Cardinality constrained subgraph problems

Rémi Watrigant

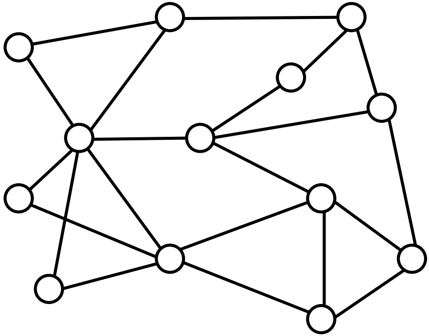
Postdoc @ Hong Kong Polytechnic University

results picked from two works:
one with É. Bonnet, M. Bougeret, V. Paschos, F. Sikora
one with M. Bougeret, N. Bousquet, R. Giroudeau

Workshop on Parameterized Complexity
Chofu, Tokyo, Japan
Feb 28 - Mar 1, 2015

Introduction

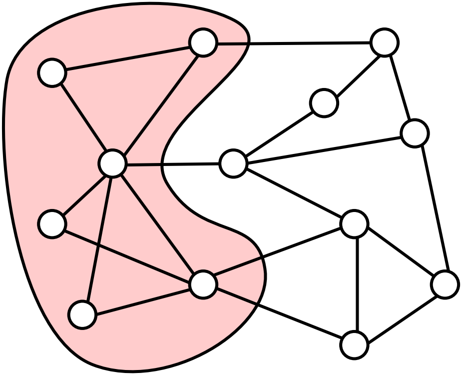
Cardinality constrained subgraph problem:
Input: a graph $G = (V, E)$, $k \in \mathbb{N}$
Output: a set S of k vertices



Introduction

Cardinality constrained subgraph problem:
Input: a graph $G = (V, E)$, $k \in \mathbb{N}$
Output: a set S of k vertices

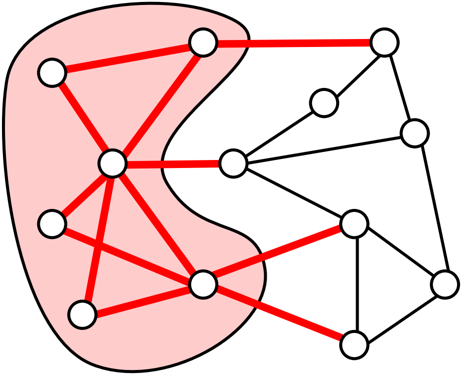
$k = 6$



Introduction

Cardinality constrained subgraph problem:
Input: a graph $G = (V, E)$, $k \in \mathbb{N}$
Output: a set S of k vertices

$k = 6$

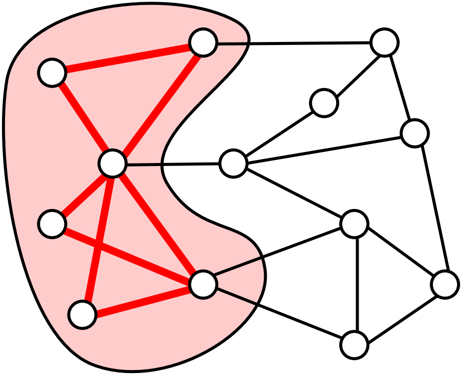


Maximum k-Cover
Minimum k-Cover

Introduction

Cardinality constrained subgraph problem:
Input: a graph $G = (V, E)$, $k \in \mathbb{N}$
Output: a set S of k vertices

$k = 6$

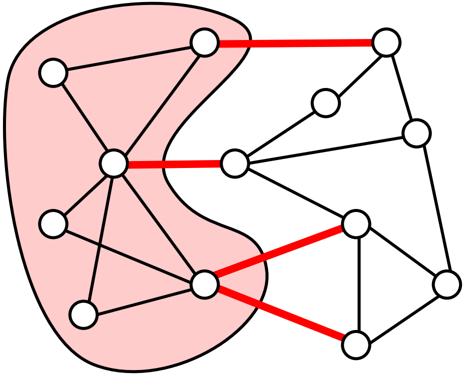


Densest k-Subgraph
Sparsest k-Subgraph

Introduction

Cardinality constrained subgraph problem:
Input: a graph $G = (V, E)$, $k \in \mathbb{N}$
Output: a set S of k vertices

$k = 6$

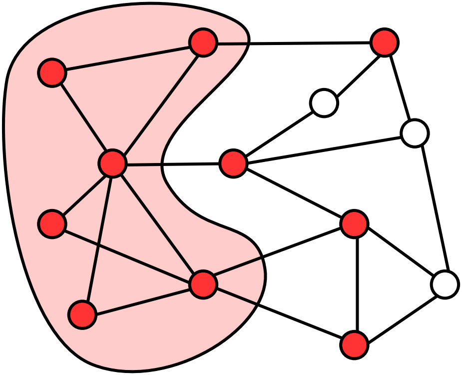


Maximum $(k, n - k)$-Cut
Minimum $(k, n - k)$-Cut

Introduction

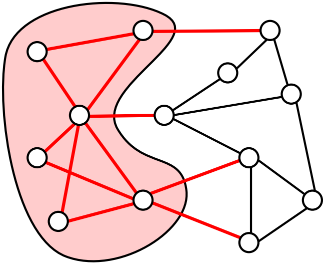
Cardinality constrained subgraph problem:
Input: a graph $G = (V, E)$, $k \in \mathbb{N}$
Output: a set S of k vertices

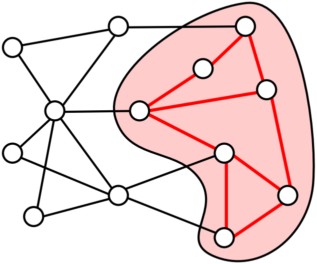
$k = 6$



Maximum k-Dominating Set
Minimum k-Dominating Set

Introduction





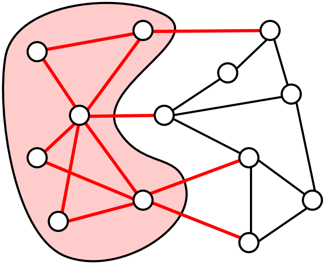
Maximum k-Cover

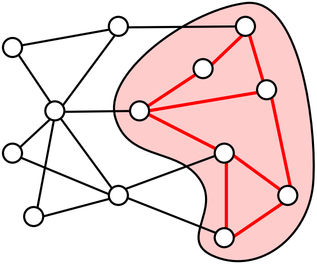
Densest k-Subgraph

Minimum k-Cover

Sparsest k-Subgraph

Introduction





Maximum k-Cover

complement

$G \leftrightarrow \bar{G}$

Minimum k-Cover

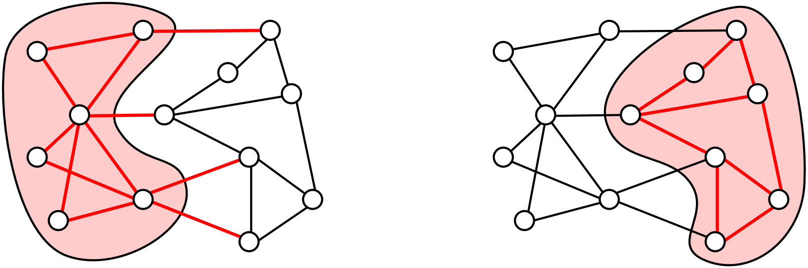
Densest k-Subgraph

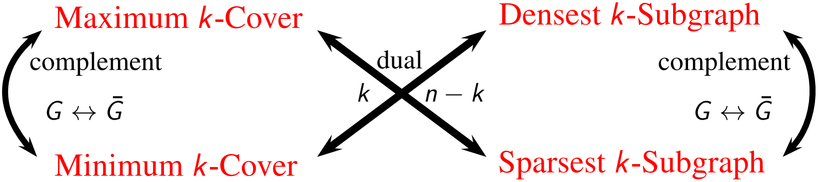
complement

$G \leftrightarrow \bar{G}$

Sparsest k-Subgraph

Introduction





Known results/summary

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General				
Bipartite				
Chordal				

Known results/summary

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	NP-h (generalization of Clique, Independent Set)			
Bipartite				
Chordal				

Known results/summary

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	NP-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08]			
Bipartite				
Chordal				

Known results/summary

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	NP-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08] FPT(std. param.) [Blaser'03]			
Bipartite				
Chordal				

Known results/summary

	Covering		Inducing	
Graphs	Max	Min	Max	Min
	Max-k-Cover	Min-k-Cover	Densest	Sparsest
General	*NP*-h (generalization of Clique, Independent Set)			
	$W[1]$-h (k) [Cai '08]			
	FPT (std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14]			
Chordal				

Known results/summary

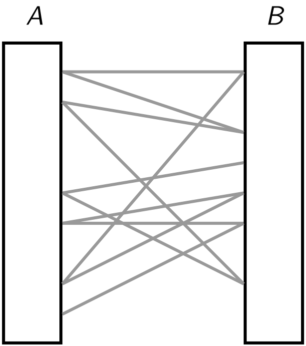
Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08]			
	FPT (std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14]		*NP*-h [CP'84] $W[1]$-h [CP'84]	
Chordal				

