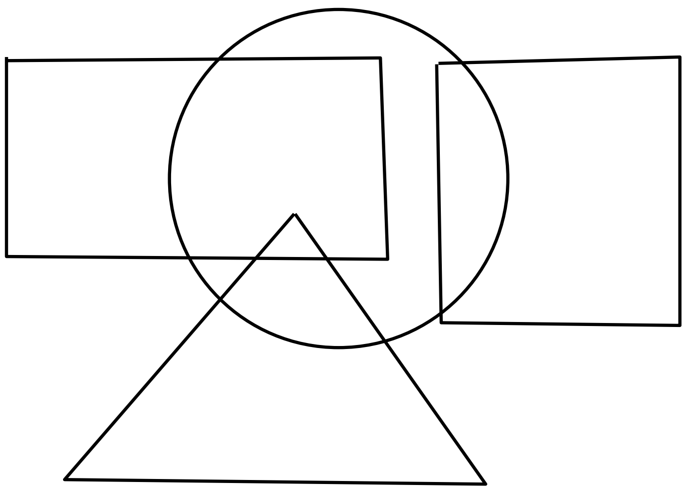
Faster Algorithms for Cycle Hitting Problems on Disk Graphs

Shinwoo An, Kyungjin Cho, and Eunjin Oh
Department of Computer Science and Engineering
Pohang University of Science and Technology (POSTECH)

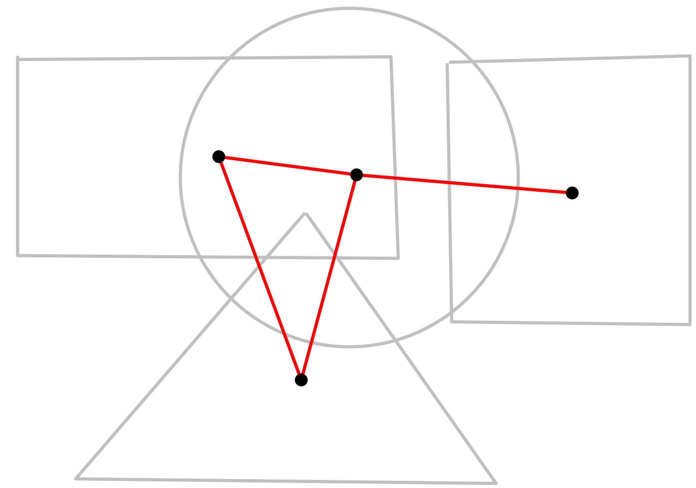
Disk Graphs

Geometric Intersection Graphs: Intersection graphs of geometric objects

F

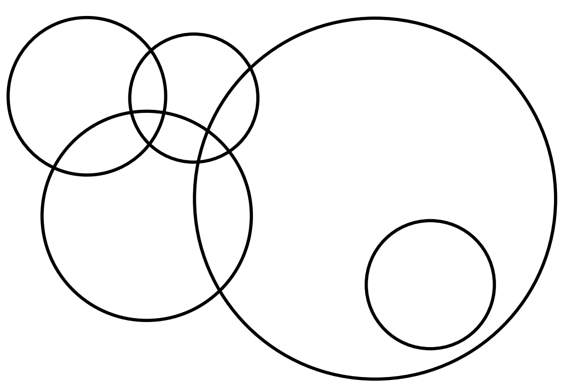


$G[F]$

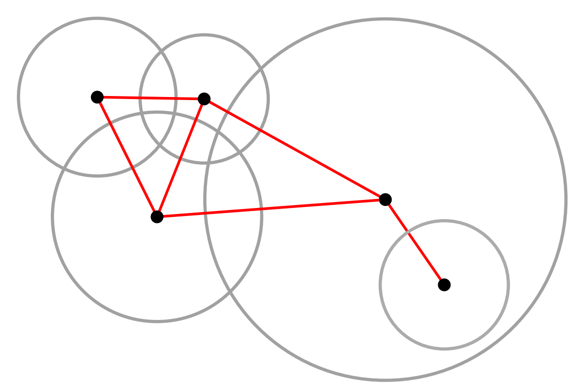


Disk Graphs: Intersection graphs of a set of disks

F

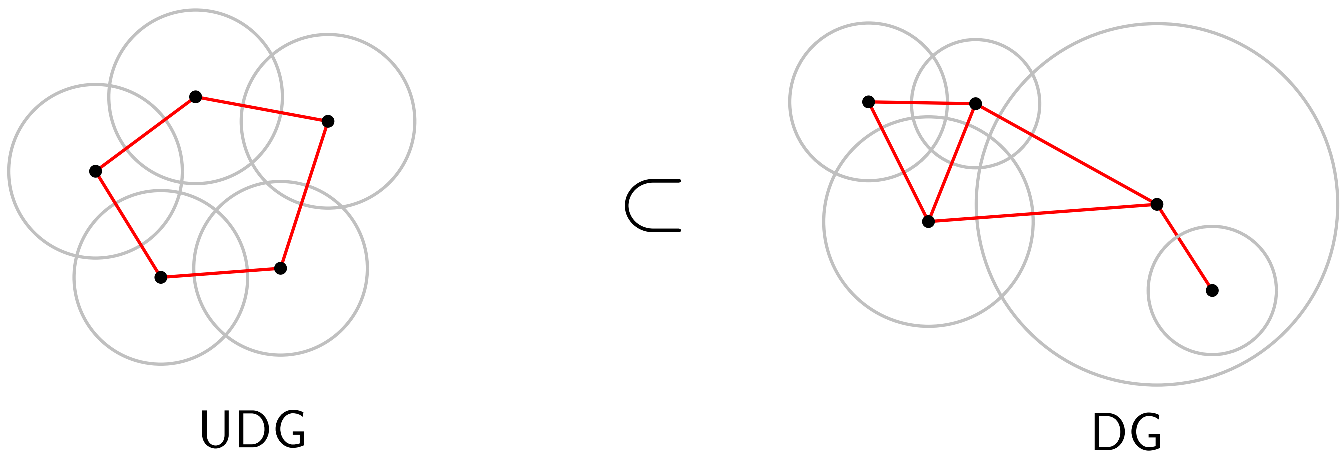


$G[F]$



Disk Graphs

Disk Graphs generalizes unit disk graphs



UDG admits **Subexponential-time algorithms** for Cycle Hitting Problems.

