposition.
  - In the DP, keep track of the minimum possible number of fill edges.
- **Ingredients**:
  - Final DP that assembles the decomposition.

The approach of FV

- **Algorithm**:
  - Enumerate potential building blocks. Hopefully there is $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ of them.
  - Perform a dynamic programming algorithm that assembles the decomposition.
  - In the DP, keep track of the minimum possible number of fill edges.
- **Ingredients**:
  - Final DP that assembles the decomposition.
  - Enumeration of building blocks.

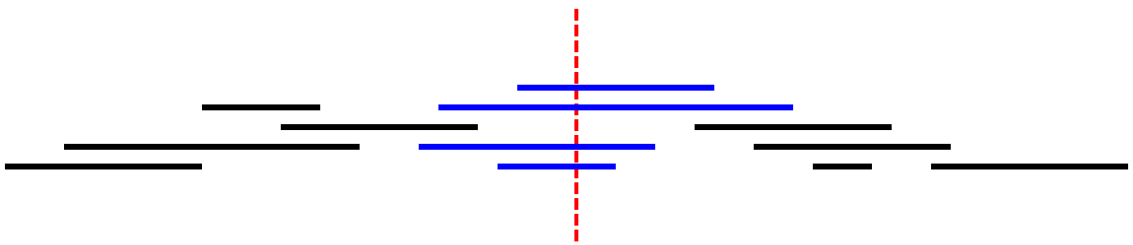
First try for Interval Completion

- **Section**: a set of intervals pinpointed by a vertical line.



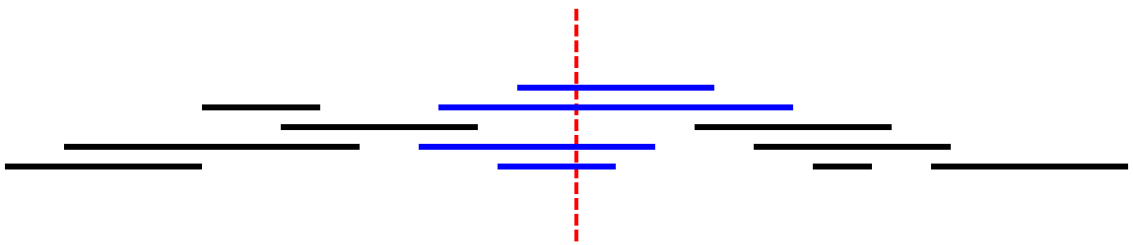
First try for Interval Completion

- **Section**: a set of intervals pinpointed by a vertical line.
- **DP state**: A pair (L, Ω), where



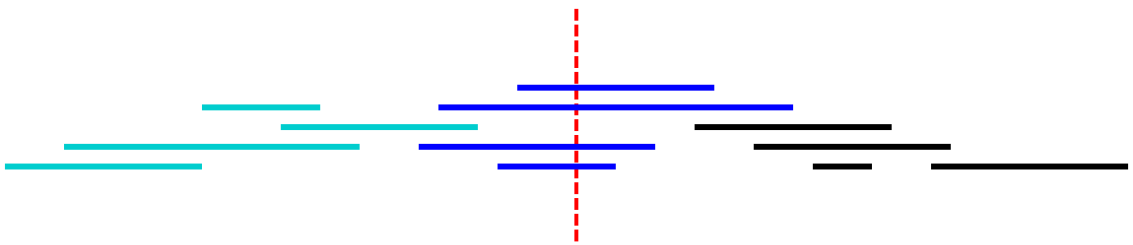
First try for Interval Completion

- **Section**: a set of intervals pinpointed by a vertical line.
- **DP state**: A pair (L, Ω), where
  - Ω is a potential section;



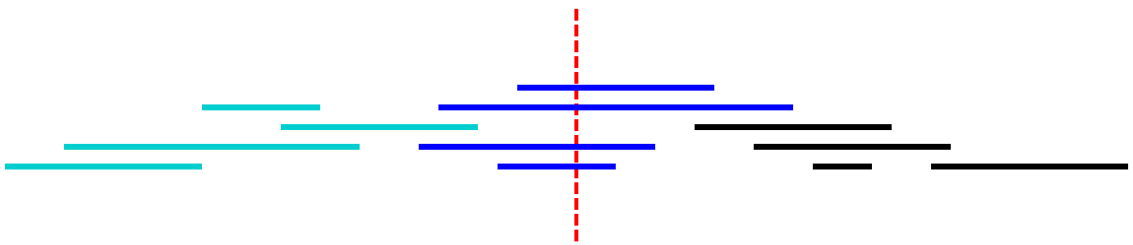
First try for Interval Completion

- **Section**: a set of intervals pinpointed by a vertical line.
- **DP state**: A pair (L, Ω), where
  - Ω is a potential section;
  - L is a subset of cc of $G - \Omega$ that will go to the left.



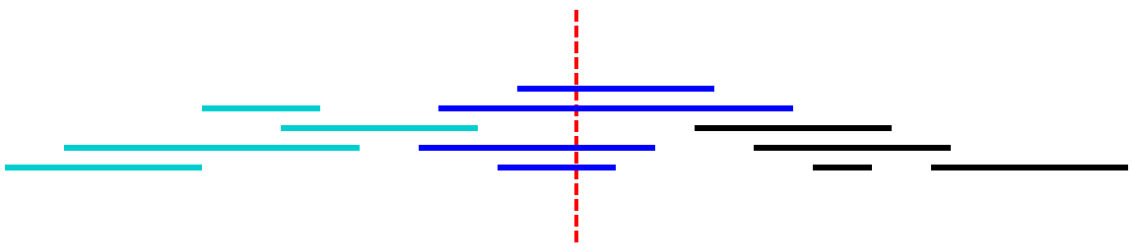
First try for Interval Completion

- **Section**: a set of intervals pinpointed by a vertical line.
- **DP state**: A pair (L, Ω), where
  - Ω is a potential section;
  - L is a subset of cc of $G - \Omega$ that will go to the left.
- $\mathrm{DP}[L, \Omega]$ is the minimum number of fill edges needed to make $G[L \cup \Omega]$ interval with Ω being the end-clique.