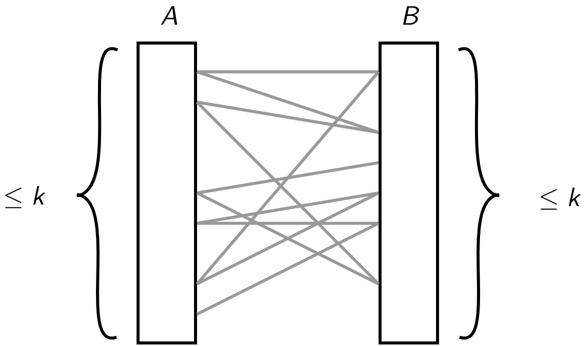
Known results/summary

Graphs	Covering: Max (Max-k-Cover)	Covering: Min (Min-k-Cover)	Inducing: Max (Densest)	Inducing: Min (Sparsest)
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08] *FPT* (std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14]	*NP*-h (dual)	*NP*-h [CP'84] $W[1]$-h [CP'84]	*NP*-h (dual)
Chordal				

Known results/summary

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08] *FPT*(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14]	*NP*-h (dual)	*NP*-h [CP'84] $W[1]$-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal				


A
B



A
B
$\leq k$
$\leq k$

Known results/summary

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08]			
	FPT(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14]	*NP*-h (dual)	*NP*-h [CP'84] $W[1]$-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal				

Known results/summary

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08] *FPT*(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) $W[1]$-h [BBPSW'14]	*NP*-h [CP'84] $W[1]$-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal				

Known results/summary

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08]			
	FPT(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) $W[1]$-h [BBPSW'14]	*NP*-h [CP'84] $W[1]$-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal		*NP*-h (dual)	*NP*-h [CP'84] *FPT*(k) [trivial]	

Densest k-Subgraph in chordal graphs

- if G has a clique of size k (polynomial)

Densest k-Subgraph in chordal graphs

- if G has a clique of size k (polynomial)
  $\Rightarrow$ optimal solution

- otherwise:

Densest k-Subgraph in chordal graphs

- if G has a clique of size k (polynomial)
  $\Rightarrow$ optimal solution
- otherwise: treewidth $\leq k$
  $\Rightarrow$ classical dynamic programming gives $O^*(2^k)$ algorithm

Known results/summary

Graphs	Covering: Max Max-k-Cover	Covering: Min Min-k-Cover	Inducing: Max Densest	Inducing: Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) *W*[1]-h (k) [Cai '08] *FPT* (std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) *W*[1]-h [BBPSW'14]	*NP*-h [CP'84] *W*[1]-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal		*NP*-h (dual)	*NP*-h [CP'84] *FPT*(k) [trivial]	

Known results/summary

Graphs	Covering: Max Max-k-Cover	Covering: Min Min-k-Cover	Inducing: Max Densest	Inducing: Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08] *FPT*(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) $W[1]$-h [BBPSW'14]	*NP*-h [CP'84] $W[1]$-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal	*NP*-h (dual)	*NP*-h (dual)	*NP*-h [CP'84] *FPT*(k) [trivial]	*NP*-h [BBGW'14] *FPT*(k) [BBBGW'14]

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k-1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

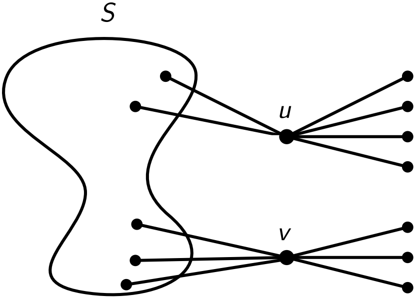
FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k - 1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Proof:



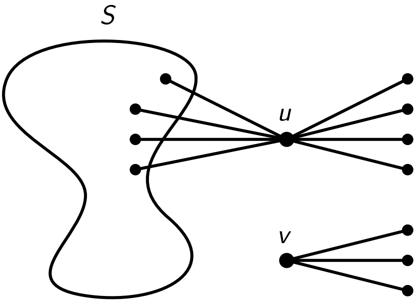
FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k - 1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Proof:



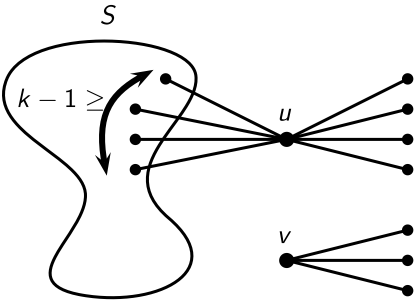
FPT algorithm for Max k-Cover in bipartite graphs

<u>Input:</u> $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k - 1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Proof:



FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k - 1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k-1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Proof:

$$\begin{array}{l} d(v_1) = \Delta \\ d(v_2) \\ \vdots \\ d(v_i) \\ \vdots \\ d(v_k) \\ \vdots \\ d(v_j) \\ \vdots \\ d(v_n) \end{array}$$

- sort vertices by non-increasing degrees

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k-1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ ($\Delta =$ max. degree of $G[V']$)

Proof:

$$d(v_1) = \Delta$$
$$d(v_2)$$
$$\vdots$$
$$d(v_i)$$
$$\vdots$$
$$d(v_k)$$
$$\vdots$$
$$d(v_j)$$
$$\vdots$$
$$d(v_n)$$

- sort vertices by non-increasing degrees
- if $d(v_j) + k \leq d(v_k)$ then $v_j \notin opt$

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k - 1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Proof:

$d(v_1) = \Delta$
$d(v_2)$
$\vdots$
$d(v_i)$
$\vdots$
$d(v_k)$
$\vdots$
$d(v_j)$
$\vdots$
$d(v_n)$

- sort vertices by non-increasing degrees
- if $d(v_j) + k \leq d(v_k)$ then $v_j \notin opt$
  Proof: suppose $v_j \in opt$

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k - 1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Proof:

$d(v_1) = \Delta$
$d(v_2)$
$\vdots$
$d(v_i)$
$\vdots$
$d(v_k)$
$\vdots$
$d(v_j)$
$\vdots$
$d(v_n)$

- sort vertices by non-increasing degrees
- if $d(v_j) + k \leq d(v_k)$ then $v_j \notin opt$
  Proof: suppose $v_j \in opt$
  then $v_i \notin opt$ for some $i \in \{1, \cdots, k\}$

FPT algorithm for Max *k*-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k - 1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Proof:

$d(v_1) = \Delta$
$d(v_2)$
$\vdots$
$d(v_i)$
$\vdots$
$d(v_k)$
$\vdots$
$k \geq$ $d(v_j)$
$\vdots$
$d(v_n)$

- sort vertices by non-increasing degrees
- if $d(v_j) + k \leq d(v_k)$ then $v_j \notin opt$
  Proof: suppose $v_j \in opt$
  then $v_i \notin opt$ for some $i \in \{1, \cdots, k\}$
  but $d(v_i) \geq d(v_k) \geq d(v_j) + k$
  $\Rightarrow$ v_i is better than v_j !

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k - 1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ ($\Delta =$ max. degree of $G[V']$)

Proof:

$d(v_1) = \Delta$
$d(v_2)$
$\vdots$
$d(v_i)$
$\vdots$
$d(v_k)$
$\vdots$
$k \geq$ $d(v_j)$
$\vdots$
$d(v_n)$

- sort vertices by non-increasing degrees
- if $d(v_j) + k \leq d(v_k)$ then $v_j \notin opt$
- if $d(v_k) + k \leq d(v_1)$ then $v_1 \in opt$

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k - 1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Proof:

$d(v_1) = \Delta$
$d(v_2)$
$\vdots$
$d(v_i)$
$\vdots$
$d(v_k)$
$\vdots$
$k \geq$ $d(v_j)$
$\vdots$
$d(v_n)$

- sort vertices by non-increasing degrees
- if $d(v_j) + k \leq d(v_k)$ then $v_j \notin opt$
- if $d(v_k) + k \leq d(v_1)$ then $v_1 \in opt$
  Proof: suppose $v_1 \notin opt$

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k-1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Proof:

$d(v_1) = \Delta$
$d(v_2)$
$\vdots$
$d(v_i)$
$\vdots$
$d(v_k)$
$\vdots$
$k \geq$ $d(v_j)$
$\vdots$
$d(v_n)$

- sort vertices by non-increasing degrees
- if $d(v_j) + k \leq d(v_k)$ then $v_j \notin opt$
- if $d(v_k) + k \leq d(v_1)$ then $v_1 \in opt$
  Proof: suppose $v_1 \notin opt$
  then $v_j \in opt$ for some $j > k$

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 1

Let $S \subseteq V$ be of size $k - 1$, and $u, v \notin S$.
If $d(u) \geq d(v) + k$ then $S \cup \{u\}$ is strictly better than $S \cup \{v\}$

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Proof:

$d(v_1) = \Delta$
$d(v_2)$
$\vdots$
$k \geq$ $d(v_i)$
$\vdots$
$d(v_k)$
$\vdots$
$k \geq$ $d(v_j)$
$\vdots$
$d(v_n)$

- sort vertices by non-increasing degrees
- if $d(v_j) + k \leq d(v_k)$ then $v_j \notin opt$
- if $d(v_k) + k \leq d(v_1)$ then $v_1 \in opt$
  Proof: suppose $v_1 \notin opt$
  then $v_j \in opt$ for some $j > k$
  but $d(v_j) \leq d(v_k) \leq d(v_1) - k$
  $\Rightarrow$ v_1 is better than v_j !

