The $k$-Sparsest Subgraph Problem in (Proper) Interval Graphs

Rémi Watrigant, Marin Bougeret and Rodolphe Giroudeau

LIRMM, Montpellier, France

Séminaire AlGCo, 20/09/2012
## Contents

1. **Introduction**

2. **FPT Algorithm in Interval Graphs**

3. **PTAS in Proper Interval Graphs**

4. **Open Problems and Future Work**
**Introduction**

**k-Sparsest Subgraph Problem (k-SS)**

**Input:** a graph $G = (V, E)$, $k \leq |V|$.

**Output:** a set $S \subseteq V$ of size exactly $k$.

**Goal:** minimize $E(S)$ (the number of edges induced by $S$)
**Introduction**

**$k$-Sparsest Subgraph Problem (k-SS)**

**Input:** a graph $G = (V, E)$, $k \leq |V|$.

**Output:** a set $S \subseteq V$ of size exactly $k$.

**Goal:** minimize $E(S)$ (the number of edges induced by $S$)

- generalization of independent set
  - $k$-SS $NP$-hard in general graphs (+ $W[1]$-hard, inapproximable)

Watrigant, Bougeret, Giroudeau
Introduction

$k$-Sparsest Subgraph Problem ($k$-SS)

**Input:** a graph $G = (V, E)$, $k \leq |V|$.
**Output:** a set $S \subseteq V$ of size exactly $k$.
**Goal:** minimize $E(S)$ (the number of edges induced by $S$)

- generalization of independent set
  $\Rightarrow$ $k$-SS $NP$-hard in general graphs (+ $W[1]$-hard, inapproximable)

- maximization version ($k$-Densest Subgraph) $NP$-hard on chordal graphs
  $\Rightarrow$ $k$-SS $NP$-hard in co-chordal $\subseteq$ perfect graphs
**Introduction**

**$k$-Sparsest Subgraph Problem ($k$-SS)**

**Input:** a graph $G = (V, E)$, $k \leq |V|$.

**Output:** a set $S \subseteq V$ of size exactly $k$.

**Goal:** minimize $|E(S)|$ (the number of edges induced by $S$)

- Generalization of independent set
  - $k$-SS $NP$-hard in general graphs (+ $W[1]$-hard, inapproximable)

- Maximization version ($k$-Densest Subgraph) $NP$-hard on chordal graphs
  - $k$-SS $NP$-hard in co-chordal $\subseteq$ perfect graphs

- $k$-SS polynomial in split graphs
Introduction

**k-Sparsest Subgraph Problem (k-SS)**

**Input:** a graph $G = (V, E)$, $k \leq |V|$.

**Output:** a set $S \subseteq V$ of size exactly $k$.

**Goal:** minimize $E(S)$ (the number of edges induced by $S$)

- generalization of independent set
  $\Rightarrow$ k-SS NP-hard in general graphs (+ W[1]-hard, inapproximable)

- maximization version (k-Densest Subgraph) NP-hard on chordal graphs
  $\Rightarrow$ k-SS NP-hard in co-chordal $\subseteq$ perfect graphs

- k-SS polynomial in split graphs

- complexity of k-DS unknown in (proper) interval graphs. PTAS in interval graphs, 3-approximation in chordal graphs
**k-Sparsest Subgraph Problem (k-SS)**

**Input:** a graph $G = (V, E)$, $k \leq |V|$.

**Output:** a set $S \subseteq V$ of size exactly $k$.

**Goal:** minimize $E(S)$ (the number of edges induced by $S$)

In this talk:
Introduction

**k-Sparsest Subgraph Problem (k-SS)**

**Input:** a graph $G = (V, E)$, $k \leq |V|$.

**Output:** a set $S \subseteq V$ of size exactly $k$.

**Goal:** minimize $E(S)$ (the number of edges induced by $S$)

In this talk:

- FPT algorithm in interval graphs (parameterized by the cost of the solution)
- PTAS in proper interval graphs
Introduction

$k$-Sparsest Subgraph Problem ($k$-SS)

**Input:** a graph $G = (V, E)$, $k \leq |V|$.

**Output:** a set $S \subseteq V$ of size exactly $k$.

**Goal:** minimize $E(S)$ (the number of edges induced by $S$)

In this talk:
- FPT algorithm in interval graphs (parameterized by the cost of the solution)
- PTAS in proper interval graphs

FPT Algorithm

An *FPT* algorithm for a parameterized problem is an algorithm that exactly solves the problem in $O(f(k) \cdot poly(n))$ where $n$ is the size of the instance and $k$ the parameter of the instance.