ex) Triangle Hitting Set, Feedback Vertex Set

Question

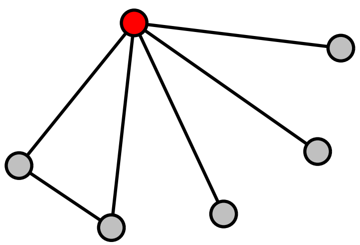
Does a **DG** admit subexponential-time algorithms for those problems?

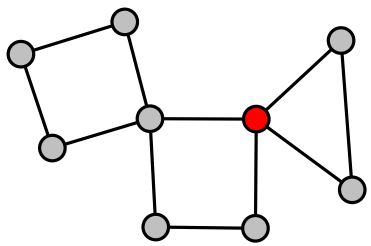
Two Cycle Hitting Problems

$G = (V, E)$: Graph, k : integer, S : solution of size k

Triangle Hitting Set

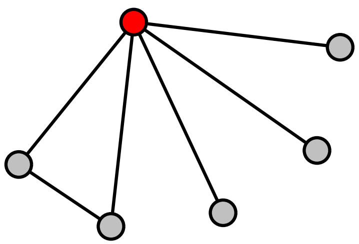
$G - S$ is triangle-free

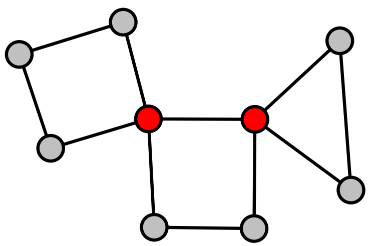




Feedback Vertex Set

$G - S$ is cycle-free



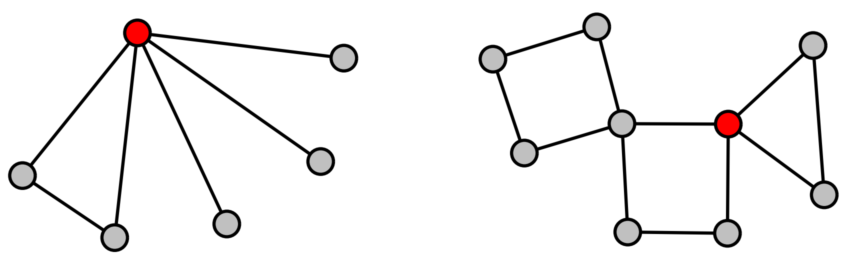


Two Cycle Hitting Problems

$G = (V, E)$: Graph, k : integer, S : solution of size k

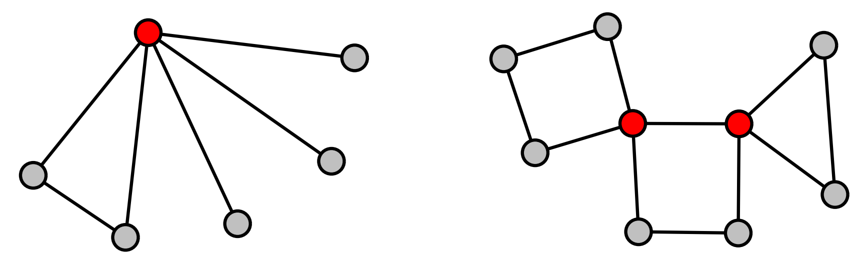
Triangle Hitting Set

$G - S$ is triangle-free



Feedback Vertex Set

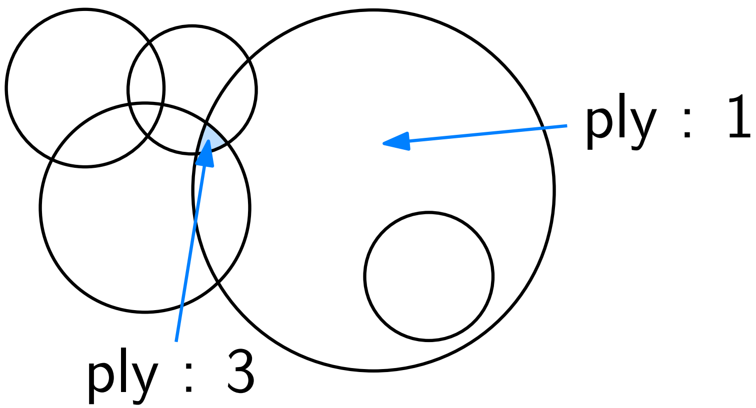
$G - S$ is cycle-free



	THS	FVS
[LPSXZ 23]	$2^{O(k^{9/10}\log k)}n^{O(1)}$	$2^{O(k^{13/14}\log k)}n^{O(1)}$
Ours	$2^{O(k^{4/5}\log k)}n^{O(1)}$	$2^{O(k^{9/10}\log k)}n^{O(1)}$