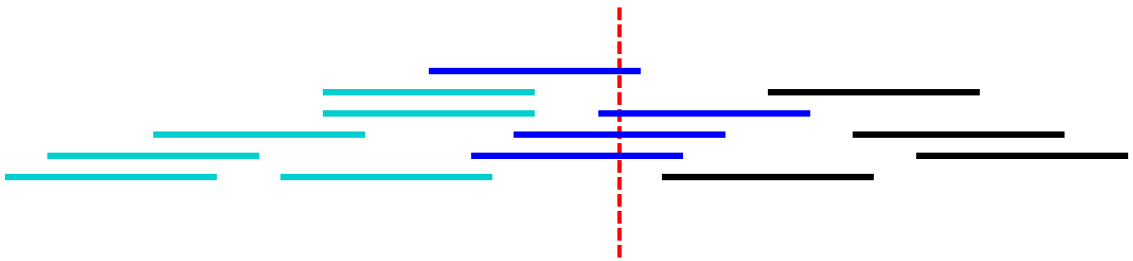
First try for Interval Completion

- **Section**: a set of intervals pinpointed by a vertical line.
- **DP state**: A pair (L, Ω), where
  - Ω is a potential section;
  - L is a subset of cc of $G - \Omega$ that will go to the left.
- $\mathrm{DP}[L, \Omega]$ is the minimum number of fill edges needed to make $G[L \cup \Omega]$ interval with Ω being the end-clique.
- If we had a family $\mathcal{N}$ capturing all the relevant states, then the DP computes the optimum solution in time $\mathrm{poly}(|\mathcal{N}|)$.



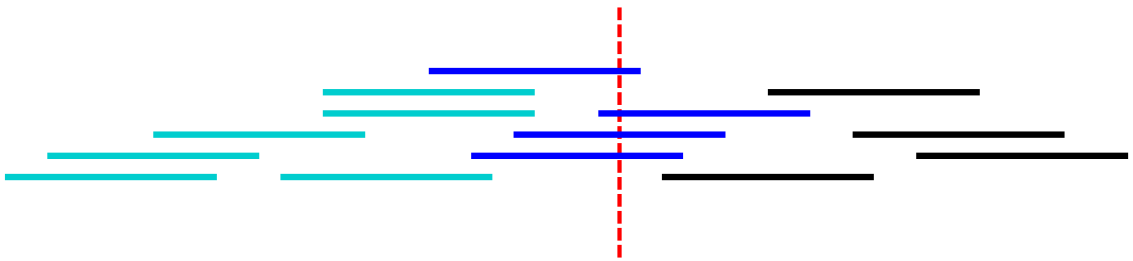
First try for Proper Interval Completion

- Let's try the same approach for the states.



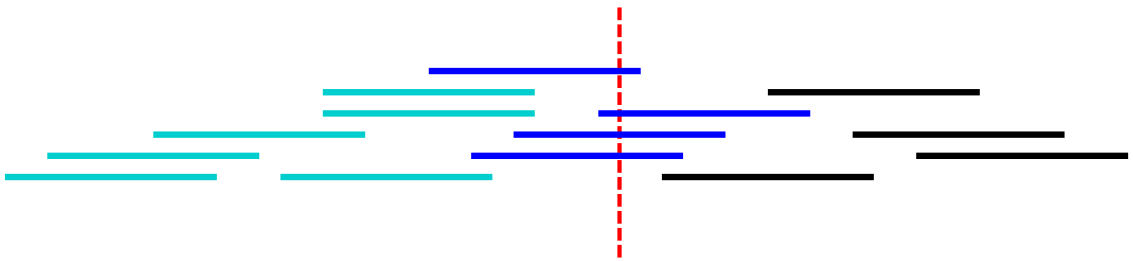
First try for Proper Interval Completion

- Let's try the same approach for the states.
- **Problem**: The section itself is not enough!



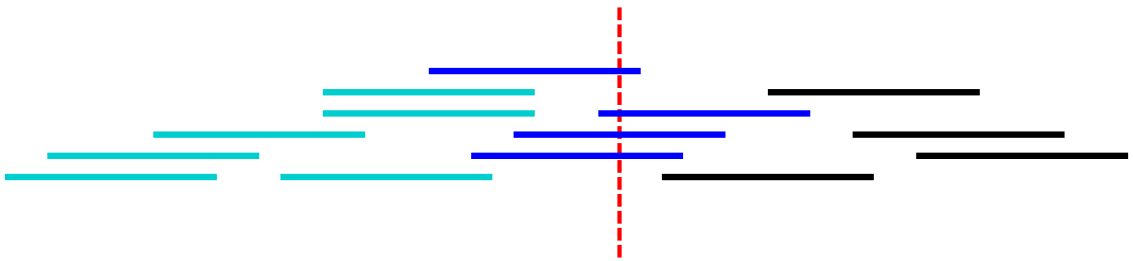
First try for Proper Interval Completion

- Let's try the same approach for the states.
- **Problem**: The section itself is not enough!
  - During construction we need to make sure intervals are closed in the same order they were opened.



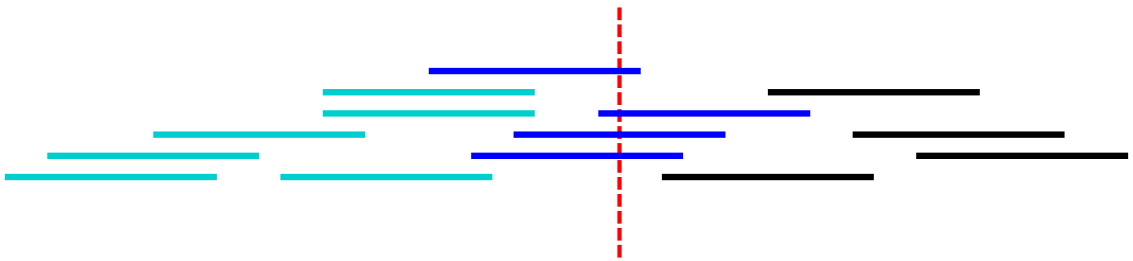
First try for Proper Interval Completion

- Let's try the same approach for the states.
- **Problem**: The section itself is not enough!
  - During construction we need to make sure intervals are closed in the same order they were opened.
- Proposition for a state: $(L, \Omega, \sigma_\Omega)$, where σ_Ω is an ordering of Ω.