Polynomial-Time Approximation Scheme

A *PTAS* for a minimization problem is an algorithm $A_\epsilon$ such that for any fixed $\epsilon > 0$, $A_\epsilon$ runs in polynomial time and outputs a solution of cost $< (1 + \epsilon)OPT$
**Introduction**

### $k$-Sparsest Subgraph Problem ($k$-SS)

**Input:** a graph $G = (V, E)$, $k \leq |V|$.  
**Output:** a set $S \subseteq V$ of size exactly $k$.  
**Goal:** minimize $E(S)$ (the number of edges induced by $S$)

In this talk:
- FPT algorithm in interval graphs (parameterized by the cost of the solution)
- PTAS in proper interval graphs
Introduction

$k$-Sparsest Subgraph Problem ($k$-SS)

**Input:** a graph $G = (V, E)$, $k \leq |V|$.

**Output:** a set $S \subseteq V$ of size exactly $k$.

**Goal:** minimize $E(S)$ (the number of edges induced by $S$)

In this talk:
- FPT algorithm in interval graphs (parameterized by the cost of the solution)
- PTAS in proper interval graphs

Interval graphs = intersection graphs of intervals in the real line.
Introduction

\textbf{\textit{k}-Sparsest Subgraph Problem (k-SS)}

\textbf{Input:} a graph \( G = (V, E) \), \( k \leq |V| \).

\textbf{Output:} a set \( S \subseteq V \) of size exactly \( k \).

\textbf{Goal:} minimize \( E(S) \) (the number of edges induced by \( S \)).

In this talk:

- FPT algorithm in interval graphs (parameterized by the cost of the solution)
- PTAS in proper interval graphs

Interval graphs = intersection graphs of intervals in the real line.

Proper interval graph = no interval contains properly another one = unit interval graphs.
**Introduction**

**k-Sparsest Subgraph Problem (k-SS)**

**Input:** a graph $G = (V, E)$, $k \leq |V|$.

**Output:** a set $S \subseteq V$ of size exactly $k$.

**Goal:** minimize $E(S)$ (the number of edges induced by $S$).

In this talk:
- FPT algorithm in interval graphs (parameterized by the cost of the solution)
- PTAS in proper interval graphs

Interval graphs = intersection graphs of intervals in the real line.

Proper interval graph = no interval contains properly another one = unit interval graphs
Contents

1 Introduction

2 FPT Algorithm in Interval Graphs

3 PTAS in Proper Interval Graphs

4 Open Problems and Future Work
FPT Algorithm in Interval Graphs

Given a set $\mathcal{I}$ of intervals, $k \leq |\mathcal{I}|$ and a cost $C^*$
FPT Algorithm in Interval Graphs

Given a set $\mathcal{I}$ of intervals, $k \leq |\mathcal{I}|$ and a cost $C^*$

Idea of the algorithm:
FPT Algorithm in Interval Graphs

Given a set $\mathcal{I}$ of intervals, $k \leq |\mathcal{I}|$ and a cost $C^*$

Idea of the algorithm:

- we sort intervals according to their right endpoints
FPT Algorithm in Interval Graphs

Given a set $\mathcal{I}$ of intervals, $k \leq |\mathcal{I}|$ and a cost $C^*$

Idea of the algorithm:

- we sort intervals according to their right endpoints
- parameters of the dynamic programming:
  $s \leftarrow$ left endpoint of the leftmost interval, $k' \leftarrow k$, $C' \leftarrow C^*$
FPT Algorithm in Interval Graphs

Given a set $\mathcal{I}$ of intervals, $k \leq |\mathcal{I}|$ and a cost $C^*$

Idea of the algorithm:

- we sort intervals according to their right endpoints
- parameters of the dynamic programming:
  $s \leftarrow$ left endpoint of the leftmost interval, $k' \leftarrow k$, $C' \leftarrow C^*$
- given the parameters, we construct all subsets $T$ s.t.
FPT Algorithm in Interval Graphs

Given a set $\mathcal{I}$ of intervals, $k \leq |\mathcal{I}|$ and a cost $C^*$

Idea of the algorithm:

- we sort intervals according to their right endpoints
- parameters of the dynamic programming:
  
  $s \leftarrow$ left endpoint of the leftmost interval, $k' \leftarrow k$, $C' \leftarrow C^*$

- given the parameters, we construct all subsets $T$ s.t.
  
