Finding a Sparse k-Subgraph in Restricted Graph Classes

Rémi Watrigant, Marin Bougeret and Rodolphe Giroudeau

LIRMM, Montpellier, France

Séminaire d'Optimisation Combinatoire
LAMSADE, Université Paris-Dauphine
25/02/2013

Contents

Sparsest k-Subgraph Problem (SkS)

Input: a simple undirected 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)

Sparsest k-Subgraph Problem (SkS)

Input: a simple undirected 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
  $\Rightarrow$ SkS NP-hard in general graphs (+ W[1]-hard, inapproximable)

Sparsest k-Subgraph Problem (SkS)

Input: a simple undirected 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
  $\Rightarrow$ SkS NP-hard in general graphs (+ W[1]-hard, inapproximable)

- Related problems:
  - maximisation version: Densest k-Subgraph (DkS)
    - exact result for DkS on $\mathcal{C}$ $\Leftrightarrow$ exact result for SkS on $\bar{\mathcal{C}}$
  - dual version: Maximum Vertex Coverage (MVC)
    - $S \subseteq V$ opt. solution for SkS $\Leftrightarrow$ $V \setminus S$ opt. solution for MVC(n-k)

but approximation do not transfer...

Sparsest k-Subgraph Problem (SkS)

Input: a simple undirected 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
  $\Rightarrow$ SkS NP-hard in general graphs (+ W[1]-hard, inapproximable)

- Related problems:
  - maximisation version: Densest k-Subgraph (DkS)
    - exact result for DkS on $\mathcal{C}$ $\Leftrightarrow$ exact result for SkS on $\bar{\mathcal{C}}$
  - dual version: Maximum Vertex Coverage (MVC)
    - $S \subseteq V$ opt. solution for SkS $\Leftrightarrow$ $V \setminus S$ opt. solution for MVC(n-k)

  but approximation do not transfer...

- we study SkS in classes where INDEPENDENT SET is polynomial-time solvable e.g. perfect graphs and their subclasses

In restricted graphs classes:

Densest k-Subgraph

Sparsest k-Subgraph

In restricted graphs classes:

Densest k-Subgraph

- Chordal graphs:
  NP-hard [Corneil,Perl,1984]
  3-approx. [Liazi,Milis,Zissimopoulos,2008]
  FPT ("obvious")

Sparsest k-Subgraph

- Chordal graphs:
  NP-hard [Bougeret,Giroudeau,W.,2013]

In restricted graphs classes:

Densest k-Subgraph

- Chordal graphs:
  NP-hard [Corneil,Perl,1984]
  3-approx. [Liazi,Milis,Zissimopoulos,2008]
  FPT ("obvious")
- Interval graphs:
  NP-h/Poly : open (since [Corneil,Perl,1984])
  PTAS [Nonner,2011]

Sparsest k-Subgraph

- Chordal graphs:
  NP-hard [Bougeret,Giroudeau,W.,2013]
- Interval graphs:
  NP-h/Poly : open
  2-approx. [Bougeret,Giroudeau,W.,2013]
  FPT [Bougeret,Giroudeau,W.,2013]

In restricted graphs classes:

Densest k-Subgraph

- Chordal graphs:
  NP-hard [Corneil,Perl,1984]
  3-approx. [Liazi,Milis,Zissimopoulos,2008]
  FPT ("obvious")
- Interval graphs:
  NP-h/Poly : open (since [Corneil,Perl,1984])
  PTAS [Nonner,2011]
- Proper interval graphs:
  NP-h/Poly : open

Sparsest k-Subgraph

- Chordal graphs:
  NP-hard [Bougeret,Giroudeau,W.,2013]
- Interval graphs:
  NP-h/Poly : open
  2-approx. [Bougeret,Giroudeau,W.,2013]
  FPT [Bougeret,Giroudeau,W.,2013]
- Proper interval graphs:
  NP-h/Poly : open
  PTAS [Bougeret,Giroudeau,W.,2013]

In restricted graphs classes:

Densest k-Subgraph

- Chordal graphs:
  NP-hard [Corneil,Perl,1984]
  3-approx. [Liazi,Milis,Zissimopoulos,2008]
  FPT ("obvious")
- Interval graphs:
  NP-h/Poly : open (since [Corneil,Perl,1984])
  PTAS [Nonner,2011]
- Proper interval graphs:
  NP-h/Poly : open
- Bipartite graphs:
  NP-h [Corneil,Perl,1984]