Overview: Triangle Hitting Set

(G, k): **yes**-insatnce, p : maximum ply

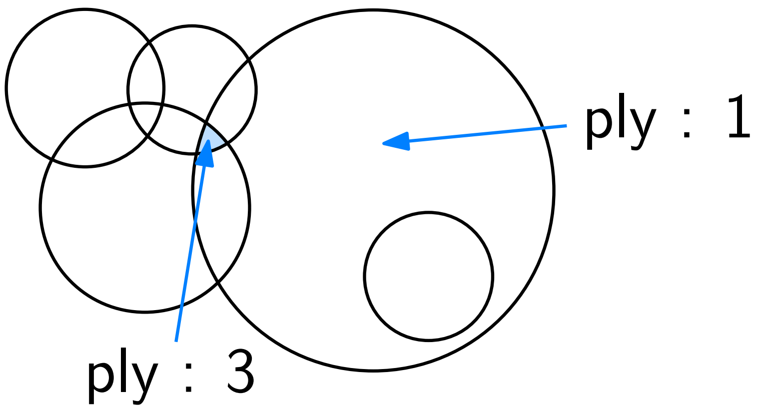


Overview: Triangle Hitting Set

(G, k): **yes**-insatnce, p : maximum ply

Step 1. Branching

- $2^{O((k/p)\log k)}$ instances
- Core of size $O(pk)$

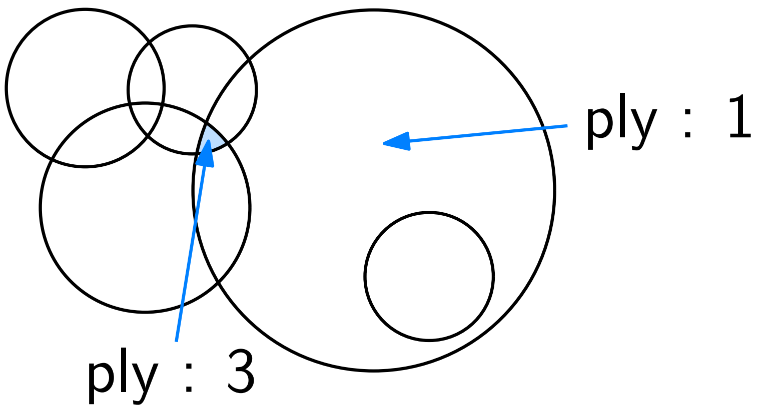


Overview: Triangle Hitting Set

(G, k): **yes**-insatnce, p : maximum ply

Step 1. Branching

- $2^{O((k/p)\log k)}$ instances
- Core of size $O(pk)$

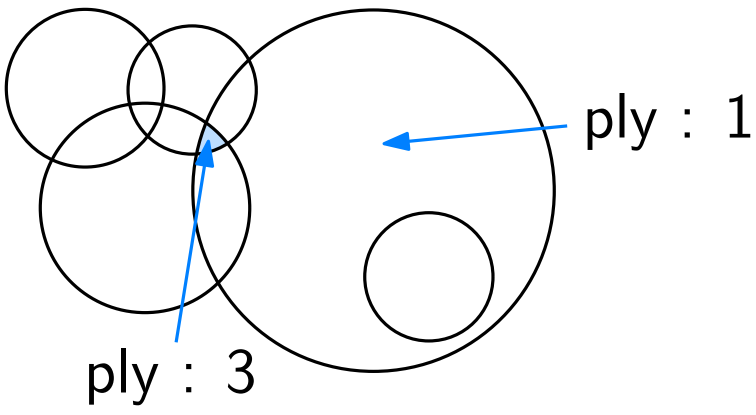


Step 2. Kernelization

- $O(pk)$-size Core $\rightarrow$ $O(pk)$-size kernel

Overview: Triangle Hitting Set

(G, k): **yes**-insatnce, p : maximum ply



Step 1. Branching

- $2^{O((k/p)\log k)}$ instances
- Core of size $O(pk)$

Step 2. Kernelization

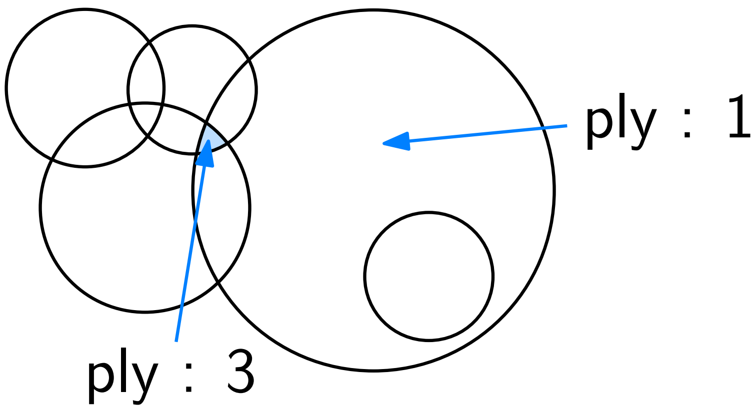
- $O(pk)$-size Core $\rightarrow$ $O(pk)$-size kernel