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

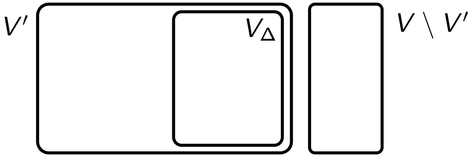
Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Let us consider $V_\Delta \subseteq V'$ the set of vertices of degree Δ

Algorithm:

Restricted graph class:



FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

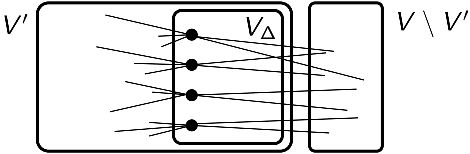
Let us consider $V_\Delta \subseteq V'$ the set of vertices of degree Δ

Algorithm:

- if V_Δ has an independent set of size $\geq k$
  $\Rightarrow$ we are done

Restricted graph class:

- large independent set can be computed efficiently



FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

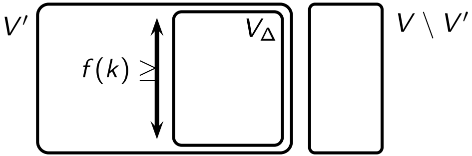
Let us consider $V_\Delta \subseteq V'$ the set of vertices of degree Δ

Algorithm:

- if V_Δ has an independent set of size $\geq k$
  $\Rightarrow$ we are done
- otherwise $|V_\Delta| \leq f(k)$
  $\Rightarrow$ branch on $opt \cap V_\Delta$:

Restricted graph class:

- large independent set can be computed efficiently
- bounded indep. set implies bounded size



FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

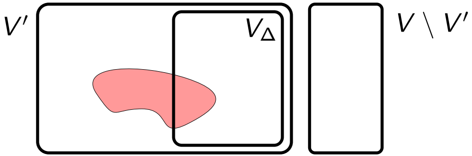
Let us consider $V_\Delta \subseteq V'$ the set of vertices of degree Δ

Algorithm:

- if V_Δ has an independent set of size $\geq k$
  $\Rightarrow$ we are done
- otherwise $|V_\Delta| \leq f(k)$
  $\Rightarrow$ branch on $opt \cap V_\Delta$:
  - if $opt \cap V_\Delta \neq \emptyset \Rightarrow$ guess intersection and decrease k

Restricted graph class:

- large independent set can be computed efficiently
- bounded indep. set implies bounded size



FPT algorithm for Max k-Cover in bipartite graphs

<u>Input:</u> $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

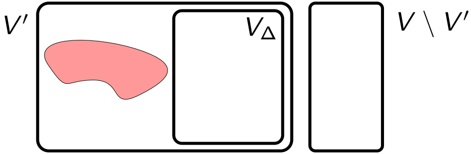
Let us consider $V_\Delta \subseteq V'$ the set of vertices of degree Δ

Algorithm:

- if V_Δ has an independent set of size $\geq k$
  $\Rightarrow$ we are done
- otherwise $|V_\Delta| \leq f(k)$
  $\Rightarrow$ branch on $opt \cap V_\Delta$:
  - if $opt \cap V_\Delta \neq \emptyset \Rightarrow$ guess intersection
    and decrease k
  - if $opt \cap V_\Delta = \emptyset \Rightarrow$ delete V_Δ from V'
    number of $\neq$ degrees decreases

<u>Restricted graph class:</u>

- large independent set can be computed efficiently
- bounded indep. set implies bounded size



FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

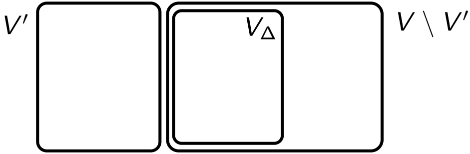
Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Let us consider $V_\Delta \subseteq V'$ the set of vertices of degree Δ

Algorithm:

- if V_Δ has an independent set of size $\geq k$
  $\Rightarrow$ we are done
- otherwise $|V_\Delta| \leq f(k)$
  $\Rightarrow$ branch on $opt \cap V_\Delta$:
  - if $opt \cap V_\Delta \neq \emptyset \Rightarrow$ guess intersection
    and decrease k
  - if $opt \cap V_\Delta = \emptyset \Rightarrow$ delete V_Δ from V'
    number of $\neq$ degrees decreases



Restricted graph class:

- large independent set can be computed efficiently
- bounded indep. set implies bounded size

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Let us consider $V_\Delta \subseteq V'$ the set of vertices of degree Δ

Algorithm:

- if V_Δ has an independent set of size $\geq k$
  $\Rightarrow$ we are done
- otherwise $|V_\Delta| \leq f(k)$
  $\Rightarrow$ branch on $opt \cap V_\Delta$:
  - if $opt \cap V_\Delta \neq \emptyset \Rightarrow$ guess intersection
    and decrease k
  - if $opt \cap V_\Delta = \emptyset \Rightarrow$ delete V_Δ from V'
    number of $\neq$ degrees decreases
- $\Rightarrow$ bounded search tree

Restricted graph class:

- large independent set can be computed efficiently
- bounded indep. set implies bounded size

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Let us consider $V_\Delta \subseteq V'$ the set of vertices of degree Δ

Algorithm:

- if V_Δ has an independent set of size $\geq k$
  $\Rightarrow$ we are done
- otherwise $|V_\Delta| \leq f(k)$
  $\Rightarrow$ branch on $opt \cap V_\Delta$:
  - if $opt \cap V_\Delta \neq \emptyset \Rightarrow$ guess intersection and decrease k
  - if $opt \cap V_\Delta = \emptyset \Rightarrow$ delete V_Δ from V' number of $\neq$ degrees decreases
- $\Rightarrow$ bounded search tree

Restricted graph class:

- large independent set can be computed efficiently
- bounded indep. set implies bounded size

$\Rightarrow$ planar graphs, r-partite (r fixed)

FPT algorithm for Max k-Cover in bipartite graphs

Input: $G = (V, E)$, $k \in \mathbb{N}$, a set $V' \subseteq V$ in which we have to pick the solution

Lemma 2

For all $v \in V'$, $d(v) \in [\Delta - 2k, \Delta]$ (Δ = max. degree of $G[V']$)

Let us consider $V_\Delta \subseteq V'$ the set of vertices of degree Δ

Algorithm:

- if V_Δ has an independent set of size $\geq k$
  $\Rightarrow$ we are done
- otherwise $|V_\Delta| \leq f(k)$
  $\Rightarrow$ branch on $opt \cap V_\Delta$:
  - if $opt \cap V_\Delta \neq \emptyset \Rightarrow$ guess intersection and decrease k
  - if $opt \cap V_\Delta = \emptyset \Rightarrow$ delete V_Δ from V' number of $\neq$ degrees decreases
- $\Rightarrow$ bounded search tree

Restricted graph class:

- large independent set can be computed efficiently
- bounded indep. set implies bounded size

$\Rightarrow$ planar graphs, r-partite (r fixed)

A similar algorithm can be designed for Max-$(k, n-k)$-Cut

Summary

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) *W*[1]-h (k) [Cai '08] *FPT*(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) *W*[1]-h [BBPSW'14]	*NP*-h [CP'84] *W*[1]-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal	*NP*-h (dual)	*NP*-h (dual)	*NP*-h [CP'84] *FPT*(k) [trivial]	*NP*-h [BBGW'14] *FPT*(k) [BBBGW'14]

Summary

Graphs	Covering Max Max-k-Cover	 Min Min-k-Cover	Inducing Max Densest	 Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) *W*[1]-h (k) [Cai '08]			
	FPT(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) *W*[1]-h [BBPSW'14]	*NP*-h [CP'84] *W*[1]-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal	*NP*-h (dual)	*NP*-h (dual)	*NP*-h [CP'84] *FPT*(k) [trivial]	*NP*-h [BBGW'14] *FPT*(k) [BBBGW'14]

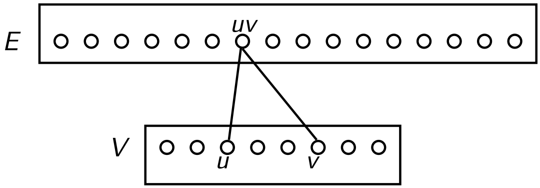
Min-k-Cover is $W[1]$-hard in bipartite graphs