First try for Proper Interval Completion

- Let's try the same approach for the states.
- **Problem**: The section itself is not enough!
  - During construction we need to make sure intervals are closed in the same order they were opened.
- Proposition for a state: $(L, \Omega, \sigma_\Omega)$, where σ_Ω is an ordering of Ω.
- Having all possible orderings is too expensive.



And how this really works...

- Interval Completion:

And how this really works...

- Interval Completion:
  - Finding $n^{\mathcal{O}(\sqrt{k})}$ candidates for sections is not that difficult.

And how this really works...

- Interval Completion:
  - Finding $n^{\mathcal{O}(\sqrt{k})}$ candidates for sections is not that difficult.
  - Finding $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ candidates for sections is more difficult.

And how this really works...

- Interval Completion:
  - Finding $n^{\mathcal{O}(\sqrt{k})}$ candidates for sections is not that difficult.
  - Finding $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ candidates for sections is more difficult.
  - We cannot have $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ partitions into left and right!

And how this really works...

- Interval Completion:
  - Finding $n^{\mathcal{O}(\sqrt{k})}$ candidates for sections is not that difficult.
  - Finding $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ candidates for sections is more difficult.
  - We cannot have $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ partitions into left and right!
  - We need to remodel the whole DP to construct this partition along the way.

And how this really works...

- Interval Completion:
  - Finding $n^{\mathcal{O}(\sqrt{k})}$ candidates for sections is not that difficult.
  - Finding $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ candidates for sections is more difficult.
  - We cannot have $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ partitions into left and right!
  - We need to remodel the whole DP to construct this partition along the way.
- Proper Interval Completion:

And how this really works...

- Interval Completion:
  - Finding $n^{\mathcal{O}(\sqrt{k})}$ candidates for sections is not that difficult.
  - Finding $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ candidates for sections is more difficult.
  - We cannot have $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ partitions into left and right!
  - We need to remodel the whole DP to construct this partition along the way.
- Proper Interval Completion:
  - Bessy, Perez: $\mathcal{O}(k^3)$ kernel for the problem, so $n = \mathcal{O}(k^3)$.

And how this really works...

- Interval Completion:
  - Finding $n^{\mathcal{O}(\sqrt{k})}$ candidates for sections is not that difficult.
  - Finding $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ candidates for sections is more difficult.
  - We cannot have $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ partitions into left and right!
  - We need to remodel the whole DP to construct this partition along the way.
- Proper Interval Completion:
  - Bessy, Perez: $\mathcal{O}(k^3)$ kernel for the problem, so $n = \mathcal{O}(k^3)$.
  - Finding $2^{o(k)}$ candidates for sections is not that difficult.

And how this really works...

- Interval Completion:
  - Finding $n^{\mathcal{O}(\sqrt{k})}$ candidates for sections is not that difficult.
  - Finding $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ candidates for sections is more difficult.
  - We cannot have $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ partitions into left and right!
  - We need to remodel the whole DP to construct this partition along the way.
- Proper Interval Completion:
  - Bessy, Perez: $\mathcal{O}(k^3)$ kernel for the problem, so $n = \mathcal{O}(k^3)$.
  - Finding $2^{o(k)}$ candidates for sections is not that difficult.
  - Getting partition into left/right from a candidate is very easy.

And how this really works...

- Interval Completion:
  - Finding $n^{\mathcal{O}(\sqrt{k})}$ candidates for sections is not that difficult.
  - Finding $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ candidates for sections is more difficult.
  - We cannot have $k^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ partitions into left and right!
  - We need to remodel the whole DP to construct this partition along the way.
- Proper Interval Completion:
  - Bessy, Perez: $\mathcal{O}(k^3)$ kernel for the problem, so $n = \mathcal{O}(k^3)$.
  - Finding $2^{o(k)}$ candidates for sections is not that difficult.
  - Getting partition into left/right from a candidate is very easy.
  - We are not able to enumerate $2^{o(k)}$ candidates for the ordering.

Expensive vertices and sections

- **Expensive vertices**: vertices that have more than $\tau = k^{1/3}$ incident fill edges.

Expensive vertices and sections

- **Expensive vertices**: vertices that have more than $\tau = k^{1/3}$ incident fill edges.
  - There is at most $2k^{2/3}$ of them.

Expensive vertices and sections

- **Expensive vertices**: vertices that have more than $\tau = k^{1/3}$ incident fill edges.
  - There is at most $2k^{2/3}$ of them.
  - Guess all of them and their positions.

Expensive vertices and sections

- **Expensive vertices**: vertices that have more than $\tau = k^{1/3}$ incident fill edges.
  - There is at most $2k^{2/3}$ of them.
  - Guess all of them and their positions.
  - Move to a sandwich problem.

Expensive vertices and sections

- **Expensive vertices**: vertices that have more than $\tau = k^{1/3}$ incident fill edges.
  - There is at most $2k^{2/3}$ of them.
  - Guess all of them and their positions.
  - Move to a sandwich problem.
- Provided that expensive vertices are guessed, there is $k^{\mathcal{O}(\tau)} = k^{\mathcal{O}(k^{1/3})}$ candidates for sections and left/right.

Dealing with the ordering of a section

- The ordering would be possible to reconstruct if we knew all the fill edges incident to the section.

Dealing with the ordering of a section

- The ordering would be possible to reconstruct if we knew all the fill edges incident to the section.
- If there were $k^{2/3}$ of them, then there would be $n^{2k^{2/3}}$ guesses for such edges.

Dealing with the ordering of a section

- The ordering would be possible to reconstruct if we knew all the fill edges incident to the section.
- If there were $k^{2/3}$ of them, then there would be $n^{2k^{2/3}}$ guesses for such edges.
- Ergo, we have $k^{\mathcal{O}(k^{2/3})}$ candidates for sections together with their orderings, provided they are *cheap* — incident to at most $k^{2/3}$ fill edges.