Step 3. Treewidth Analysis

- $O(pk)$-size kernel $\rightarrow$ $O(p^{1.5}\sqrt{k})$ treewidth

Overview: Triangle Hitting Set

(G, k): **yes**-insatnce, p : maximum ply



Step 1. Branching

- $2^{O((k/p)\log k)}$ instances
- Core of size $O(pk)$

Step 2. Kernelization

- $O(pk)$-size Core $\rightarrow O(pk)$-size kernel

Step 3. Treewidth Analysis

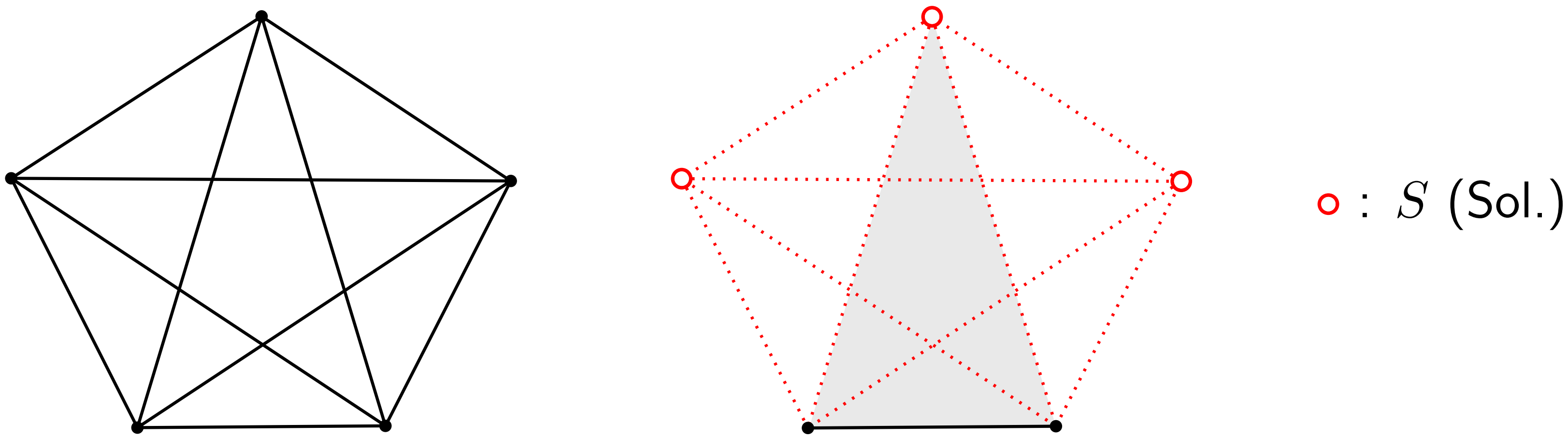
- $O(pk)$-size kernel $\rightarrow O(p^{1.5}\sqrt{k})$ treewidth

Step 4. Dynamic Programming

- Standard DP

Branching: Phase 1

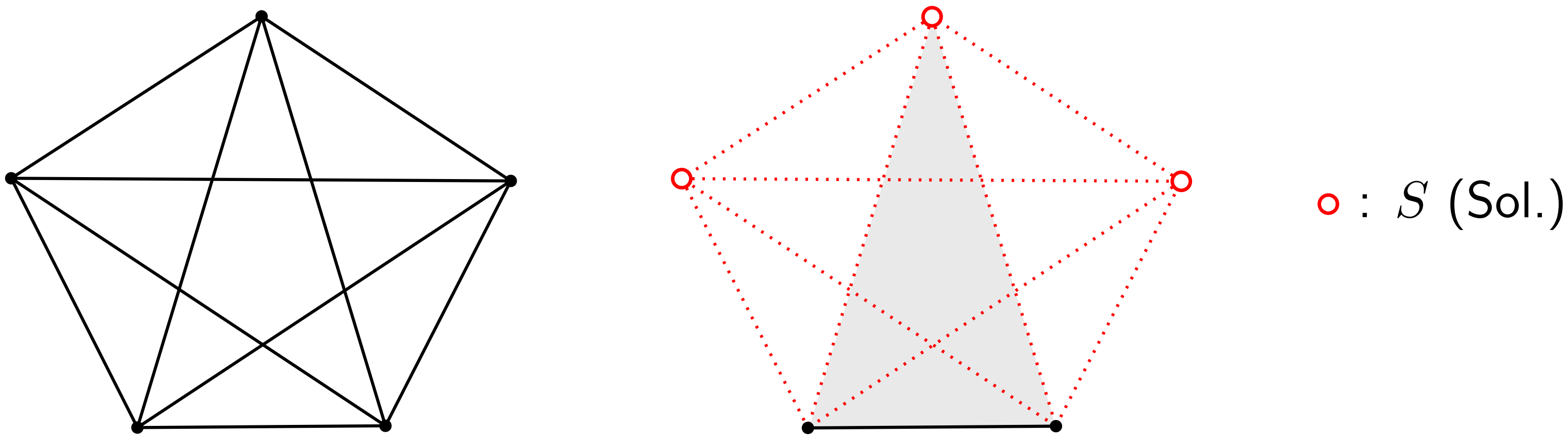
p: **maximum ply**, k: **Solution size**



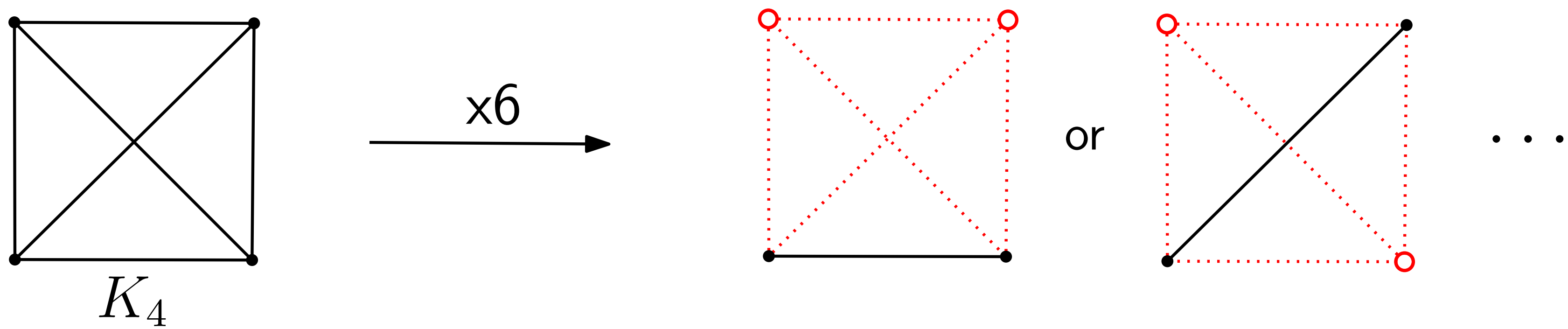
Clique of size t **contains** $t-2$ **solutions**.