Reduction from k-Clique in regular graphs:

Let $G = (V, E)$, $k \in \mathbb{N}$, and Δ =degree of G

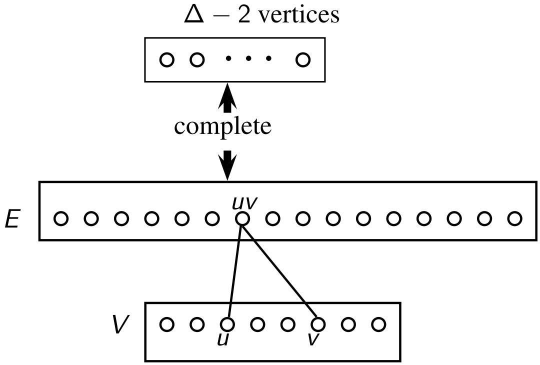
Min-k-Cover is $W[1]$-hard in bipartite graphs

Reduction from k-Clique in regular graphs:
Let $G = (V, E)$, $k \in \mathbb{N}$, and Δ =degree of G



Min-k-Cover is $W[1]$-hard in bipartite graphs

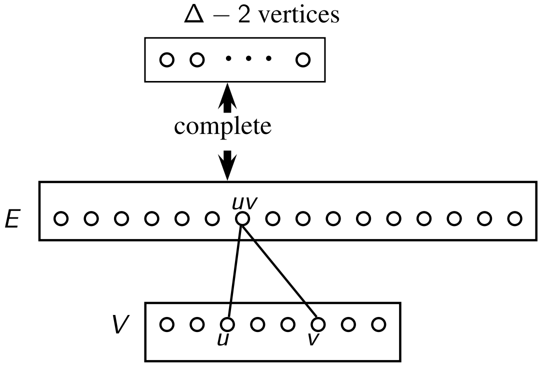
Reduction from k-Clique in regular graphs:
Let $G = (V, E)$, $k \in \mathbb{N}$, and Δ =degree of G



Min-k-Cover is $W[1]$-hard in bipartite graphs

Reduction from k-Clique in regular graphs:
Let $G = (V, E)$, $k \in \mathbb{N}$, and Δ =degree of G

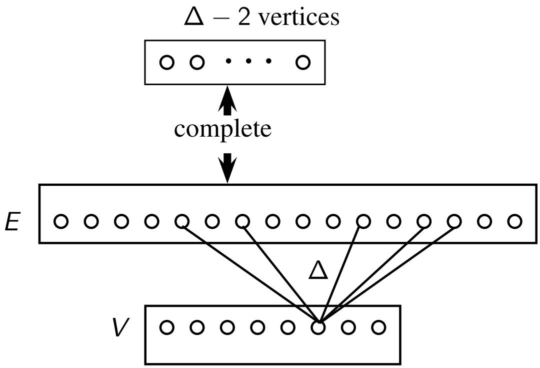
$$k' = k + \binom{k}{2} \text{ vertices covering } k'\Delta - 2\binom{k}{2} \text{ edges ?}$$



Min-k-Cover is $W[1]$-hard in bipartite graphs

Reduction from k-Clique in regular graphs:
Let $G = (V, E)$, $k \in \mathbb{N}$, and Δ =degree of G

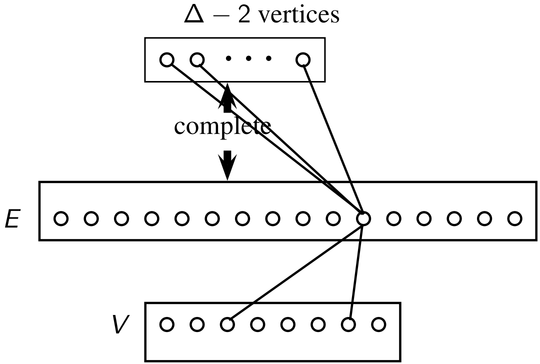
$$k' = k + \binom{k}{2} \text{ vertices covering } k'\Delta - 2\binom{k}{2} \text{ edges ?}$$



Min-k-Cover is $W[1]$-hard in bipartite graphs

Reduction from k-Clique in regular graphs:
Let $G = (V, E)$, $k \in \mathbb{N}$, and Δ =degree of G

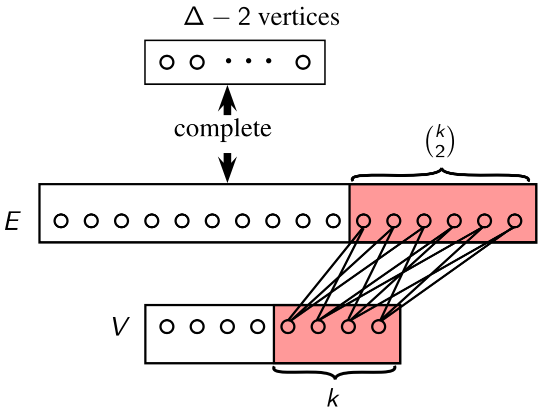
$$k' = k + \binom{k}{2} \text{ vertices covering } k'\Delta - 2\binom{k}{2} \text{ edges ?}$$



Min-k-Cover is $W[1]$-hard in bipartite graphs

Reduction from k-Clique in regular graphs:
Let $G = (V, E)$, $k \in \mathbb{N}$, and $\Delta =$degree of G

$$k' = k + \binom{k}{2} \text{ vertices covering } k'\Delta - 2\binom{k}{2} \text{ edges ?}$$
$$\Leftrightarrow \text{ equivalent to find a } k'\text{-densest subgraph}$$

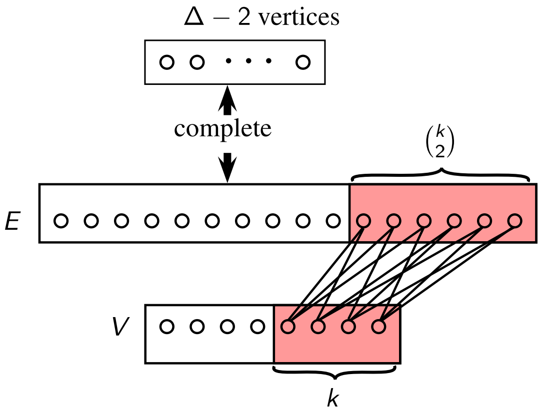


Min-k-Cover is $W[1]$-hard in bipartite graphs

Reduction from k-Clique in regular graphs:
Let $G = (V, E)$, $k \in \mathbb{N}$, and Δ =degree of G

$$k' = k + \binom{k}{2} \text{ vertices covering } k'\Delta - 2\binom{k}{2} \text{ edges ?}$$

$\Leftrightarrow$ equivalent to find a k'-densest subgraph
$\Leftrightarrow$ equivalent to find a k-clique in G



Summary

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08] *FPT*(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) $W[1]$-h [BBPSW'14]	*NP*-h [CP'84] $W[1]$-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal	*NP*-h (dual)	*NP*-h (dual)	*NP*-h [CP'84] *FPT*(k) [trivial]	*NP*-h [BBGW'14] *FPT*(k) [BBBGW'14]

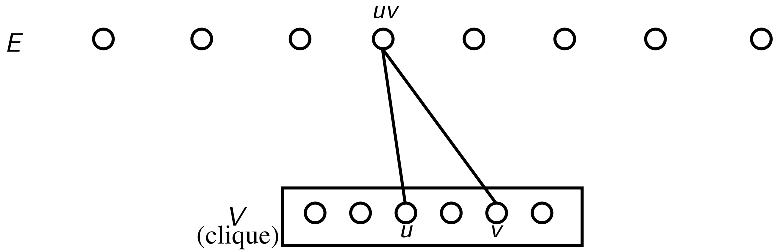
Summary

Graphs	Covering: Max Max-k-Cover	Covering: Min Min-k-Cover	Inducing: Max Densest	Inducing: Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08] *FPT*(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) $W[1]$-h [BBPSW'14]	*NP*-h [CP'84] $W[1]$-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal	*NP*-h (dual)	*NP*-h (dual)	*NP*-h [CP'84] *FPT*(k) [trivial]	*NP*-h [BBGW'14] *FPT*(k) [BBBGW'14]

Densest k-Subgraph is NP-h in chordal graphs [Corneil,Perl,'84]

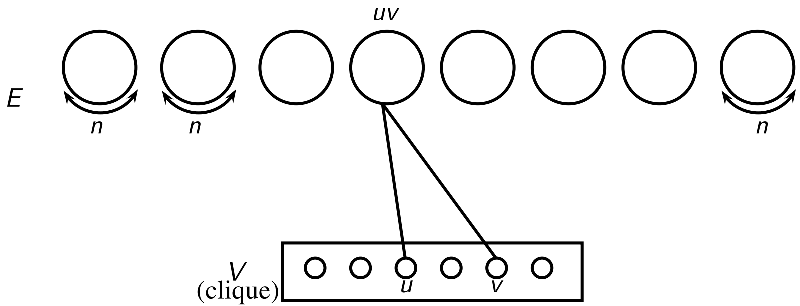
Densest k-Subgraph is NP-h in chordal graphs [Corneil,Perl,'84]

$$k' = k + \quad \binom{k}{2}$$



Densest k-Subgraph is NP-h in chordal graphs [Corneil,Perl,'84]