  (i) $T$ is connected
  (ii) $T$ starts after $s$ (i.e. to the right of $s$)
  (iii) $E(T) \leq C'$
FPT Algorithm in Interval Graphs

Given a set \( \mathcal{I} \) of intervals, \( k \leq |\mathcal{I}| \) and a cost \( C^* \)

Idea of the algorithm:
- we sort intervals according to their right endpoints
- parameters of the dynamic programming:
  \( s \leftarrow \) left endpoint of the leftmost interval, \( k' \leftarrow k \), \( C' \leftarrow C^* \)
- given the parameters, we construct all subsets \( T \) s.t.
  (i) \( T \) is connected
  (ii) \( T \) starts after \( s \) (i.e. to the right of \( s \))
  (iii) \( E(T) \leq C' \)
- recursive call with :
  - \( k' \leftarrow k' - |T| \)
  - \( C' \leftarrow C - E(T) \)
  - \( s \leftarrow \) left endpoint of the rightmost interval after \( T \)
FPT Algorithm in Interval Graphs

Given a set $\mathcal{I}$ of intervals, $k \leq |\mathcal{I}|$ and a cost $C^*$

Idea of the algorithm:

- we sort intervals according to their right endpoints
- parameters of the dynamic programming:
  - $s \leftarrow$ left endpoint of the leftmost interval, $k' \leftarrow k$, $C' \leftarrow C^*$
- given the parameters, we construct all subsets $T$ s.t.
  - (i) $T$ is connected
  - (ii) $T$ starts after $s$ (i.e. to the right of $s$)
  - (iii) $E(T) \leq C'$
- recursive call with :
  - $k' \leftarrow k' - |T|$  
  - $C' \leftarrow C - E(T)$  
  - $s \leftarrow$ left endpoint of the rightmost interval after $T$
- $\Rightarrow$ at most $k.C^*.n$ different inputs
  - what about the running time of one call?
FPT Algorithm in Interval Graphs

Given a set $\mathcal{I}$ of intervals, $k \leq |\mathcal{I}|$ and a cost $C^*$

Idea of the algorithm:
- we sort intervals according to their right endpoints
- parameters of the dynamic programming:
  - $s \leftarrow$ left endpoint of the leftmost interval, $k' \leftarrow k$, $C' \leftarrow C^*$
- given the parameters, we construct all subsets $T$ s.t.
  1. $T$ is connected
  2. $T$ starts after $s$ (i.e. to the right of $s$)
  3. $E(T) \leq C'$
- recursive call with:
  - $k' \leftarrow k' - |T|$
  - $C' \leftarrow C - E(T)$
  - $s \leftarrow$ left endpoint of the rightmost interval after $T$

$\Rightarrow$ at most $k.C^*.n$ different inputs
what about the running time of one call?

Let $\Omega_s(C')$ be the set of all subsets satisfying (i), (ii) and (iii)
FPT Algorithm in Interval Graphs

- Given the parameters, we construct all subsets $T$ s.t.
  - (i) $T$ is connected
  - (ii) $T$ starts after $s$ (i.e. to the right of $s$)
  - (iii) $E(T) \leq C'$

Let $\Omega_s(C')$ be the set of all subsets satisfying (i), (ii) and (iii)
FPT Algorithm in Interval Graphs

Given the parameters, we construct all subsets $T$ s.t.

(i) $T$ is connected
(ii) $T$ starts after $s$ (i.e. to the right of $s$)
(iii) $E(T) \leq C'$

Let $\Omega_s(C')$ be the set of all subsets satisfying (i), (ii) and (iii)

**Lemma**

$\Omega_s(C')$

\[ \begin{array}{c}
T_1 \\
T_2 \\
\vdots \\
T_l
\end{array} \]

\[ \Gamma_s(C') \]

\[ \begin{array}{c}
T'_1 \\
T'_2 \\
\vdots \\
T'_{l'}
\end{array} \]

Cost

Restructuration

Can be enumerated in FPT time

$< y_1, \ldots, y_i, \ldots, y_t >$
FPT Algorithm in Interval Graphs
FPT Algorithm in Interval Graphs