Sparsest k-Subgraph

- Chordal graphs:
  NP-hard [Bougeret,Giroudeau,W.,2013]
- Interval graphs:
  NP-h/Poly : open
  2-approx. [Bougeret,Giroudeau,W.,2013]
  FPT [Bougeret,Giroudeau,W.,2013]
- Proper interval graphs:
  NP-h/Poly : open
  PTAS [Bougeret,Giroudeau,W.,2013]
- Bipartite graphs:
  NP-h [Joret,Vetta,2012]

In restricted graphs classes:

Densest k-Subgraph

- Chordal graphs:
  NP-hard [Corneil,Perl,1984]
  3-approx. [Liazi,Milis,Zissimopoulos,2008]
  FPT ("obvious")
- Interval graphs:
  NP-h/Poly : open (since [Corneil,Perl,1984])
  PTAS [Nonner,2011]
- Proper interval graphs:
  NP-h/Poly : open
- Bipartite graphs:
  NP-h [Corneil,Perl,1984]
- Split graphs:
  Polynomial

Sparsest k-Subgraph

- Chordal graphs:
  NP-hard [Bougeret,Giroudeau,W.,2013]
- Interval graphs:
  NP-h/Poly : open
  2-approx. [Bougeret,Giroudeau,W.,2013]
  FPT [Bougeret,Giroudeau,W.,2013]
- Proper interval graphs:
  NP-h/Poly : open
  PTAS [Bougeret,Giroudeau,W.,2013]
- Bipartite graphs:
  NP-h [Joret,Vetta,2012]
- Split graphs:
  Polynomial

In restricted graphs classes:

Densest k-Subgraph

- Chordal graphs:
  NP-hard [Corneil,Perl,1984]
  3-approx. [Liazi,Milis,Zissimopoulos,2008]
  FPT ("obvious")
- Interval graphs:
  NP-h/Poly : open (since [Corneil,Perl,1984])
  PTAS [Nonner,2011]
- Proper interval graphs:
  NP-h/Poly : open
- Bipartite graphs:
  NP-h [Corneil,Perl,1984]
- Split graphs:
  Polynomial
- Planar graphs:
  NP-h/Poly : open

Sparsest k-Subgraph

- Chordal graphs:
  NP-hard [Bougeret,Giroudeau,W.,2013]
- Interval graphs:
  NP-h/Poly : open
  2-approx. [Bougeret,Giroudeau,W.,2013]
  FPT [Bougeret,Giroudeau,W.,2013]
- Proper interval graphs:
  NP-h/Poly : open
  PTAS [Bougeret,Giroudeau,W.,2013]
- Bipartite graphs:
  NP-h [Joret,Vetta,2012]
- Split graphs:
  Polynomial
- Planar graphs:
  NP-h (from Independent Set)
  FPT ("obvious" linear kernel)

In this talk:

- *PTAS* in Proper Interval graphs.
- *FPT* algorithm in Interval graphs parameterized by the number of edges in the solution (stronger parameterization than by k).

Definitions

Polynomial-Time Approximation Scheme (PTAS)

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

Definitions

Polynomial-Time Approximation Scheme (PTAS)

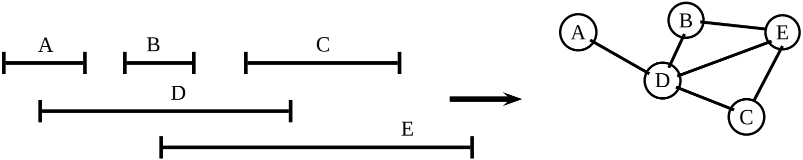
A *PTAS* for a minimization problem is an algorithm $\mathcal{A}_\epsilon$ such that for any fixed $\epsilon > 0$, $\mathcal{A}_\epsilon$ runs in polynomial time and outputs a solution of cost $< (1+\epsilon)OPT$

Fixed-Parameter Tractable (FPT)

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

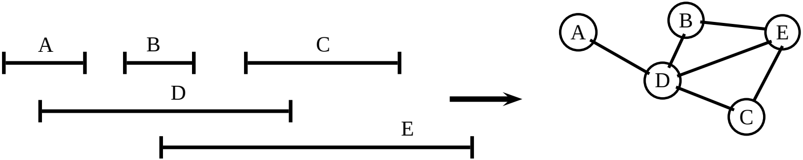
Definitions

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



Definitions

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



proper interval graph = no interval contains properly another one = unit interval graphs

Contents

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

PTAS in Proper Interval Graphs

Idea of the algorithm:

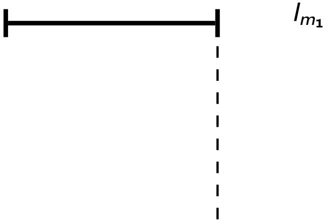
- sorting intervals according to their right endpoints
- greedy decomposition of the graph into a path of separators
- re-structuration of an optimal solution into a near optimal solution such that all near optimal solutions can be enumerated in polynomial time
- dynamic programming processes the graph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

PTAS in Proper Interval Graphs

The decomposition:

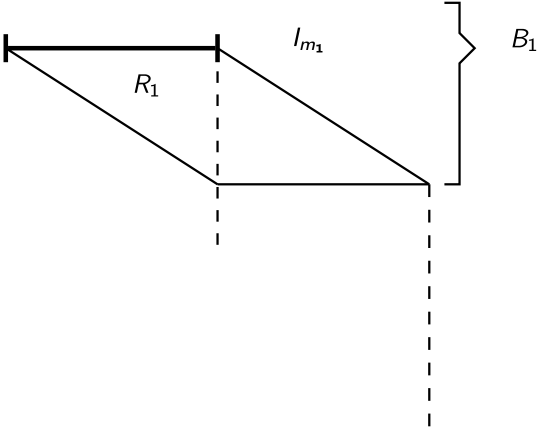
PTAS in Proper Interval Graphs

The decomposition:



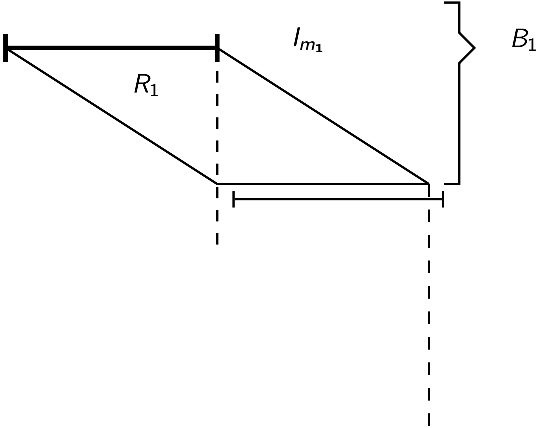
PTAS in Proper Interval Graphs

The decomposition:



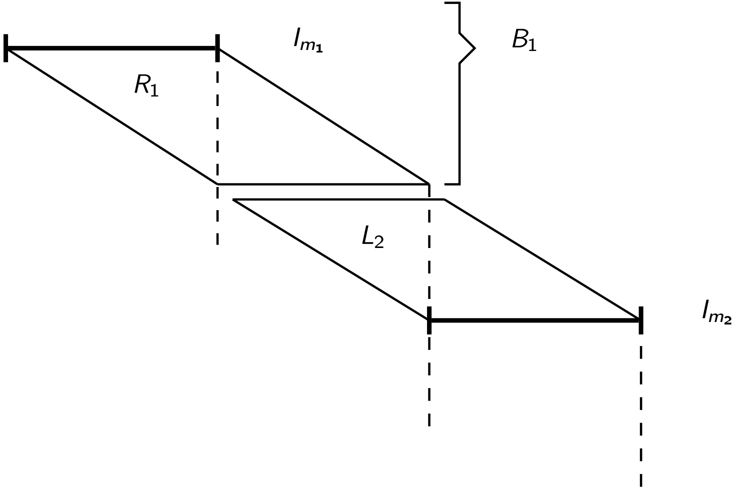
PTAS in Proper Interval Graphs

The decomposition:



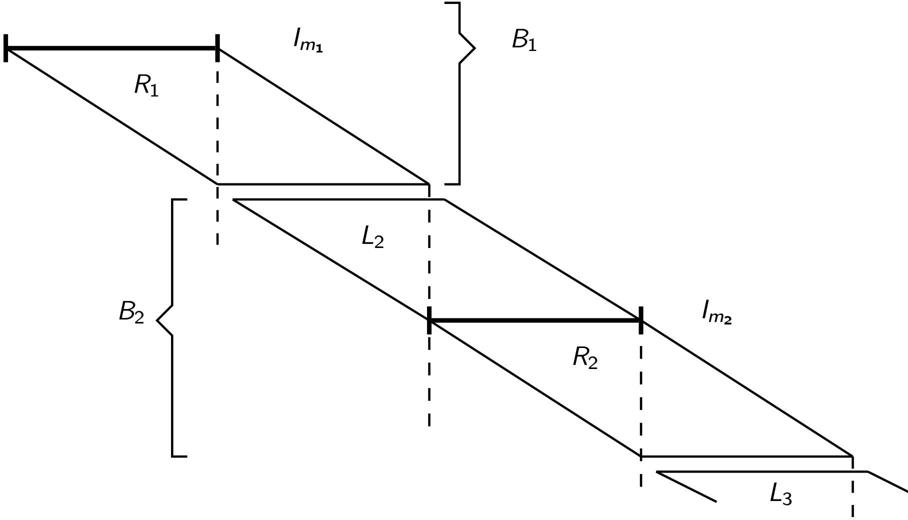
PTAS in Proper Interval Graphs

The decomposition:



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:

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

PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

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 $I \in S \cap L_i$, $comp(I) \in L_i$ and is at the right of I
- for all $I \in S \cap R_i$, $comp(I) \in R_i$ and is at the left of I

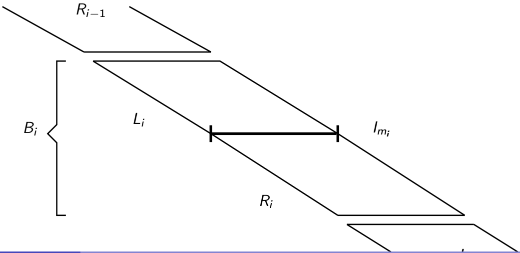
PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

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 $I \in S \cap L_i$, $comp(I) \in L_i$ and is at the right of I
- for all $I \in S \cap R_i$, $comp(I) \in R_i$ and is at the left of I



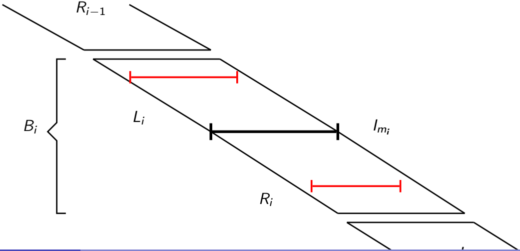
PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

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 $I \in S \cap L_i$, $comp(I) \in L_i$ and is at the right of I
- for all $I \in S \cap R_i$, $comp(I) \in R_i$ and is at the left of I



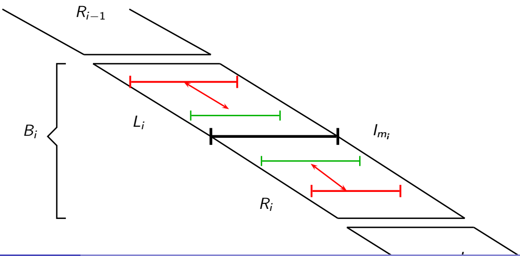
PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

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 $I \in S \cap L_i$, $comp(I) \in L_i$ and is at the right of I
- for all $I \in S \cap R_i$, $comp(I) \in R_i$ and is at the left of I



PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

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 $I \in S \cap L_i$, $comp(I) \in L_i$ and is at the right of I
- for all $I \in S \cap R_i$, $comp(I) \in R_i$ and is at the left of I

Lemma

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

$$E(comp(S \cap B_i)) \leq \rho E(S \cap B_i)$$

Then $comp(S)$ is a ρ-approximation of S.

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.

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 $q_L \rightarrow X_1^L, ..., X_P^L$
- we divide X_R into P consecutive subsets of same size $q_R \rightarrow X_1^R, ..., X_P^R$

Then define the compaction: for any $t \in \{1, ..., P\}$

PTAS in Proper Interval Graphs

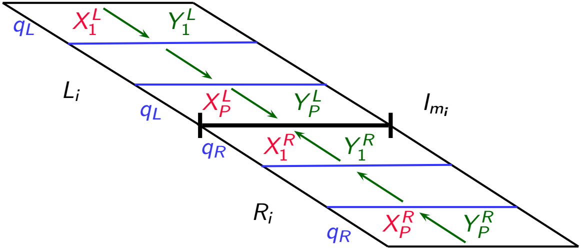
Re-structuration of optimal solutions

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 $q_L \rightarrow X_1^L, ..., X_P^L$
- we divide X_R into P consecutive subsets of same size $q_R \rightarrow X_1^R, ..., X_P^R$

Then define the compaction: for any $t \in \{1, ..., P\}$

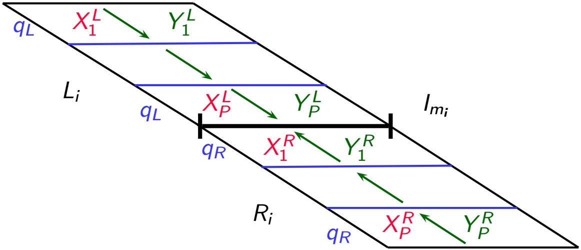
- Y_t^L are the q_L rightmost intervals at the left of the rightmost interval of X_t^L
- Y_t^R are the q_R leftmost intervals at the right of the leftmost interval of X_t^R



PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

What do we need to construct such a solution ?