Dealing with the ordering of a section

- The ordering would be possible to reconstruct if we knew all the fill edges incident to the section.
- If there were $k^{2/3}$ of them, then there would be $n^{2k^{2/3}}$ guesses for such edges.
- Ergo, we have $k^{\mathcal{O}(k^{2/3})}$ candidates for sections together with their orderings, provided they are *cheap* — incident to at most $k^{2/3}$ fill edges.
- **Layer-one DP**: go from a cheap section to a cheap section.

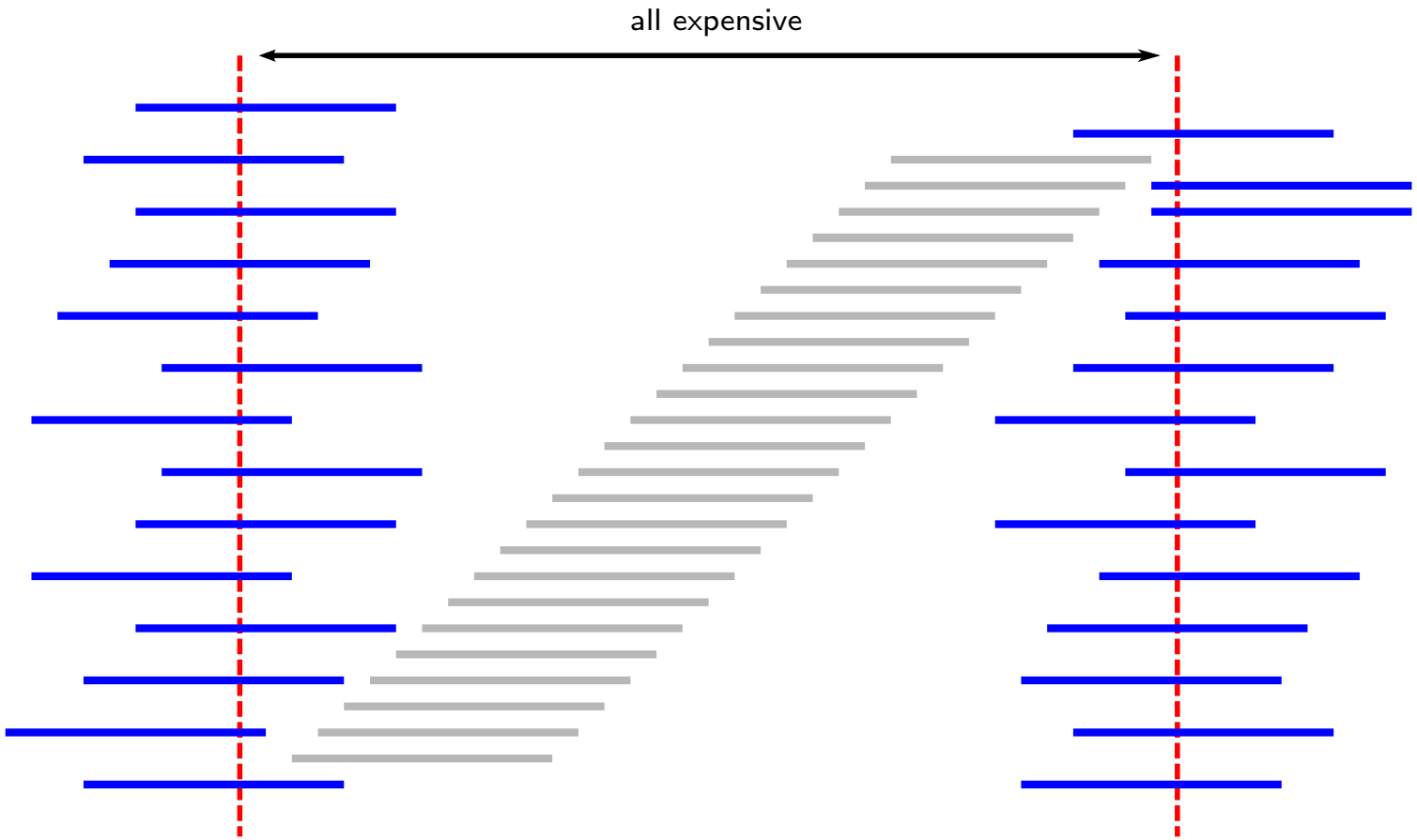
Dealing with the ordering of a section

- The ordering would be possible to reconstruct if we knew all the fill edges incident to the section.
- If there were $k^{2/3}$ of them, then there would be $n^{2k^{2/3}}$ guesses for such edges.
- Ergo, we have $k^{\mathcal{O}(k^{2/3})}$ candidates for sections together with their orderings, provided they are *cheap* — incident to at most $k^{2/3}$ fill edges.
- **Layer-one DP**: go from a cheap section to a cheap section.
- We need to compute the best possible completion between two cheap sections, assuming that all the sections in between are expensive.