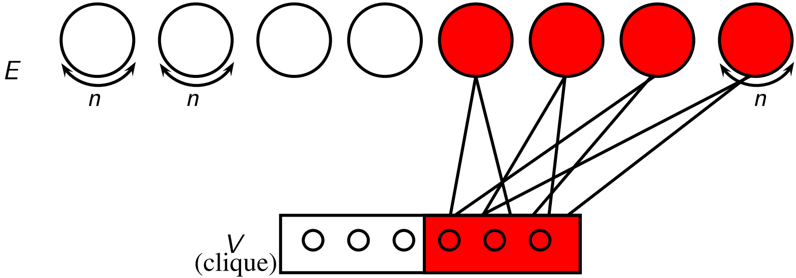
$k' = k + n \binom{k}{2}$



Densest k-Subgraph is NP-h in chordal graphs [Corneil,Perl,'84]

$$k' = k + n \binom{k}{2}$$

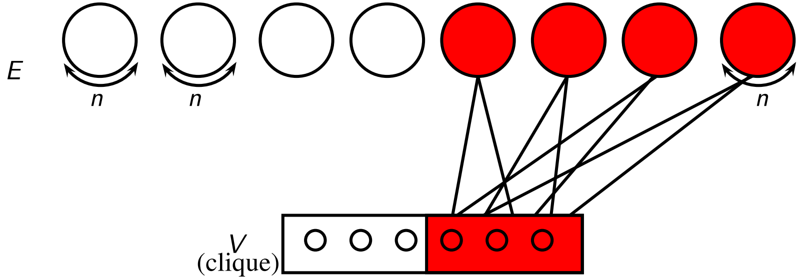
$$inducing \binom{k}{2} + n\binom{k}{2} + 2n\binom{k}{2} \text{ edges}$$



Densest k-Subgraph is NP-h in chordal graphs [Corneil,Perl,'84]

$k' = k + n \binom{k}{2}$

$inducing \binom{k}{2} + n\binom{k}{2} + 2n\binom{k}{2}$ edges



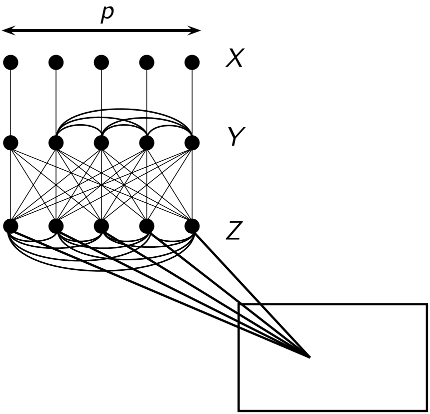
What about Sparsest k-Subgraph now..?

Sparsest *k*-Subgraph is *NP*-hard in chordal graphs [B. G. W., '14]

Idea: gadget

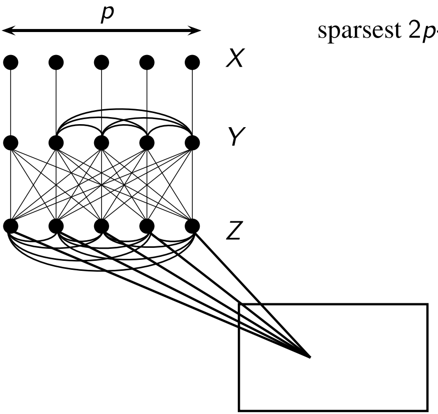
Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

Idea: gadget



Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

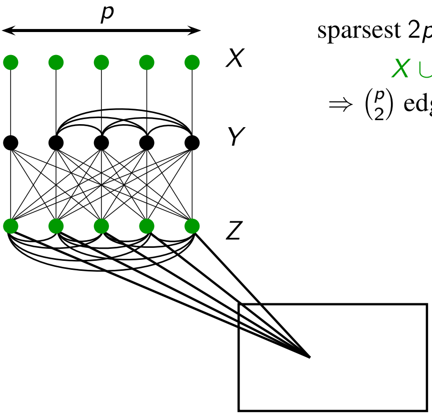
Idea: gadget



sparsest $2p$-subgraph:

Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

Idea: gadget



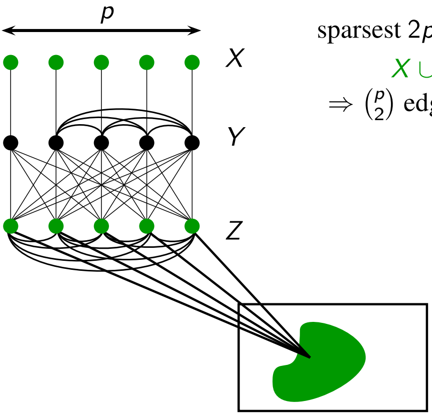
sparsest $2p$-subgraph:

$X \cup Z$

$\Rightarrow \binom{p}{2}$ edges

Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

<u>Idea:</u> gadget



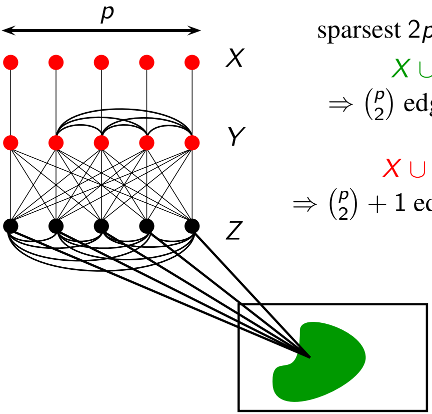
sparsest $2p$-subgraph:

$X \cup Z$

$\Rightarrow \binom{p}{2}$ edges

Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

Idea: gadget



sparsest $2p$-subgraph:

$X \cup Z$

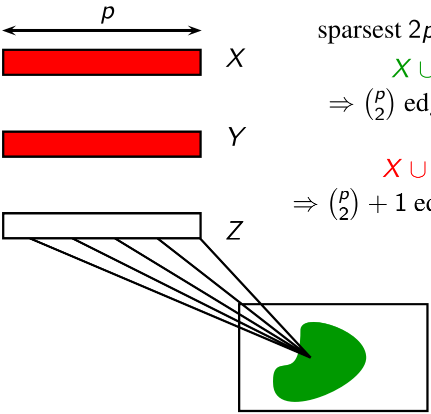
$\Rightarrow \binom{p}{2}$ edges

$X \cup Y$

$\Rightarrow \binom{p}{2} + 1$ edges

Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

Idea: gadget



sparsest $2p$-subgraph:

$X \cup Z$

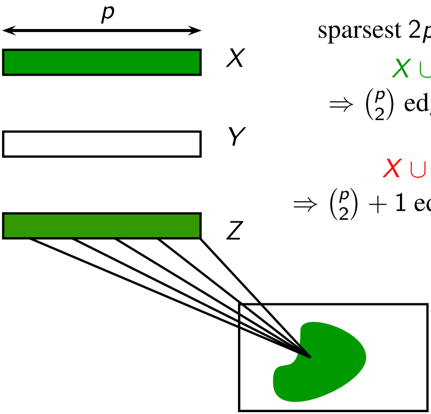
$\Rightarrow \binom{p}{2}$ edges

$X \cup Y$

$\Rightarrow \binom{p}{2} + 1$ edges

Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

Idea: gadget



sparsest $2p$-subgraph:

$X \cup Z$

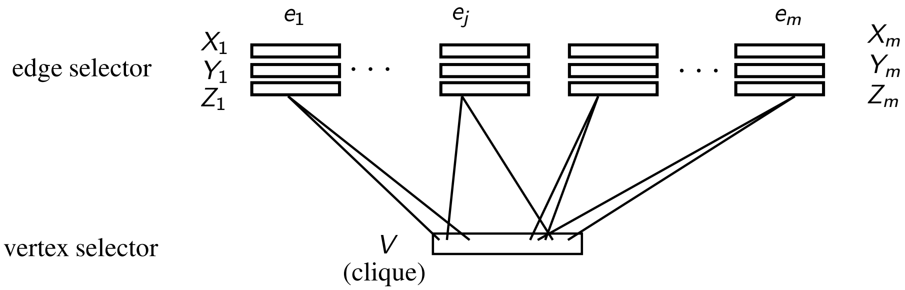
$\Rightarrow \binom{p}{2}$ edges

$X \cup Y$

$\Rightarrow \binom{p}{2} + 1$ edges

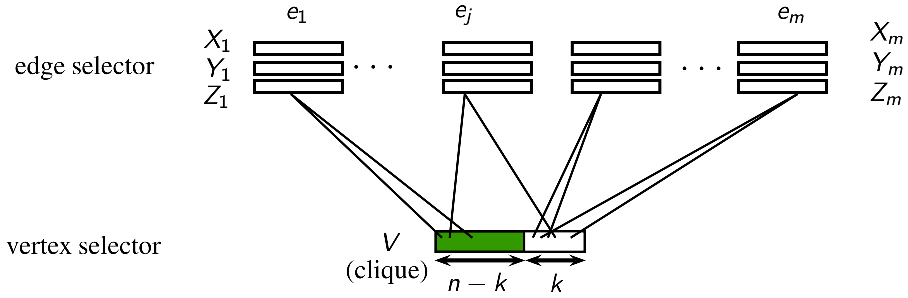
Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