$S$

$I_{i_1}$: leftmost interval of $T$ crossing $s$
FPT Algorithm in Interval Graphs

\( l_{i_1} \): leftmost interval of \( T \) crossing \( s \)

\( l^* \): leftmost interval after \( s \) overlapping \( l_{i_1} \)
FPT Algorithm in Interval Graphs

\[ l_{i_1} : \text{leftmost interval of } T \text{ crossing } s \]

\[ \tilde{l} : \text{leftmost interval of } T \text{ overlapping } l_{i_1} \]

\[ l^* : \text{leftmost interval after } s \text{ overlapping } l_{i_1} \]
FPT Algorithm in Interval Graphs

\( l_{i_1} \): leftmost interval of \( T \) crossing \( s \)

\( \tilde{l} \): leftmost interval of \( T \) overlapping \( l_{i_1} \)

\( l^* \): leftmost interval after \( s \) overlapping \( l_{i_1} \)
$l_i$: leftmost interval of $T$ crossing $s$

$\tilde{l}$: leftmost interval of $T$ overlapping $l_i$

$l^*$: leftmost interval after $s$ overlapping $l_i$
FPT Algorithm in Interval Graphs

$l_{i_1}$: leftmost interval of $T$ crossing $s$

$l^*$: leftmost interval after $s$ overlapping $l_{i_1}$
$y_i = 1$

$l_{i_1}$: leftmost interval of $T$ crossing $s$

$l^*$: leftmost interval after $s$ overlapping $l_{i_1}$
$y_i = 1 + x$

$S$

$l_{i_1}$: leftmost interval of $T$ crossing $s$

$l^*$: leftmost interval after $s$ overlapping $l_{i_1}$
$y_i = 1 + x$

$I_i^\ast$: leftmost interval after $s$ overlapping $I_{i_1}$

$I_i$: leftmost interval of $T$ crossing $s$
$y_i = 1 + x$

$l_{i_1}$: leftmost interval of $T$ crossing $s$

$l^*$: leftmost interval after $s$ overlapping $l_{i_1}$

$x$ leftmost intervals after $s$ overlapping $l_{i_1}$
Any element of $\Gamma_s(C')$ can be encoded by a vector $< y_1, \ldots, y_i, \ldots, y_t >$

We now bound the size of $\Gamma_s(C^*)$:

- $y_i = B \Rightarrow$ there exists a clique of size $B$ in the solution
Any element of $\Gamma_s(C')$ can be encoded by a vector $< y_1, ..., y_i, ..., y_t >$

We now bound the size of $\Gamma_s(C^*)$:

- $y_i = B \Rightarrow$ there exists a clique of size $B$ in the solution
  $\Rightarrow y_i \leq \sqrt{2C^*} + 2$
Any element of $\Gamma_s(C')$ can be encoded by a vector $< y_1, ..., y_i, ..., y_t >$

We now bound the size of $\Gamma_s(C^*)$:

- $y_i = B \Rightarrow$ there exists a clique of size $B$ in the solution
  $\Rightarrow y_i \leq \sqrt{2C^*} + 2$

- for each step $i \in \{0, ..., (t - 1)\}$ and corresponding $s$, we can find a pair of intervals (crossing or at the right of $s$) overlapping such that in the next step, one of them is at the left of $s$ (no multiple counts of the same pair)
Any element of $\Gamma_s(C')$ can be encoded by a vector $<y_1, \ldots, y_i, \ldots, y_t>$

We now bound the size of $\Gamma_s(C^*)$:

- $y_i = B \Rightarrow$ there exists a clique of size $B$ in the solution
  $\Rightarrow y_i \leq \sqrt{2C^*} + 2$

- for each step $i \in \{0, \ldots, (t - 1)\}$ and corresponding $s$, we can find a pair of intervals (crossing or at the right of $s$) overlapping such that in the next step, one of them is at the left of $s$ (no multiple counts of the same pair)
  $\Rightarrow t \leq C^* + 1$
Any element of $\Gamma_s(C')$ can be encoded by a vector $<y_1, ..., y_i, ..., y_t>$

We now bound the size of $\Gamma_s(C^*)$:

- $y_i = B \Rightarrow$ there exists a clique of size $B$ in the solution
  $\Rightarrow y_i \leq \sqrt{2C^*} + 2$