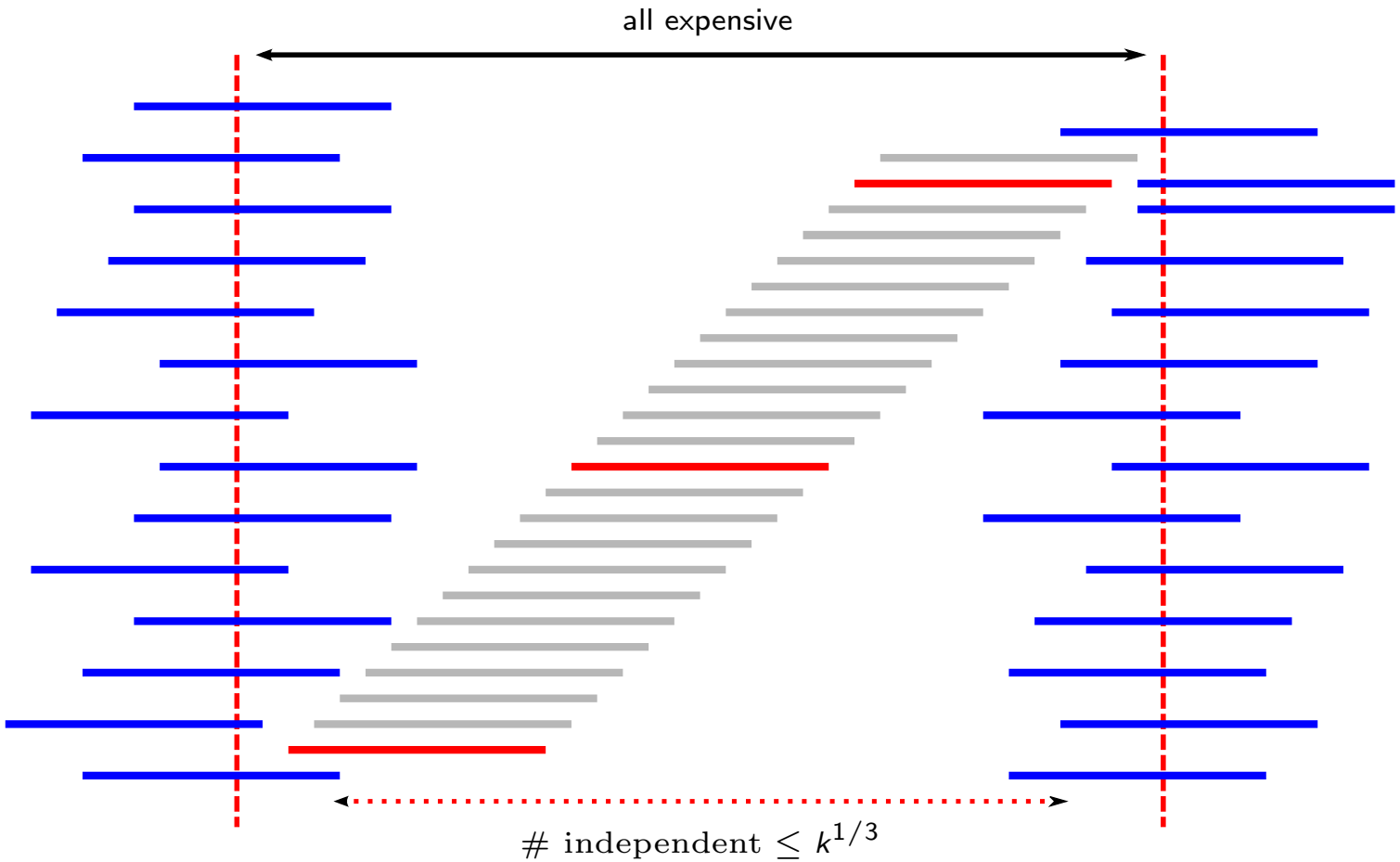
Dealing with the ordering of a section

- The ordering would be possible to reconstruct if we knew all the fill edges incident to the section.
- If there were $k^{2/3}$ of them, then there would be $n^{2k^{2/3}}$ guesses for such edges.
- Ergo, we have $k^{\mathcal{O}(k^{2/3})}$ candidates for sections together with their orderings, provided they are *cheap* — incident to at most $k^{2/3}$ fill edges.
- **Layer-one DP**: go from a cheap section to a cheap section.
- We need to compute the best possible completion between two cheap sections, assuming that all the sections in between are expensive.
- For this **Layer-two DP**.