Construction: reduction from k-Clique in general graphs



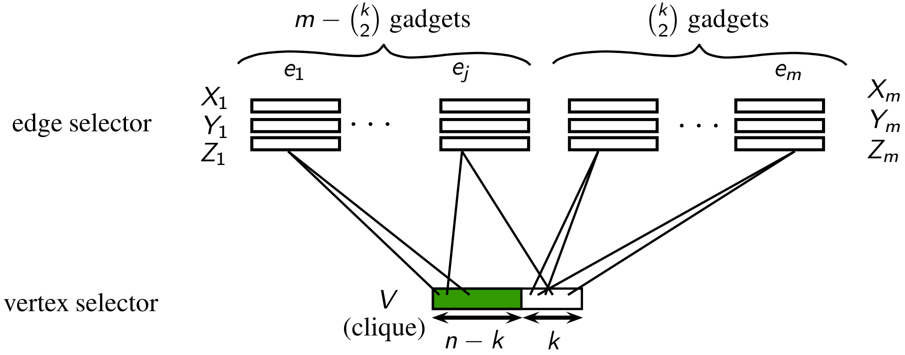
Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

Construction: reduction from k-Clique in general graphs



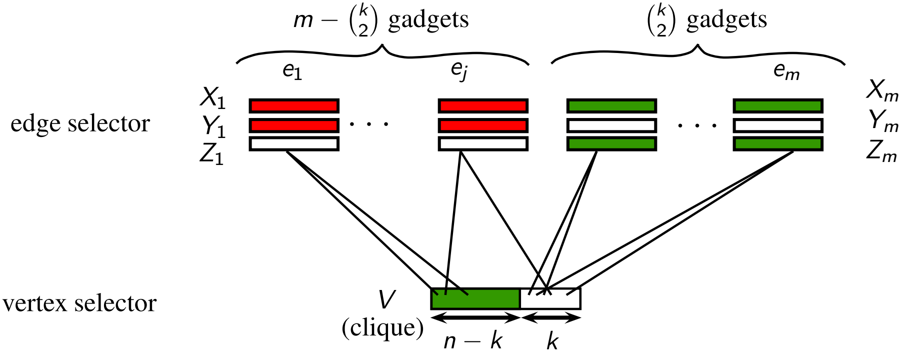
Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

Construction: reduction from k-Clique in general graphs



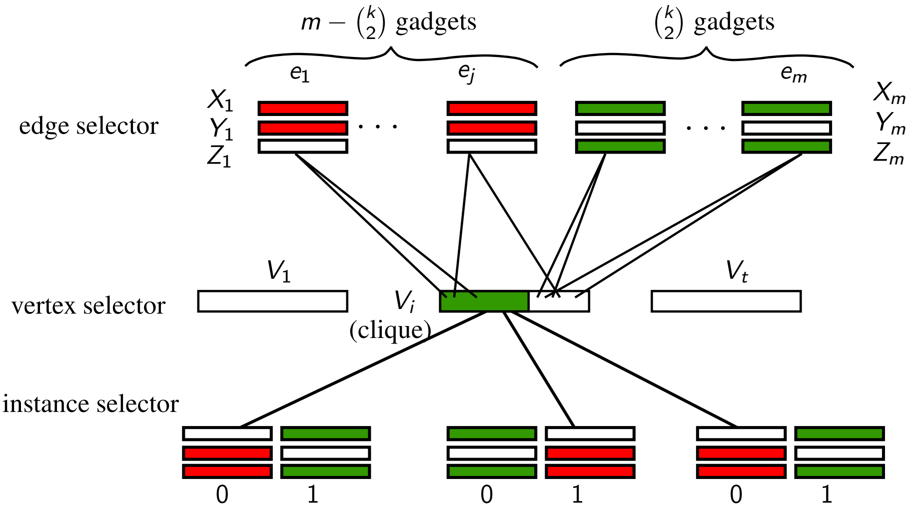
Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

Construction: reduction from k-Clique in general graphs



Sparsest k-Subgraph is NP-hard in chordal graphs [B. G. W., '14]

Extension to cross-composition $\Rightarrow$ no polynomial kernel (unless...)



Summary

Graphs	Covering: Max (Max-k-Cover)	Covering: Min (Min-k-Cover)	Inducing: Max (Densest)	Inducing: Min (Sparsest)
General	*NP*-h (generalization of Clique, Independent Set) *W*[1]-h (k) [Cai '08] *FPT*(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) *W*[1]-h [BBPSW'14]	*NP*-h [CP'84] *W*[1]-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal	*NP*-h (dual)	*NP*-h (dual)	*NP*-h [CP'84] *FPT*(k) [trivial]	*NP*-h [BBGW'14] *FPT*(k) [BBBGW'14]

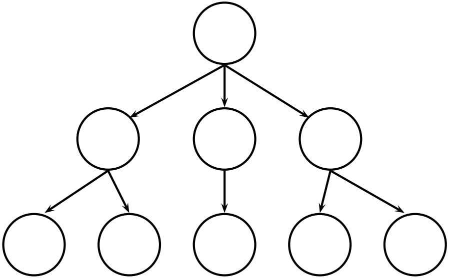
Summary

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08] *FPT*(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) $W[1]$-h [BBPSW'14]	*NP*-h [CP'84] $W[1]$-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal	*NP*-h (dual)	*NP*-h (dual)	*NP*-h [CP'84] *FPT*(k) [trivial]	*NP*-h [BBGW'14] *FPT*(k) [BBBGW'14]

Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

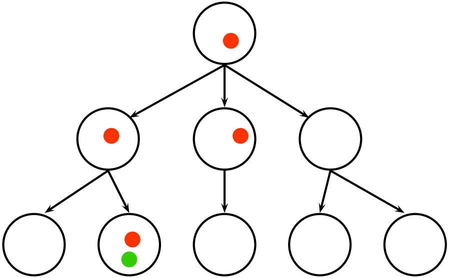
- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T} \leftrightarrow$ maximal cliques of G)



Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T}$ $\leftrightarrow$ maximal cliques of G)
- domination rule: $N($🟢$) \subseteq N($🔴$) \Rightarrow$ 🟢 is cheaper than 🔴 *

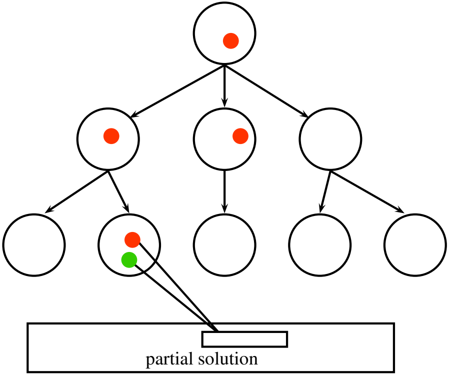


Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T} \leftrightarrow$ maximal cliques of G)
- domination rule: $N(●) \subseteq N(●) \Rightarrow$ ● is cheaper than ●*

*true if ● and ● are modules w.r.t. the partial solution



Invariants:

- nodes of $\mathcal{T}$ are modules w.r.t. S

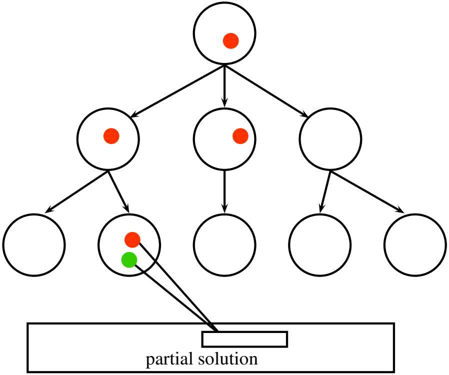
Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T}$ $\leftrightarrow$ maximal cliques of G)
- domination rule: $N($●$) \subseteq N($●$) \Rightarrow$ ● is cheaper than ● *

  *true if ● and ● are modules w.r.t. the partial solution

  $\Rightarrow$ at the beginning: take one simplicial per leaf. But what next ?