Branching: Phase 1

p: **maximum ply**, k: **Solution size**



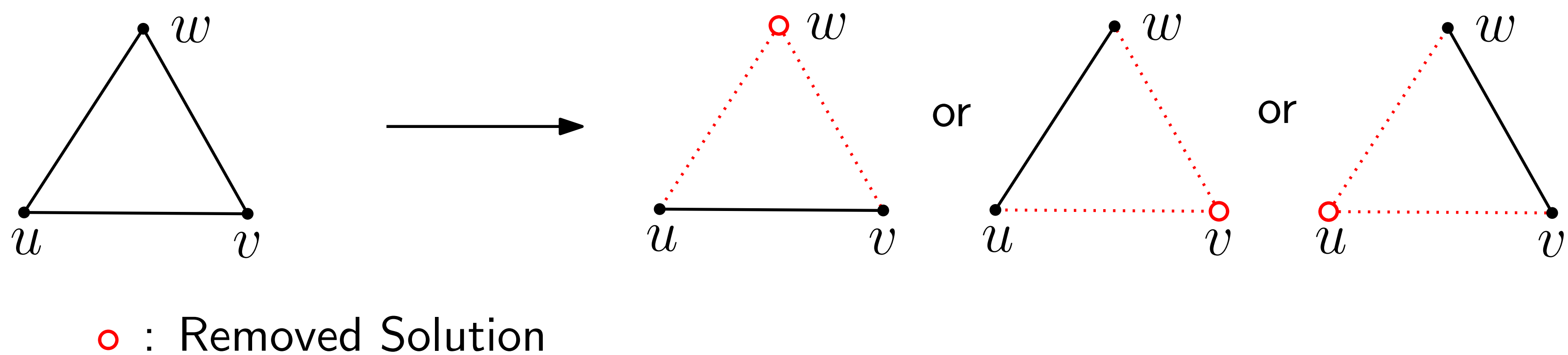
Clique of size t contains $t-2$ solutions.



- Branch all large cliques ($\geq p$)
- $2^{O((k/p)\log k)}$ instances (subexponential)

Branching: Phase 2

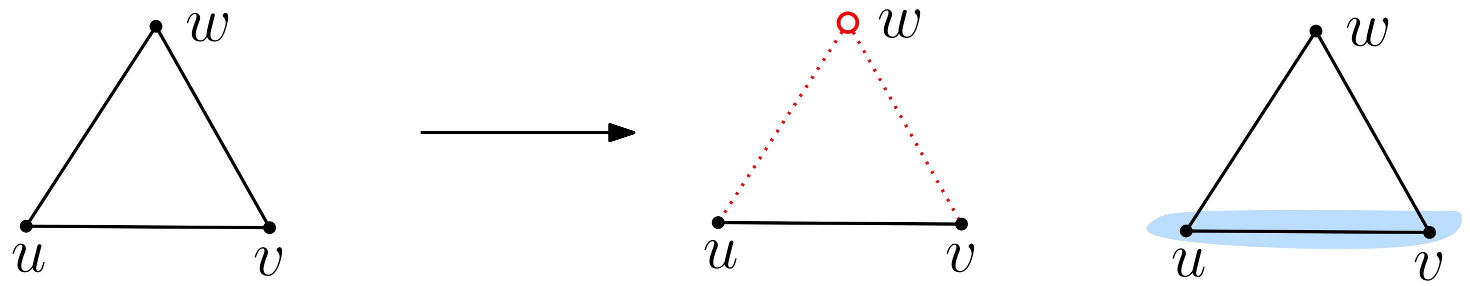
p: **maximum ply**, k: **Solution size**



$\rightarrow 2^{O(k)}$ instances..

Branching: Phase 2

p: **maximum ply**, k: **Solution size**



: Removed : Mark

Alternatives: **Remove** w or **Mark** uv

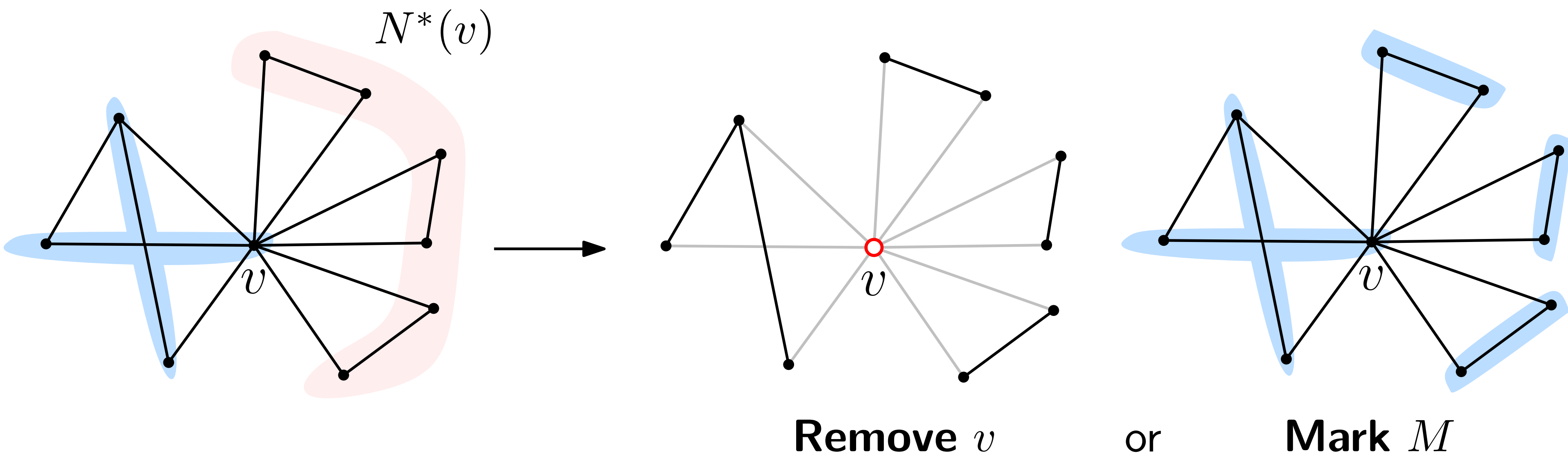
$|\text{Remove}| + |\text{Mark}| \leq k$