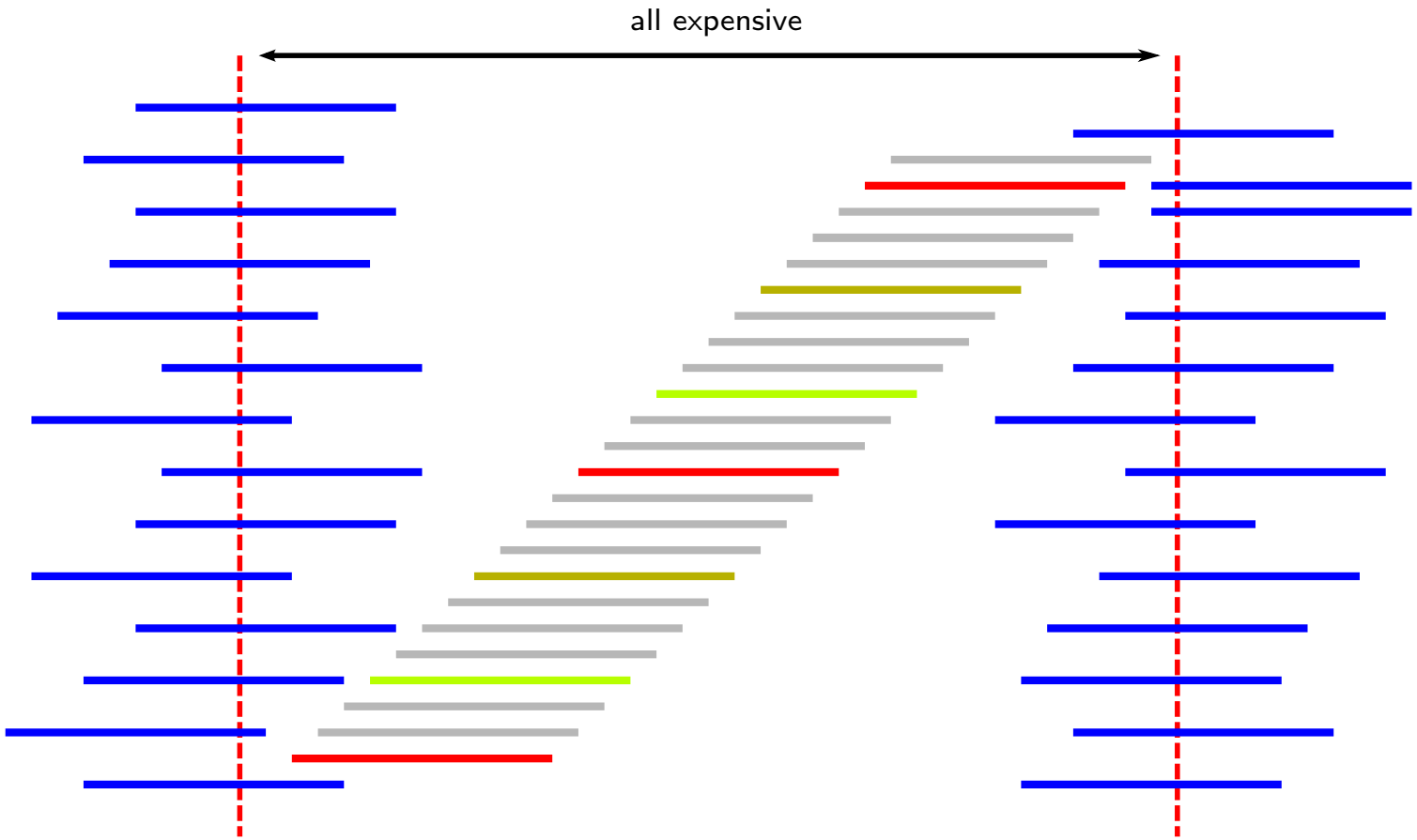
Between consecutive cheap sections



Between consecutive cheap sections

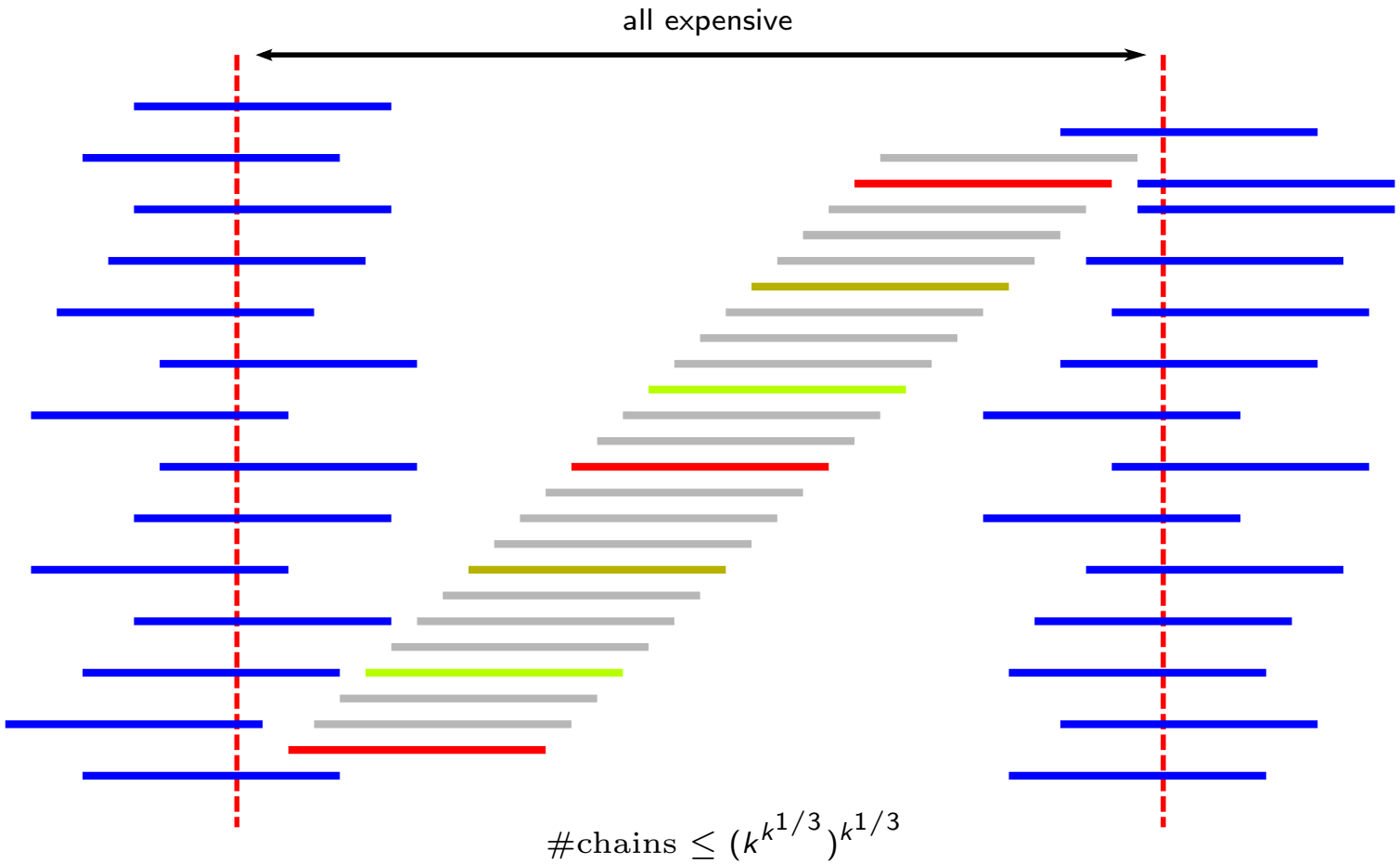


Between consecutive cheap sections



chains are DP states

Between consecutive cheap sections



$$\#\text{chains} \leq (k^{k^{1/3}})^{k^{1/3}}$$

And what happens in the paper really

- Almost all the technical details hidden in this sketch.

And what happens in the paper really

- Almost all the technical details hidden in this sketch.
- Instead of interval model, we work on an *umbrella ordering*.