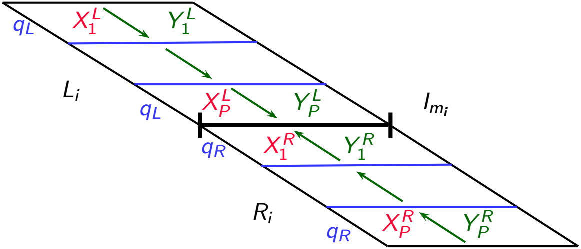
PTAS in Proper Interval Graphs

Re-structuration of optimal solutions

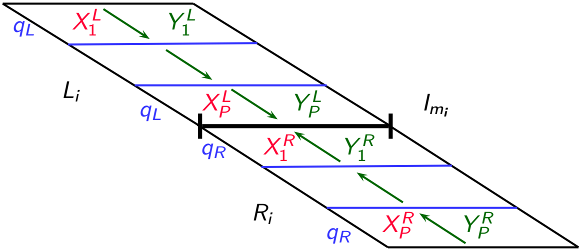
What do we need to construct such a solution ?

- 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 $\{0, ..., n\}$

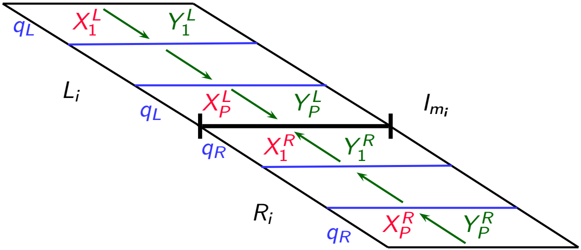


PTAS in Proper Interval Graphs



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

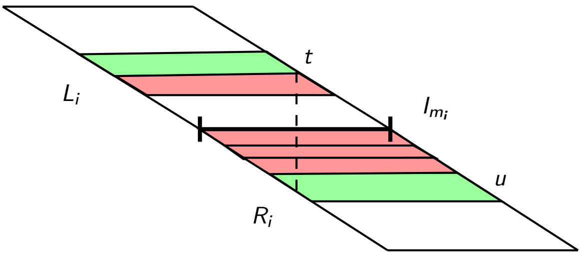
PTAS in Proper Interval Graphs



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

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

PTAS in Proper Interval Graphs

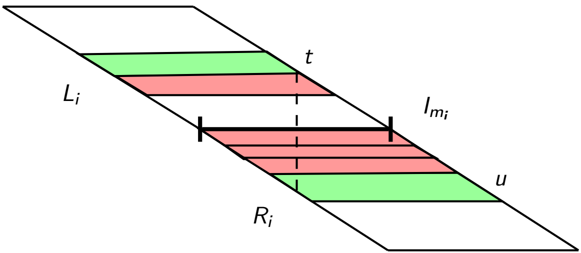


Sketch of proof of the $(1 + \frac{4}{P})$ approximation ratio:

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

But:

PTAS in Proper Interval Graphs



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

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

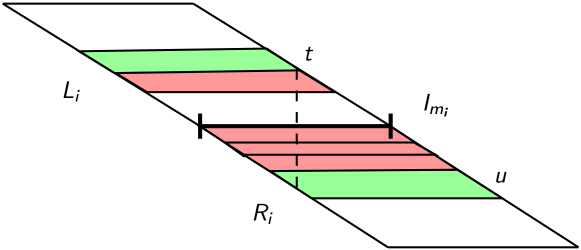
But:

- 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 $\bigcup_{i=1}^{u-1} X_i^R$

PTAS in Proper Interval Graphs



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

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

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 $\frac{SOL}{OPT} \leq 1 + \frac{4}{P}$

PTAS in Proper Interval Graphs

Conclusion:

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

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:
  $k' \leftarrow k$, $C' \leftarrow C^*$, $s \leftarrow$ left endpoint of the leftmost interval

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:
  $k' \leftarrow k$, $C' \leftarrow C^*$, $s \leftarrow$ left endpoint of the leftmost interval
- 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:
  $k' \leftarrow k$, $C' \leftarrow C^*$, $s \leftarrow$ left endpoint of the leftmost interval
- 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) $|T| \leq k'$ and $E(T) \leq C'$