- Still $2^{O(k)}$ instances

Branching: Phase 2

p: **maximum ply**, k: **Solution size**

Branch on **large-size matching**



$N^*(v)$: neighbors of v, not adjacent to Mark

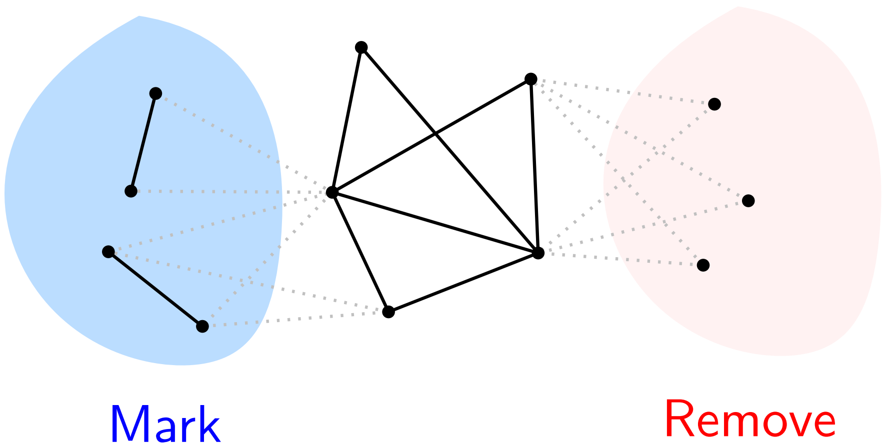
M: maximum matching of $N^*(v)$

Branch if $|M| > p$. (large size matching)

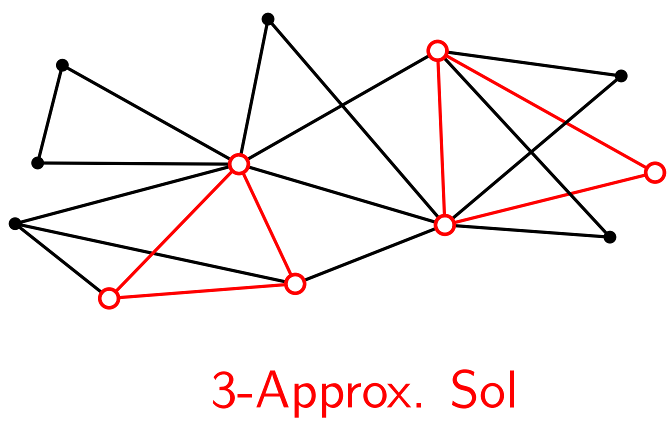
$2^{O((k/p)\log k)}$ instances (subexponential)

Core from the Branching

p : **maximum ply** and **maximum matching size**



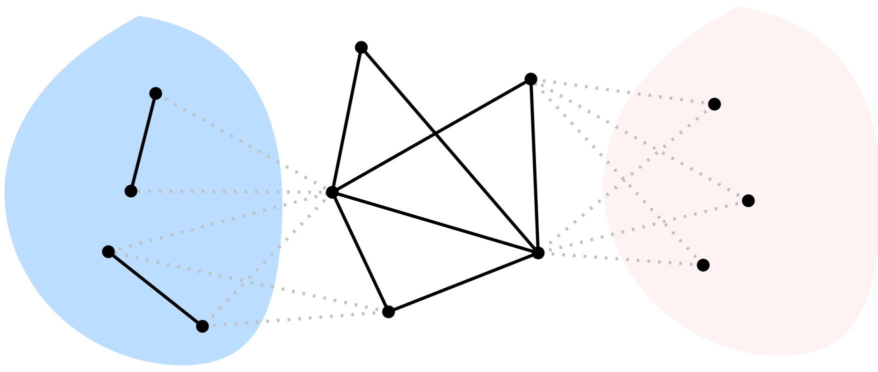
Compute 3-approx. solution by **greedy approach**



- Polynomai time

Core from the Branching

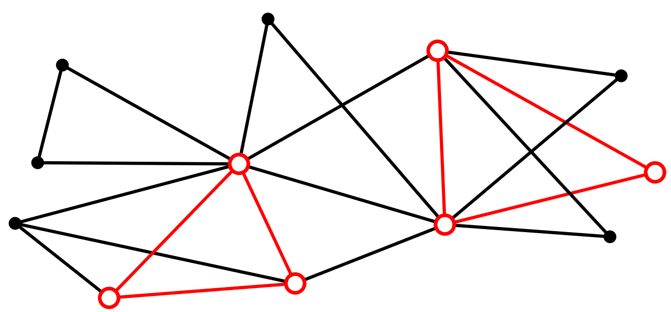
p : **maximum ply** and **maximum matching size**