- for each step $i \in \{0, ..., (t - 1)\}$ and corresponding $s$, we can find a pair of intervals (crossing or at the right of $s$) overlapping such that in the next step, one of them is at the left of $s$ (no multiple counts of the same pair)
  $\Rightarrow t \leq C^* + 1$

Thus:

$$|\Gamma_s(C^*)| \leq (\sqrt{2C^*} + 2)^{C^*+1}$$

and each step of the dynamic programming runs in $FPT$ time.
Any element of $\Gamma_s(C')$ can be encoded by a vector $< y_1, \ldots, y_i, \ldots, y_t >$

We now bound the size of $\Gamma_s(C^*)$:

- $y_i = B \Rightarrow$ there exists a clique of size $B$ in the solution
  $\Rightarrow y_i \leq \sqrt{2C^* + 2}$

- for each step $i \in \{0, \ldots, (t - 1)\}$ and corresponding $s$, we can find a pair of intervals (crossing or at the right of $s$) overlapping such that in the next step, one of them is at the left of $s$ (no multiple counts of the same pair)
  $\Rightarrow t \leq C^* + 1$

Thus:

$$|\Gamma_s(C^*)| \leq (\sqrt{2C^* + 2})^{C^*+1}$$

and each step of the dynamic programming runs in $FPT$ time.

**Theorem**

$k$-Sparsest Subgraph in Interval Graphs is $FPT$ parameterized by the cost of the solution.
1 Introduction

2 FPT Algorithm in Interval Graphs

3 PTAS in Proper Interval Graphs

4 Open Problems and Future Work
PTAS in Proper Interval Graphs

Idea of the algorithm:
PTAS in Proper Interval Graphs

Idea of the algorithm:

- sorting intervals according to their right endpoints
PTAS in Proper Interval Graphs

Idea of the algorithm:

- sorting intervals according to their right endpoints
- greedy decomposition of the graph into a path of separators
PTAS in Proper Interval Graphs

Idea of the algorithm:
- sorting intervals according to their right endpoints
- greedy decomposition of the graph into a path of separators
- re-structuration of an optimal solution into a near optimal solution such that all near optimal solutions can be enumerated in polynomial time
Idea of the algorithm:

- sorting intervals according to their right endpoints
- greedy decomposition of the graph into a path of separators
- re-structuration of an optimal solution into a near optimal solution such that all near optimal solutions can be enumerated in polynomial time
- dynamic programming processes the graph through the decomposition, enumerating all possible solutions.
PTAS in Proper Interval Graphs

The decomposition:
PTAS in Proper Interval Graphs

The decomposition:

\[ l_{m_1} \]
PTAS in Proper Interval Graphs

The decomposition:

\[ R_1 \quad I_{m_1} \quad B_1 \]
PTAS in Proper Interval Graphs

The decomposition:

\[ R_1 \quad \text{or} \quad B_1 \]
PTAS in Proper Interval Graphs

The decomposition:

\[ I_{m_1} \quad R_1 \quad L_2 \quad B_1 \quad I_{m_2} \]
PTAS in Proper Interval Graphs

The decomposition:
PTAS in Proper Interval Graphs

The decomposition

Remark

The only edges between blocks $B_i$ and $B_{i+1}$ are between $R_i$ and $L_{i+1}$. Given $S \subseteq \mathcal{I}$ we have:

$$E(S) = \sum_{i=1}^{a} E(B_i \cap S) + \sum_{i=1}^{a-1} E(R_i \cap S, L_{i+1} \cap S)$$
Compaction

Let $S \subseteq \mathcal{I}$ be a solution, and $S^c = \text{comp}(S) \subseteq \mathcal{I}$ such that for each block $i \in \{1, \ldots, a\}$:

- for all $I \in L_i$, $\text{comp}(I) \in L_i$ and is at the right of $I$ (we may have $\text{comp}(I) = I$)
- for all $I \in R_i$, $\text{comp}(I) \in R_i$ and is at the left of $I$ (we may have $\text{comp}(I) = I$)
Compaction

Let $S \subseteq \mathcal{I}$ be a solution, and $S^c = comp(S) \subseteq \mathcal{I}$ such that for each block $i \in \{1, ..., a\}$:

- for all $I \in L_i$, $comp(I) \in L_i$ and is at the right of $I$ (we may have $comp(I) = I$)
- for all $I \in R_i$, $comp(I) \in R_i$ and is at the left of $I$ (we may have $comp(I) = I$)
Compaction