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:
  $k' \leftarrow k$, $C' \leftarrow C^*$, $s \leftarrow$ left endpoint of the leftmost interval
- 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) $|T| \leq k'$ and $E(T) \leq C'$
- 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:
  $k' \leftarrow k$, $C' \leftarrow C^*$, $s \leftarrow$ left endpoint of the leftmost interval
- 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) $|T| \leq k'$ and $E(T) \leq C'$
- 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:
  $k' \leftarrow k$, $C' \leftarrow C^*$, $s \leftarrow$ left endpoint of the leftmost interval
- 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) $|T| \leq k'$ and $E(T) \leq C'$
- 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) $|T| \leq k'$ and $E(T) \leq C'$

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

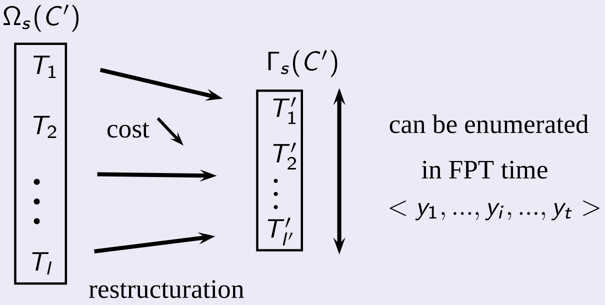
Lemma

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) $|T| \leq k'$ and $E(T) \leq C'$

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

Lemma



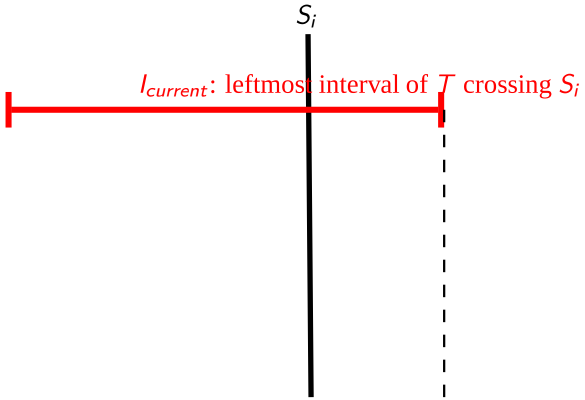
FPT Algorithm in Interval Graphs

Restructuration of a connected component T.

We process intervals of T using a cursor S_i.

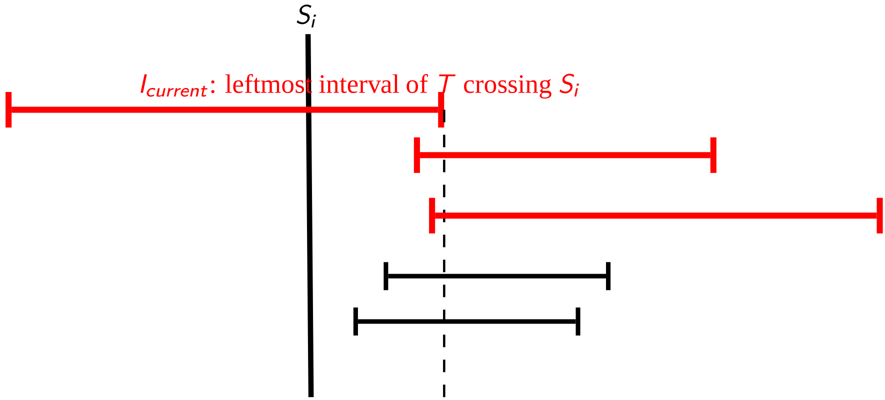
FPT Algorithm in Interval Graphs

Restructuration of a connected component T.
We process intervals of T using a cursor S_i.



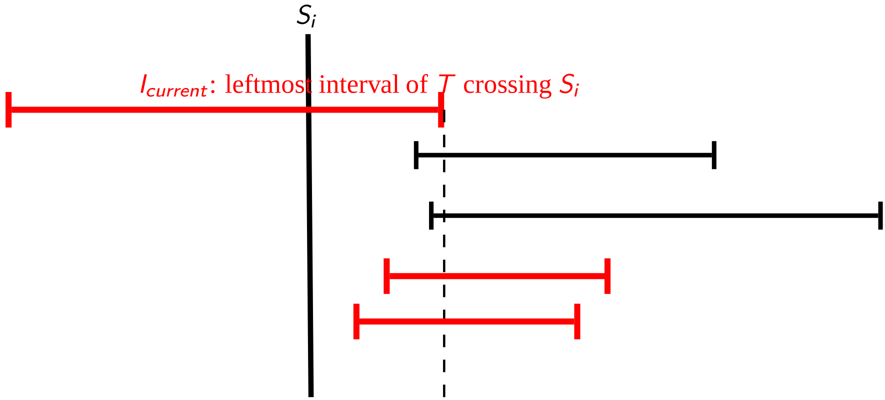
FPT Algorithm in Interval Graphs

Restructuration of a connected component T.
We process intervals of T using a cursor S_i.