Invariants:

- nodes of $\mathcal{T}$ are modules w.r.t. S

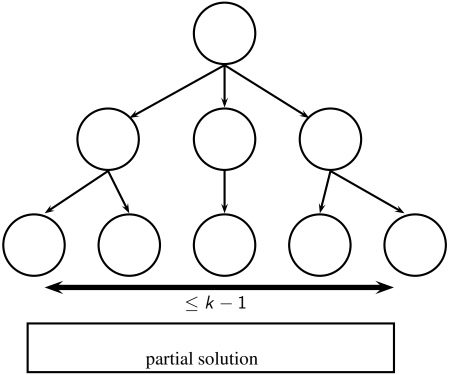
Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T}$ $\leftrightarrow$ maximal cliques of G)
- domination rule: $N($🟢$) \subseteq N($🔴$) \Rightarrow$ 🟢 is cheaper than 🔴 *

  *true if 🟢 and 🔴 are modules w.r.t. the partial solution

  $\Rightarrow$ at the beginning: take one simplicial per leaf. But what next ?
- $\alpha(G) < k$, one simplicial per leaf $\Rightarrow$ at most $k-1$ leaves



Invariants:

- nodes of $\mathcal{T}$ are modules w.r.t. S
- number of leaves $= f(k)$

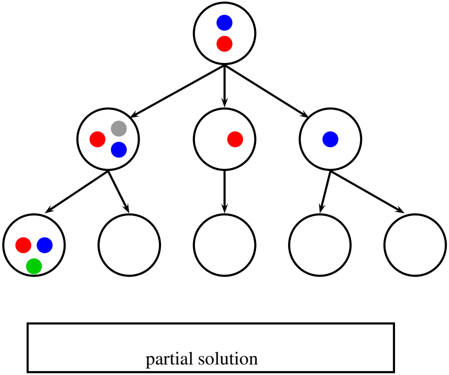
Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T}$ $\leftrightarrow$ maximal cliques of G)
- domination rule: $N($🟢$) \subseteq N($🔴$) \Rightarrow$ 🟢 is cheaper than 🔴 *

  *true if 🟢 and 🔴 are modules w.r.t. the partial solution

  $\Rightarrow$ at the beginning: take one simplicial per leaf. But what next ?
- $\alpha(G) < k$, one simplicial per leaf $\Rightarrow$ at most $k-1$ leaves



Invariants:

- nodes of $\mathcal{T}$ are modules w.r.t. S
- number of leaves $= f(k)$

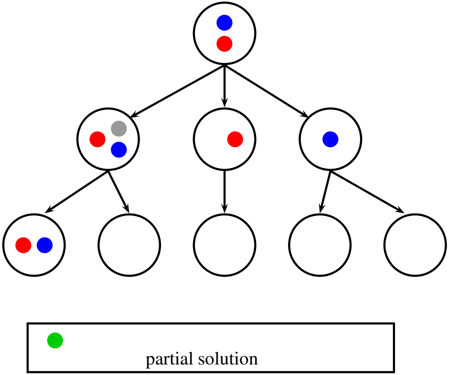
Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T}$ $\leftrightarrow$ maximal cliques of G)
- domination rule: $N(●) \subseteq N(●) \Rightarrow$ ● is cheaper than ● *

  *true if ● and ● are modules w.r.t. the partial solution

  $\Rightarrow$ at the beginning: take one simplicial per leaf. But what next ?
- $\alpha(G) < k$, one simplicial per leaf $\Rightarrow$ at most $k-1$ leaves



Invariants:

- nodes of $\mathcal{T}$ are modules w.r.t. S
- number of leaves $= f(k)$

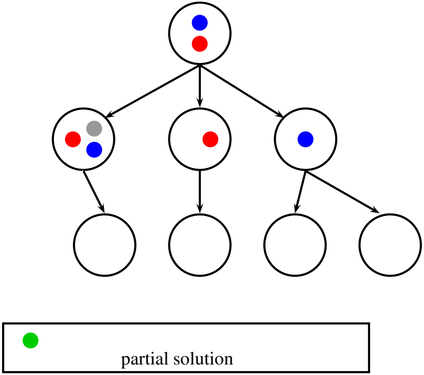
Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T}$ $\leftrightarrow$ maximal cliques of G)
- domination rule: $N(\,)\subseteq N(\,) \Rightarrow$ is cheaper than *

  *true if and are modules w.r.t. the partial solution

  $\Rightarrow$ at the beginning: take one simplicial per leaf. But what next ?
- $\alpha(G) < k$, one simplicial per leaf $\Rightarrow$ at most $k-1$ leaves



Invariants:

- nodes of $\mathcal{T}$ are modules w.r.t. S
- number of leaves $= f(k)$

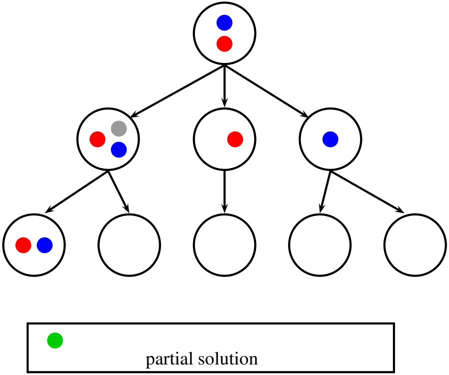
Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T} \leftrightarrow$ ~~maximal~~ cliques of G)
- domination rule: $N($●$) \subseteq N($●$) \Rightarrow$ ● is cheaper than ● *

  *true if ● and ● are modules w.r.t. the partial solution

  $\Rightarrow$ at the beginning: take one simplicial per leaf. But what next ?
- $\alpha(G) < k$, one simplicial per leaf $\Rightarrow$ at most $k-1$ leaves
- add redundancy in the leaves of $\mathcal{T}$



Invariants:

- nodes of $\mathcal{T}$ are modules w.r.t. S
- number of leaves $= f(k)$

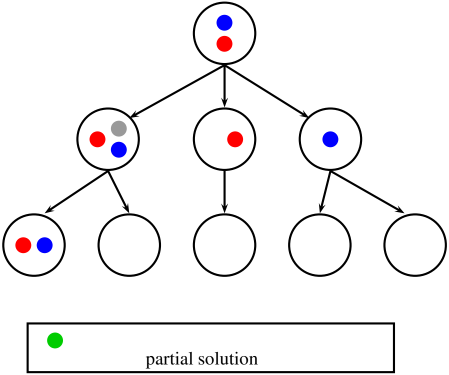
Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T}$ $\leftrightarrow$ ~~maximal~~ cliques of G)
- domination rule: $N($●$) \subseteq N($●$) \Rightarrow$ ● is cheaper than ● *

  *true if ● and ● are modules w.r.t. the partial solution

  $\Rightarrow$ at the beginning: take one simplicial per leaf. But what next ?
- $\alpha(G) < k$, one simplicial per leaf $\Rightarrow$ at most $k-1$ leaves
- add redundancy in the leaves of $\mathcal{T}$



- domination rule may not apply*

  *in leaves!

Invariants:

- nodes of $\mathcal{T}$ are modules w.r.t. S
- number of leaves $= f(k)$

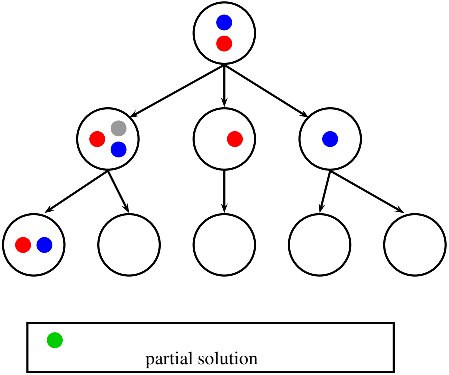
Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T}$ $\leftrightarrow$ ~~maximal~~ cliques of G)
- domination rule: $N($🟢$) \subseteq N($🔴$) \Rightarrow$ 🟢 is cheaper than 🔴*

  *true if 🟢 and 🔴 are modules w.r.t. the partial solution

  $\Rightarrow$ at the beginning: take one simplicial per leaf. But what next ?
- $\alpha(G) < k$, one simplicial per leaf $\Rightarrow$ at most $k-1$ leaves
- add redundancy in the leaves of $\mathcal{T}$



- domination rule may not apply*

  *in leaves!