Let $S \subseteq I$ be a solution, and $S^c = \text{comp}(S) \subseteq I$ such that for each block $i \in \{1, ..., a\}$:

- for all $l \in L_i$, $\text{comp}(l) \in L_i$ and is at the right of $l$ (we may have $\text{comp}(l) = l$)
- for all $l \in R_i$, $\text{comp}(l) \in R_i$ and is at the left of $l$ (we may have $\text{comp}(l) = l$)
Compaction

Let $S \subseteq \mathcal{I}$ be a solution, and $S^c = \text{comp}(S) \subseteq \mathcal{I}$ such that for each block $i \in \{1, \ldots, a\}$:

- for all $l \in L_i$, $\text{comp}(l) \in L_i$ and is at the right of $l$ (we may have $\text{comp}(l) = l$)
- for all $l \in R_i$, $\text{comp}(l) \in R_i$ and is at the left of $l$ (we may have $\text{comp}(l) = l$)
PTAS in Proper Interval Graphs
Re-structuration of optimal solutions

**Compaction**

Let $S \subseteq \mathcal{I}$ be a solution, and $S^c = \text{comp}(S) \subseteq \mathcal{I}$ such that for each block $i \in \{1, ..., a\}$:
- for all $I \in L_i$, $\text{comp}(I) \in L_i$ and is at the right of $I$ (we may have $\text{comp}(I) = I$)
- for all $I \in R_i$, $\text{comp}(I) \in R_i$ and is at the left of $I$ (we may have $\text{comp}(I) = I$)

**Lemma**

If $\text{comp}$ is a compaction of a solution $S$ such that for all block $i \in \{1, ..., a\}$, we have

$$E(\text{comp}(S \cap B_i)) \leq \rho E(S \cap B_i)$$

Then $\text{comp}(S)$ is a $\rho$-approximation of $S$. 
PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

Let us build a compaction that yields a \((1 + \frac{4}{P})\)-approximation for any fixed \(P\).
PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

Let us built a compaction that yields a \((1 + \frac{4}{P})\)-approximation for any fixed \(P\). Let \(X \subseteq B_i\) be a solution. We note \(X = X_L \cup X_R\). Set sizes are in lowercase.
PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

Let us built a compaction that yields a \((1 + \frac{4}{P})\)-approximation for any fixed \(P\).

Let \(X \subseteq B_i\) be a solution. We note \(X = X_L \cup X_R\). Set sizes are in lowercase.

- we divide \(X_L\) into \(P\) consecutive subsets of same size \(q_L \rightarrow X^L_1, \ldots, X^L_P\)
- we divide \(X_R\) into \(P\) consecutive subsets of same size \(q_R \rightarrow X^R_1, \ldots, X^R_P\)

Then define the compaction: for any \(t \in \{1, \ldots, P\}\)
PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

Let us built a compaction that yields a \((1 + \frac{4}{P})\)-approximation for any fixed \(P\). Let \(X \subseteq B_i\) be a solution. We note \(X = X_L \cup X_R\). Set sizes are in lowercase.

- we divide \(X_L\) into \(P\) consecutive subsets of same size \(q_L \rightarrow X_{L1}^L, \ldots, X_{LP}^L\)
- we divide \(X_R\) into \(P\) consecutive subsets of same size \(q_R \rightarrow X_{R1}^R, \ldots, X_{RP}^R\)

Then define the compaction: for any \(t \in \{1, \ldots, P\}\)

- \(Y_{Lt}^L\) are the \(q_L\) rightmost intervals at the left of the rightmost interval of \(X_{Lt}^L\)
- \(Y_{Rt}^R\) are the \(q_R\) leftmost intervals at the right of the leftmost interval of \(X_{Rt}^R\)
PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

What do we need to construct such a solution?
PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

What do we need to construct such a solution?

- the leftmost interval of $X_t^L$ for $t \in \{1, ..., P\}$
- the rightmost interval of $X_t^R$ for $t \in \{1, ..., P\}$
- $x_R$, $x_L$ (plus remainders of divisions by $P$...)