FPT Algorithm in Interval Graphs

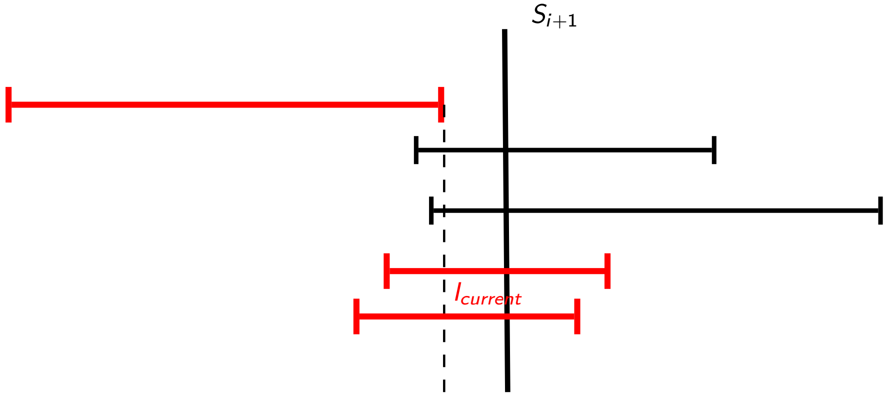
Restructuration of a connected component T.
We process intervals of T using a cursor S_i.



FPT Algorithm in Interval Graphs

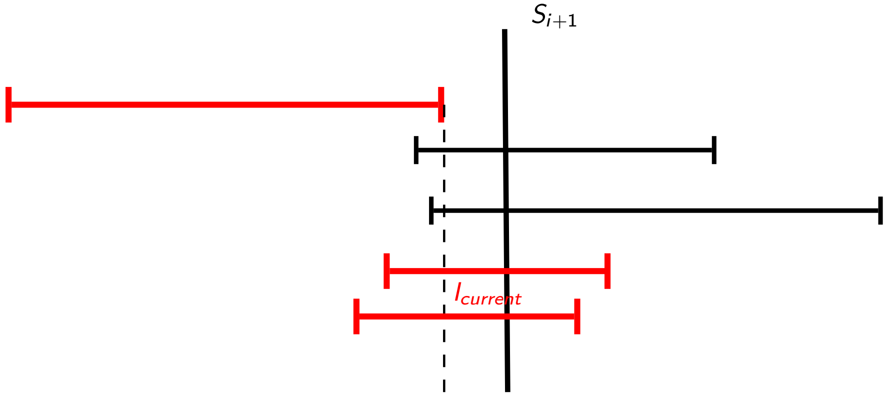
Restructuration of a connected component T.

We process intervals of T using a cursor S_i.



FPT Algorithm in Interval Graphs

Restructuration of a connected component T.
We process intervals of T using a cursor S_i.



$\Rightarrow$ we only have to "guess" the number y_i of intervals overlapping $I_{current}$

FPT Algorithm in Interval Graphs

Any connected component $T \in \Gamma_s(C)$ 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

FPT Algorithm in Interval Graphs

Any connected component $T \in \Gamma_s(C)$ 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 $\Rightarrow y_i \leq \sqrt{2C} + 2$

FPT Algorithm in Interval Graphs

Any connected component $T \in \Gamma_s(C)$ 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 $\Rightarrow y_i \leq \sqrt{2C} + 2$
- each $T \in \Gamma_s(C)$ is a connected component, and each S_i crosses a different interval of T

FPT Algorithm in Interval Graphs

Any connected component $T \in \Gamma_s(C)$ 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
  $\Rightarrow y_i \leq \sqrt{2C} + 2$
- each $T \in \Gamma_s(C)$ is a connected component, and each S_i crosses a different interval of T
  $\Rightarrow t \leq C + 1$