Invariants:

- nodes of $\mathcal{T}$ are modules w.r.t. S
- number of leaves $= f(k)$
- no redundancy in internal nodes of $\mathcal{T}$

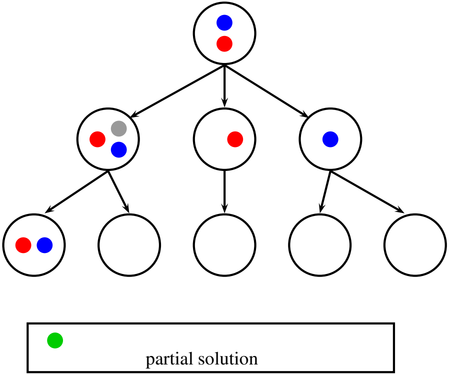
Sparsest k-Subgraph is FPT in chordal graphs [BBBGW'14]

Main arguments:

- tree decomposition $\mathcal{T}$ with (nodes of $\mathcal{T}$ $\leftrightarrow$ ~~maximal~~ cliques of G)
- domination rule: $N($🟢$) \subseteq N($🔴$) \Rightarrow$ 🟢 is cheaper than 🔴 *

  *true if 🟢 and 🔴 are modules w.r.t. the partial solution

  $\Rightarrow$ at the beginning: take one simplicial per leaf. But what next ?
- $\alpha(G) < k$, one simplicial per leaf $\Rightarrow$ at most $k-1$ leaves
- add redundancy in the leaves of $\mathcal{T}$



- domination rule may not apply*

  *in leaves!
- but can apply on predecessors of leaves (here: take ●)

<u>Invariants:</u>

- nodes of $\mathcal{T}$ are modules w.r.t. S
- number of leaves $= f(k)$
- no redundancy in internal nodes of $\mathcal{T}$

Conclusion, open problems

Graphs	Covering		Inducing	
	Max	Min	Max	Min
	Max-k-Cover	Min-k-Cover	Densest	Sparsest
General	*NP*-h (generalization of Clique, Independent Set) *W*[1]-h (k) [Cai '08]			
	FPT(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) *W*[1]-h [BBPSW'14]	*NP*-h [CP'84] *W*[1]-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal	*NP*-h (dual) ?	*NP*-h (dual) ?	*NP*-h [CP'84] *FPT*(k) [trivial]	*NP*-h [BBGW'14] *FPT*(k) [BBBGW'14]

Conclusion, open problems

Graphs	Covering: Max Max-k-Cover	Covering: Min Min-k-Cover	Inducing: Max Densest	Inducing: Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08] *FPT*(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) $W[1]$-h [BBPSW'14]	*NP*-h [CP'84] $W[1]$-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal	*NP*-h (dual) ?	*NP*-h (dual) ?	*NP*-h [CP'84] *FPT*(k) [trivial]	*NP*-h [BBGW'14] *FPT*(k) [BBBGW'14]
(Proper) Interval	?	?	?	?

Conclusion, open problems

Graphs	Covering		Inducing	
	Max Max-k-Cover	Min Min-k-Cover	Max Densest	Min Sparsest
General	*NP*-h (generalization of Clique, Independent Set) $W[1]$-h (k) [Cai '08] *FPT*(std. param.) [Blaser'03]			
Bipartite	*NP*-h [CK'14] *FPT*(k) [BBPSW'14]	*NP*-h (dual) $W[1]$-h [BBPSW'14]	*NP*-h [CP'84] $W[1]$-h [CP'84]	*NP*-h (dual) *FPT*(k)[trivial]
Chordal	*NP*-h (dual) ?	*NP*-h (dual) ?	*NP*-h [CP'84] *FPT*(k) [trivial]	*NP*-h [BBGW'14] *FPT*(k) [BBBGW'14]
(Proper) Interval	?	?	?	?
Planar	*NP*-h (dual)	?	?	*NP*-h (indep. set)

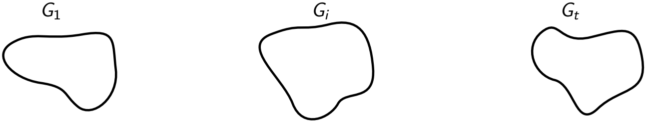
"funny" fact

"funny" fact

- Densest k-Subgraph in chordal graphs is (trivialy FPT and) unlikely to admit a polynomial kernel:
  easy or-composition:

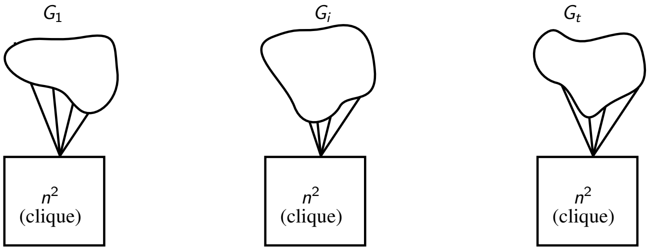
"funny" fact

- Densest k-Subgraph in chordal graphs is (trivialy FPT and) unlikely to admit a polynomial kernel:
  easy or-composition:



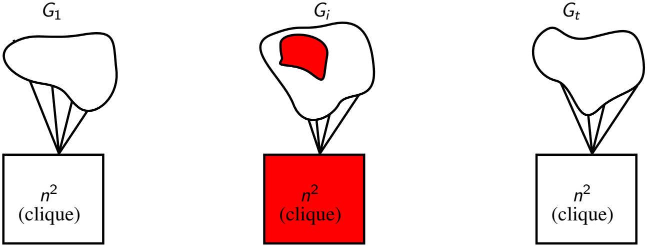
"funny" fact

- Densest k-Subgraph in chordal graphs is (trivialy FPT and) unlikely to admit a polynomial kernel:
  easy or-composition:



"funny" fact

- Densest k-Subgraph in chordal graphs is (trivialy FPT and) unlikely to admit a polynomial kernel:
  easy or-composition:



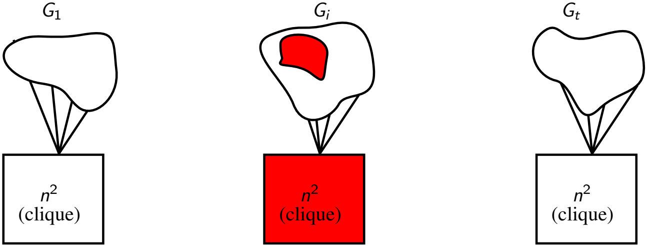
k vertices inducing C edges in G_i

$\Leftrightarrow$

$n^2 + k$ vertices inducing $\binom{n^2}{2} + C + kn^2$ edges ?

"funny" fact

- Densest k-Subgraph in chordal graphs is (trivialy FPT and) unlikely to admit a polynomial kernel:
  easy or-composition:



k vertices inducing C edges in G_i

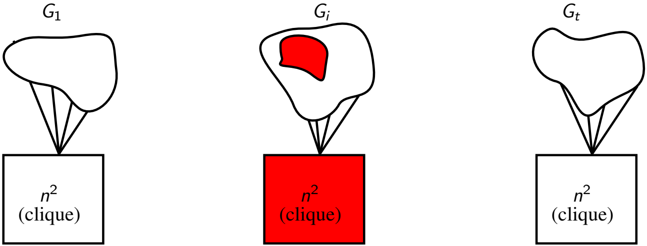
$\Leftrightarrow$

$n^2 + k$ vertices inducing $\binom{n^2}{2} + C + kn^2$ edges ?

- also holds for interval graphs !

"funny" fact

- Densest k-Subgraph in chordal graphs is (trivialy FPT and) unlikely to admit a polynomial kernel:
  easy or-composition:



k vertices inducing C edges in G_i

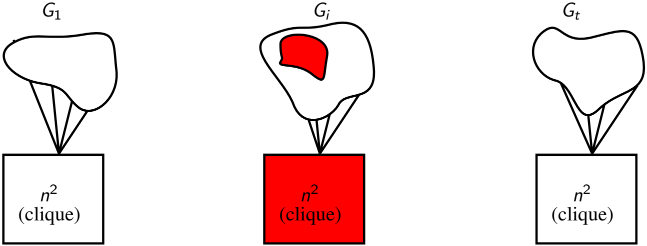
$\Leftrightarrow$

$n^2 + k$ vertices inducing $\binom{n^2}{2} + C + kn^2$ edges ?

- also holds for interval graphs !
- but we don't know if the problem is NP-hard :-(
  so:

"funny" fact

- Densest k-Subgraph in chordal graphs is (trivialy FPT and) unlikely to admit a polynomial kernel:
  easy or-composition:

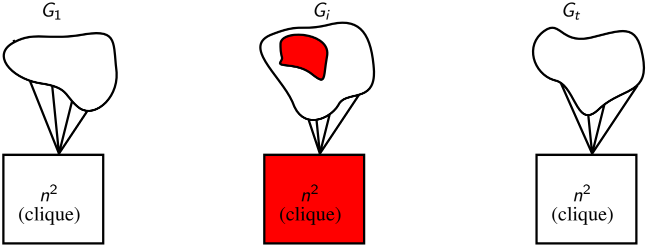


k vertices inducing C edges in G_i
$\Leftrightarrow$
$n^2 + k$ vertices inducing $\binom{n^2}{2} + C + kn^2$ edges ?

- also holds for interval graphs !
- but we don't know if the problem is NP-hard :-(
  so:
  - either it is NP-hard, and has no poly. kernel aswell (unless...) or:

"funny" fact

- Densest k-Subgraph in chordal graphs is (trivialy FPT and) unlikely to admit a polynomial kernel:
  easy or-composition:



k vertices inducing C edges in G_i
$\Leftrightarrow$
$n^2 + k$ vertices inducing $\binom{n^2}{2} + C + kn^2$ edges ?

- also holds for interval graphs !
- but we don't know if the problem is NP-hard :-(
  so:
  - either it is NP-hard, and has no poly. kernel aswell (unless...) or:
  - you believe it is in P ? Show a poly. kernel first !

Merci !