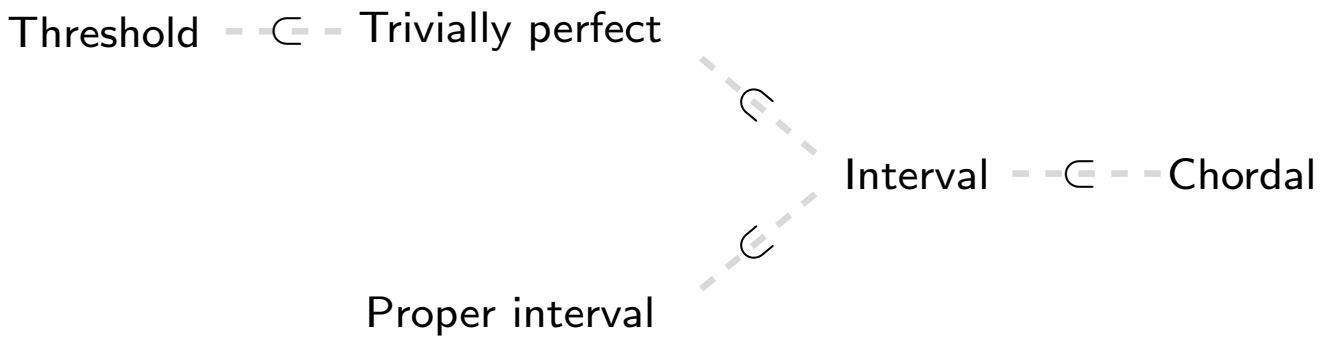
And what happens in the paper really

- Almost all the technical details hidden in this sketch.
- Instead of interval model, we work on an *umbrella ordering*.
- Perfect twins: need to canonize the model to break the ties.

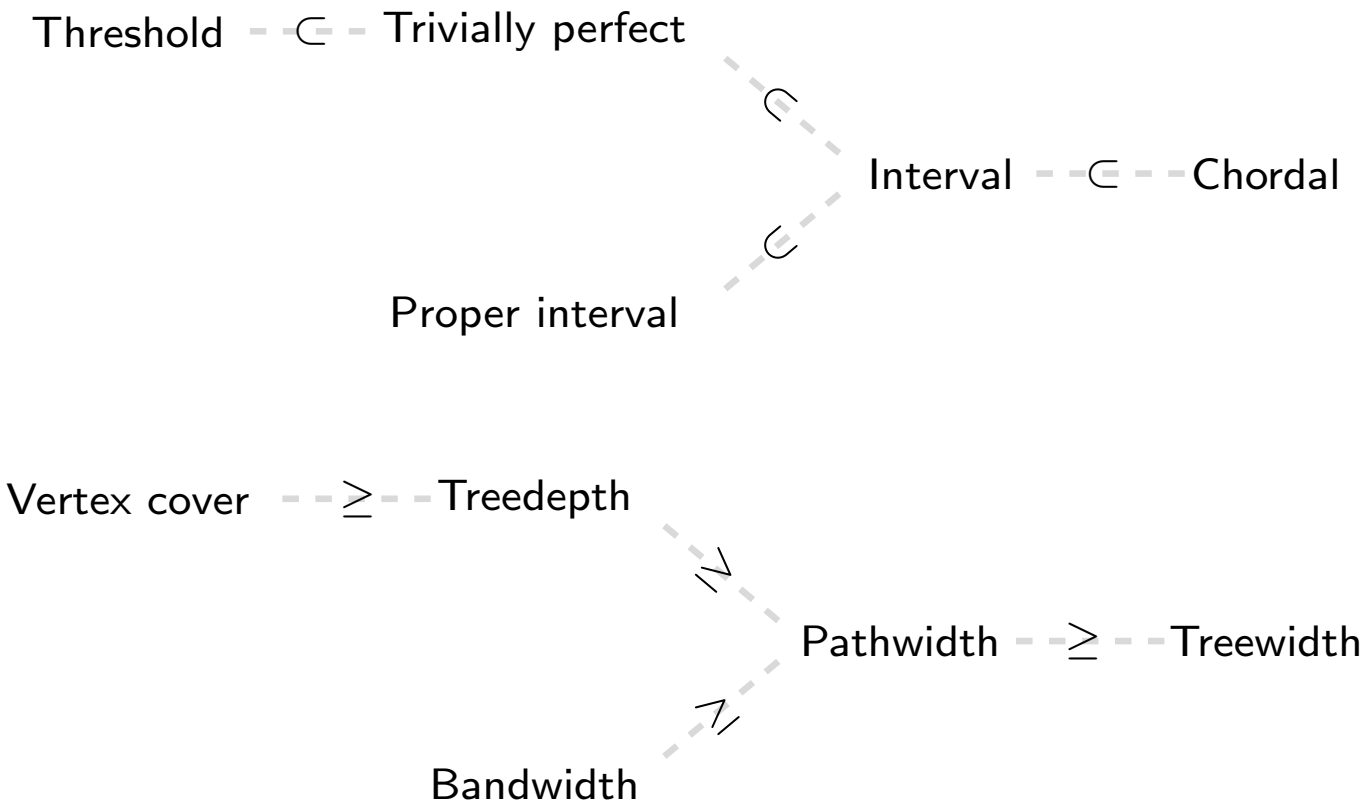
And what happens in the paper really

- Almost all the technical details hidden in this sketch.
- Instead of interval model, we work on an *umbrella ordering*.
- Perfect twins: need to canonize the model to break the ties.
- Many more details in the second-layer DP.