$\Rightarrow 2P + O(1)$ variables ranging in $\{0, ..., n\}$
Sketch of proof of the \((1 + \frac{4}{\bar{P}})\) approximation ratio:
PTAS in Proper Interval Graphs

Sketch of proof of the $(1 + \frac{4}{P})$ approximation ratio:

- **OPT** = $\binom{x_L}{2} + \binom{x_R}{2} + \sum_{t=1}^{a} \sum_{u=1}^{a} E(X^L_t, X^R_u)$
- **SOL** = $\binom{y_L}{2} + \binom{y_R}{2} + \sum_{t=1}^{a} \sum_{u=1}^{a} E(Y^L_t, Y^R_u)$
Sketch of proof of the $(1 + \frac{4}{P})$ approximation ratio:

- $OPT = (\frac{x_L}{2}) + (\frac{x_R}{2}) + \sum_{t=1}^{a} \sum_{u=1}^{a} E(X_t^L, X_u^R)$
- $SOL = (\frac{X_L}{2}) + (\frac{X_R}{2}) + \sum_{t=1}^{a} \sum_{u=1}^{a} E(Y_t^L, Y_u^R)$

But:
PTAS in Proper Interval Graphs

Sketch of proof of the \((1 + \frac{4}{P})\) approximation ratio:

- **OPT** = \((\frac{x_L}{2}) + (\frac{x_R}{2}) + \sum_{t=1}^{a} \sum_{u=1}^{a} E(X_t^L, X_u^R)\)
- **SOL** = \((\frac{y_L}{2}) + (\frac{y_R}{2}) + \sum_{t=1}^{a} \sum_{u=1}^{a} E(Y_t^L, Y_u^R)\)

But:

- if some intervals of \(Y_t^L\) overlap some intervals of \(Y_u^R\)

Then:

- all intervals of \(X_{t+1}^L\) overlap all intervals of \(\bigcup_{i=1}^{u-1} X_i^R\)
Sketch of proof of the \((1 + \frac{4}{P})\) approximation ratio:

- \(OPT = \binom{x_L}{2} + \binom{x_R}{2} + \sum_{t=1}^{a} \sum_{u=1}^{a} E(X_t^L, X_u^R)\)
- \(SOL = \binom{x_L}{2} + \binom{x_R}{2} + \sum_{t=1}^{a} \sum_{u=1}^{a} E(Y_t^L, Y_u^R)\)

But:
- if some intervals of \(Y_t^L\) overlap some intervals of \(Y_u^R\)
Then:
- all intervals of \(X_{t+1}^L\) overlap all intervals of \(\bigcup_{i=1}^{u-1} X_i^R\)

Finally, we can prove that \(\frac{SOL}{OPT} \leq 1 + \frac{4}{P}\)
Concentration:

**Theorem**

For any $P$, the previous algorithm outputs a $(1 + \frac{4}{P})$-approximation for the $k$-Sparsest Subgraph in Proper Interval graphs in $O(n^{O(P)})$. 
Contents

1 Introduction

2 FPT Algorithm in Interval Graphs

3 PTAS in Proper Interval Graphs

4 Open Problems and Future Work
Open problems and Future Work

Complexity of $k$-Sparsest Subgraph:

- Perfect
- Bipartite
- Chordal
- Tree
- Interval
- Split
- Proper Int.
Open problems and Future Work

Complexity of $k$-Sparsest Subgraph:
Open problems and Future Work

Complexity of $k$-Sparsest Subgraph:

- Perfect
  - NP-hard

- Bipartite

- Chordal
  - Tree
    - Poly.

- Interval
  - Split
    - Poly.
  - Proper
    - Int.
Open problems and Future Work

Complexity of $k$-Sparsest Subgraph:

- Perfect: NP-hard
- Bipartite: NP-hard?
- Chordal: Poly.
- Tree Poly.
- Interval: Poly.
- Split Poly.
- Proper Int.
Open problems and Future Work

Complexity of $k$-Sparsest Subgraph:

- Perfect: NP-hard
- Bipartite: NP-hard?
- Chordal: Poly.
- Tree: Poly.
- Interval: Split Poly.
- Proper Int. PTAS
Open problems and Future Work

Complexity of $k$-Sparsest Subgraph:

- Perfect: NP-hard
- Bipartite: NP-hard?
- Chordal: Tree Poly.
- Interval: FPT
- Split Poly.
- Proper Int. PTAS

Watrigant, Bougeret, Giroudeau
Open problems and Future Work

Complexity of $k$-Sparsest Subgraph:

2 main objectives:
- extend FPT and/or approximation results to Chordal graphs
- NP-hardness for Chordal graphs
Thank you for your attention!