FPT Algorithm in Interval Graphs

Any connected component $T \in \Gamma_s(C)$ 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
  $\Rightarrow y_i \leq \sqrt{2C} + 2$
- each $T \in \Gamma_s(C)$ is a connected component, and each S_i crosses a different interval of T
  $\Rightarrow t \leq C + 1$

Thus:

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

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

FPT Algorithm in Interval Graphs

Any connected component $T \in \Gamma_s(C)$ 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
  $\Rightarrow y_i \leq \sqrt{2C} + 2$
- each $T \in \Gamma_s(C)$ is a connected component, and each S_i crosses a different interval of T
  $\Rightarrow t \leq C + 1$

Thus:

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

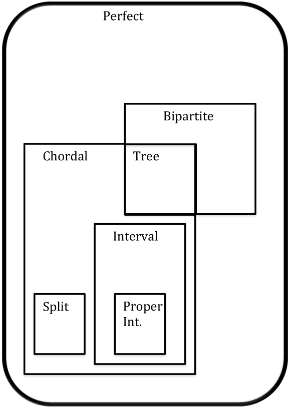
and each step of the dynamic programming runs in *FPT* time.

Theorem

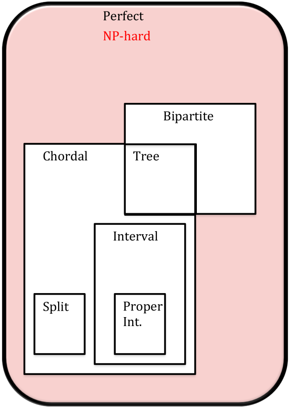
Sparsest k-Subgraph in Interval Graphs is *FPT* parameterized by the cost of the solution.

Contents

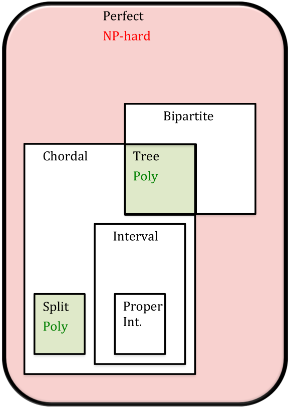
Open problems and Future Work



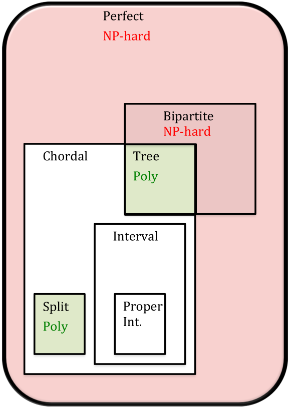
Open problems and Future Work



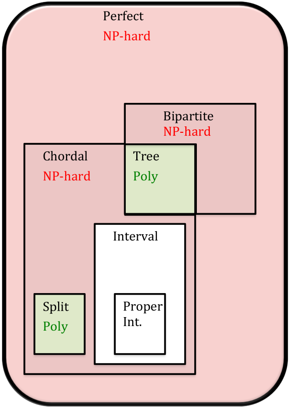
Open problems and Future Work



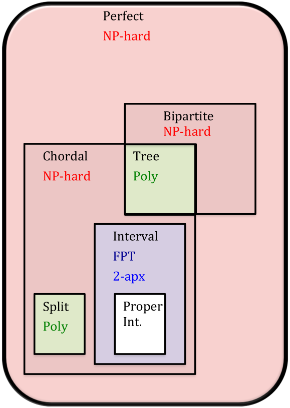
Open problems and Future Work



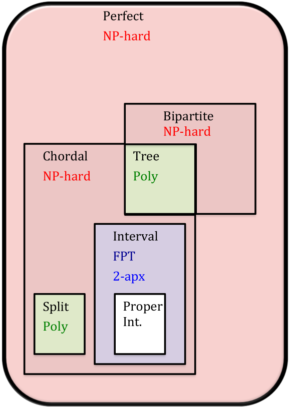
Open problems and Future Work



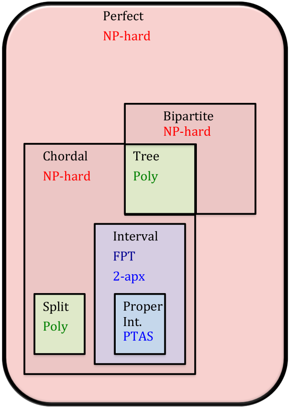
Open problems and Future Work



Open problems and Future Work



Open problems and Future Work



- *NP*-h/Poly in (Proper) Interval graphs ?
- extend FPT and/or approximation results to Chordal graphs ?
- polynomial kernel in Interval graphs ?

Thank you for your attention!