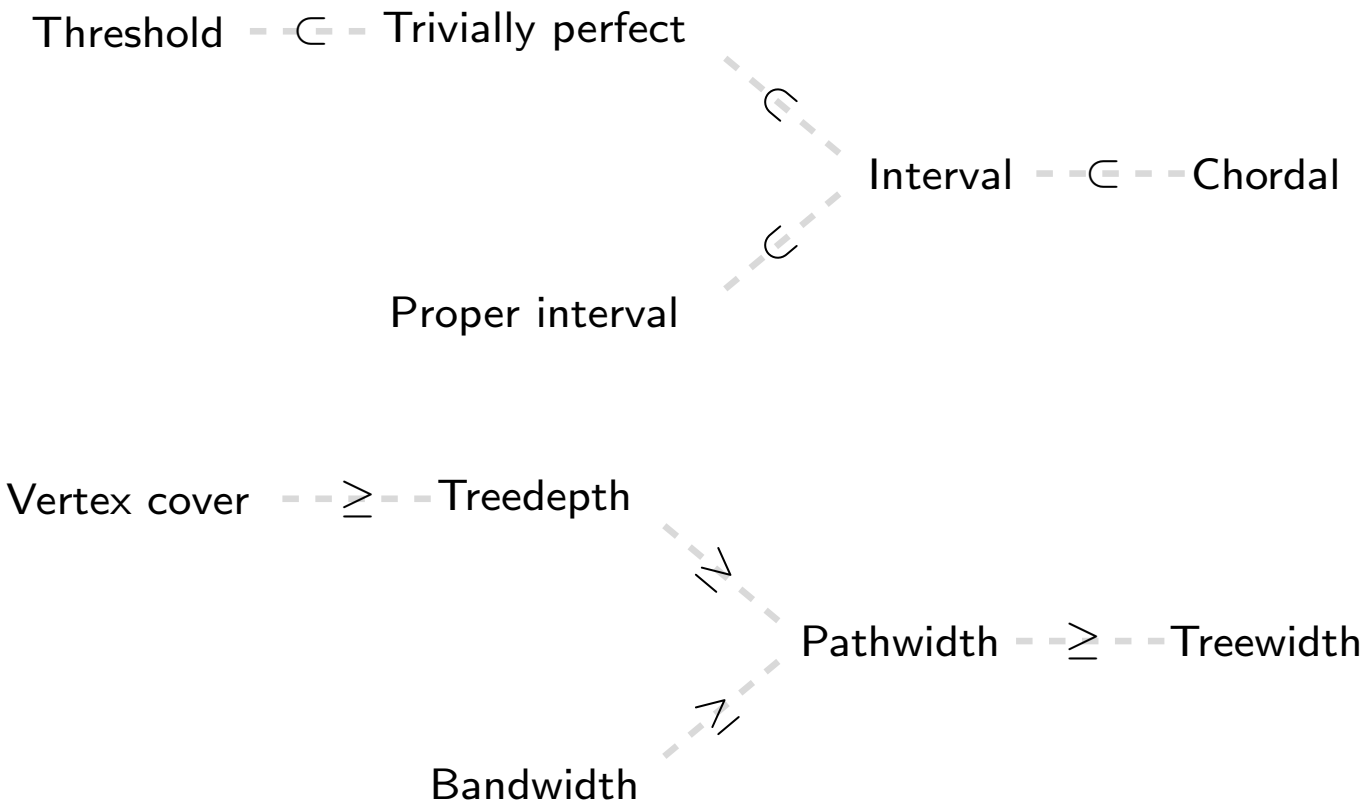
Conclusions



Conclusions



Conclusions



- **Correspondence**: parameter $p(\cdot)$ is the minimum possible maximum clique size in a completion to class Π (minus 1).

... and open problems

- We gave a $k^{\mathcal{O}(k^{2/3})} + \mathcal{O}(nm(kn+m))$ algorithm for Proper Interval Completion.

... and open problems

- We gave a $k^{\mathcal{O}(k^{2/3})} + \mathcal{O}(nm(kn+m))$ algorithm for Proper Interval Completion.
- Seems like existence of subexponential parameterized algorithms is connected to existence of a clique-like decomposition.

... and open problems

- We gave a $k^{\mathcal{O}(k^{2/3})} + \mathcal{O}(nm(kn+m))$ algorithm for Proper Interval Completion.
- Seems like existence of subexponential parameterized algorithms is connected to existence of a clique-like decomposition.
- **Open**: obtain a $k^{\mathcal{O}(k^{1/2})} \cdot n^{\mathcal{O}(1)}$-time algorithm.

... and open problems

- We gave a $k^{\mathcal{O}(k^{2/3})} + \mathcal{O}(nm(kn+m))$ algorithm for Proper Interval Completion.
- Seems like existence of subexponential parameterized algorithms is connected to existence of a clique-like decomposition.
- **Open**: obtain a $k^{\mathcal{O}(k^{1/2})} \cdot n^{\mathcal{O}(1)}$-time algorithm.
  - **Now**: Layer-two DP state is a τ-tuple of objects from a set of size roughly k^{τ}.

... and open problems

- We gave a $k^{\mathcal{O}(k^{2/3})} + \mathcal{O}(nm(kn+m))$ algorithm for Proper Interval Completion.
- Seems like existence of subexponential parameterized algorithms is connected to existence of a clique-like decomposition.
- **Open**: obtain a $k^{\mathcal{O}(k^{1/2})} \cdot n^{\mathcal{O}(1)}$-time algorithm.
  - **Now**: Layer-two DP state is a τ-tuple of objects from a set of size roughly k^{τ}.
- **Open**: obtain a lower bound excluding $2^{o(k^{1/2})} \cdot n^{\mathcal{O}(1)}$ algorithms for the completion problems.

... and open problems

- We gave a $k^{\mathcal{O}(k^{2/3})} + \mathcal{O}(nm(kn+m))$ algorithm for Proper Interval Completion.
- Seems like existence of subexponential parameterized algorithms is connected to existence of a clique-like decomposition.
- **Open**: obtain a $k^{\mathcal{O}(k^{1/2})} \cdot n^{\mathcal{O}(1)}$-time algorithm.
  - **Now**: Layer-two DP state is a τ-tuple of objects from a set of size roughly k^{τ}.
- **Open**: obtain a lower bound excluding $2^{o(k^{1/2})} \cdot n^{\mathcal{O}(1)}$ algorithms for the completion problems.
- Known reductions exclude a $2^{o(k^{1/6})} \cdot n^{\mathcal{O}(1)}$ algorithm under ETH.

... and open problems

- We gave a $k^{\mathcal{O}(k^{2/3})} + \mathcal{O}(nm(kn+m))$ algorithm for Proper Interval Completion.
- Seems like existence of subexponential parameterized algorithms is connected to existence of a clique-like decomposition.
- **Open**: obtain a $k^{\mathcal{O}(k^{1/2})} \cdot n^{\mathcal{O}(1)}$-time algorithm.
  - **Now**: Layer-two DP state is a τ-tuple of objects from a set of size roughly k^{τ}.
- **Open**: obtain a lower bound excluding $2^{o(k^{1/2})} \cdot n^{\mathcal{O}(1)}$ algorithms for the completion problems.
- Known reductions exclude a $2^{o(k^{1/6})} \cdot n^{\mathcal{O}(1)}$ algorithm under ETH.
- **Thank you for attention!**