Mark

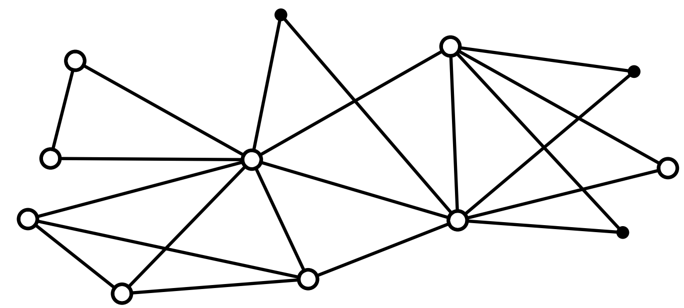
Remove

Compute 3-approx. solution by **greedy approach**



3-Approx. Sol

- Polynomai time



$\circ\ F_0$

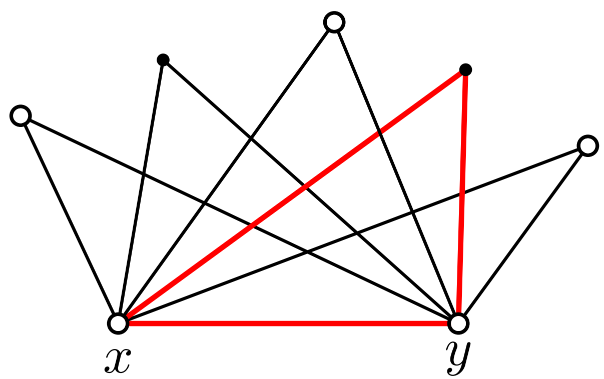
Compute F_0 : Mark $\cup$ 3-Approx

$\rightarrow |F_0| = O(k)$

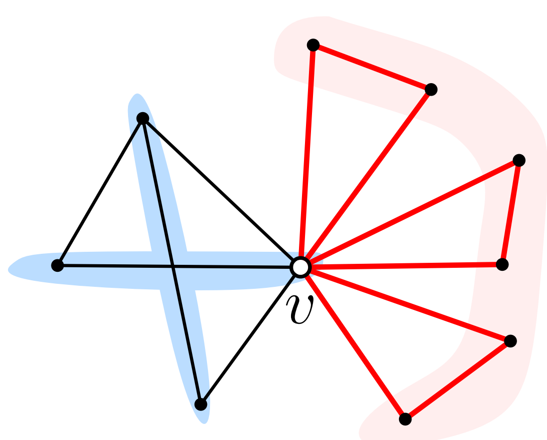
Core from the Branching

p : **maximum ply** and **maximum matching size**

Core : Some triangles of G

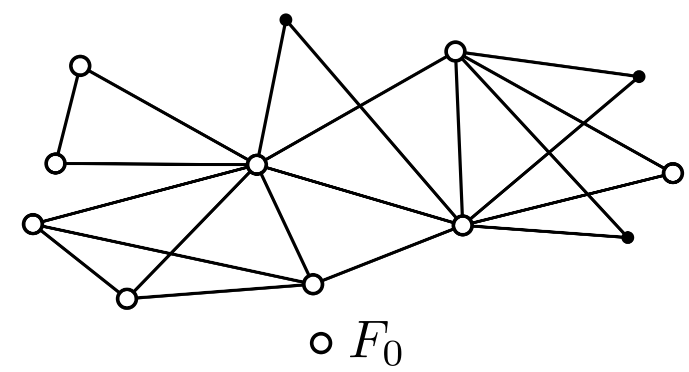


Case 1: A triangle with edge xy of F_0



Case 2: A triangle with $v \in F_0$ and matching of $N^*(v)$

Core size: $O(pk)$



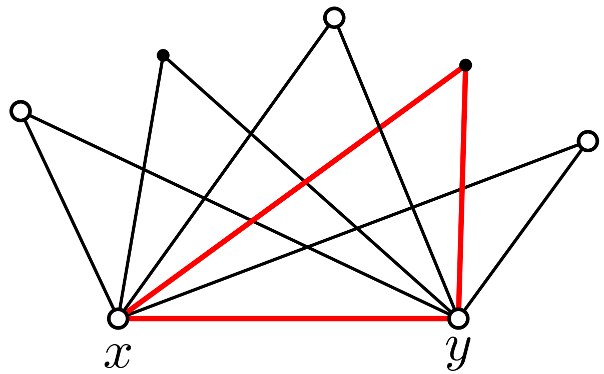
Compute F_0 : Mark $\cup$ 3-Approx

$\rightarrow |F_0| = O(k)$

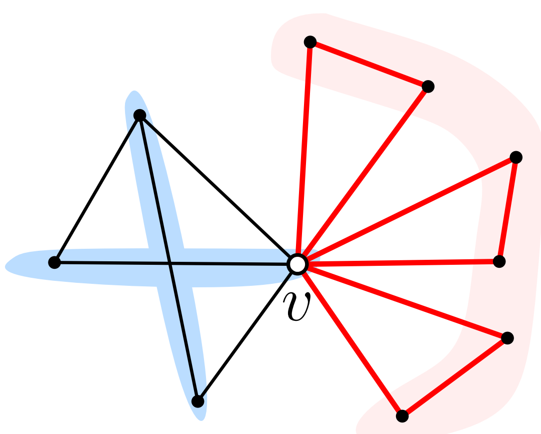
Core from the Branching

p : **maximum ply** and **maximum matching size**

Core : Some triangles of G



Case 1



Case 2

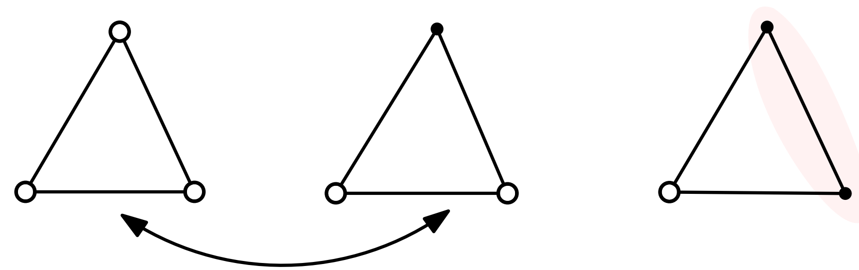
Core size: $O(pk)$

Property of Core:

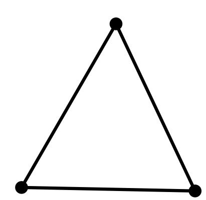
Δ: triangle of G

Δ shares an edge with Δ' of **Core**

Four cases of Δ: $\circ$: F_0 $\bullet$: $G - F_0$



Case 1 **Case 2**



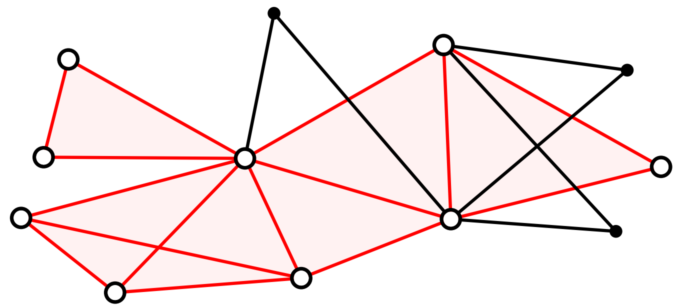
Not exists: $G - F_0$ is triangle-free

Overview: Triangle Hitting Set

(G, k): **yes**-insatnce, p : maximum ply

Step 1. Branching

- $2^{O((k/p)\log k)}$ instances
- Core of size $O(pk)$



Step 2. Kernelization

- $O(pk)$-size Core $\rightarrow$ $O(pk)$-size kernel

Step 3. Treewidth Analysis

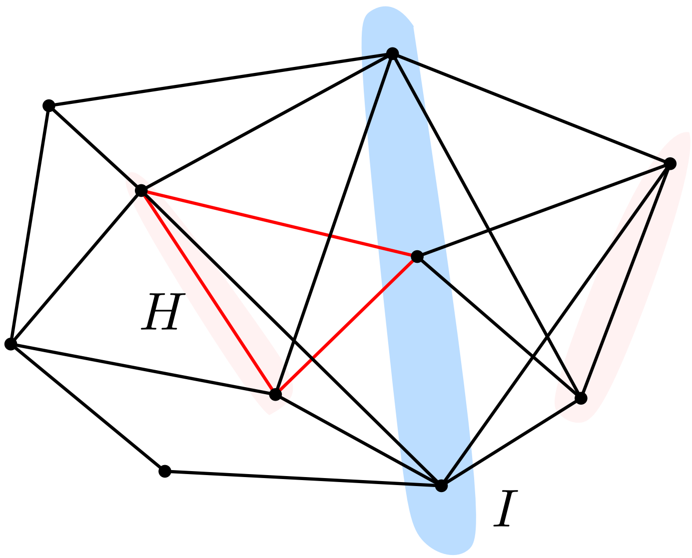
- $O(pk)$-size kernel $\rightarrow$ $O(p^{1.5}\sqrt{k})$ treewidth

Step 4. Dynamic Programming

- Standard DP

Kernelization: Crown Decomposition

Crown Decomposition: Structure of the graph

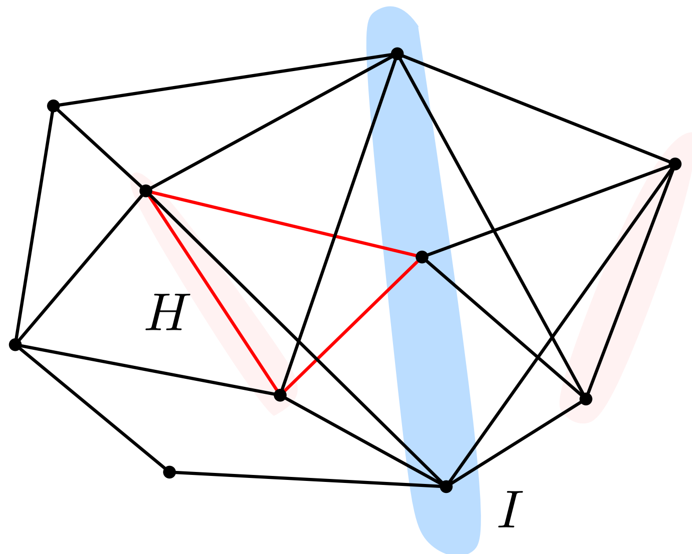


I : vertices (not in same triangle)

H : edges form triangle with I

Kernelization: Crown Decomposition

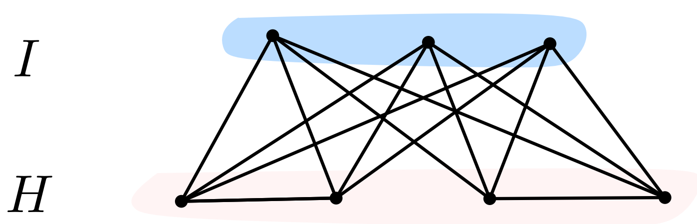
Crown Decomposition: Structure of the graph

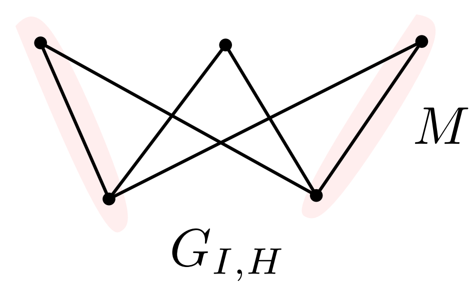


I : vertices (not in same triangle)

H : edges form triangle with I

M: matching of $G_{I,H}$

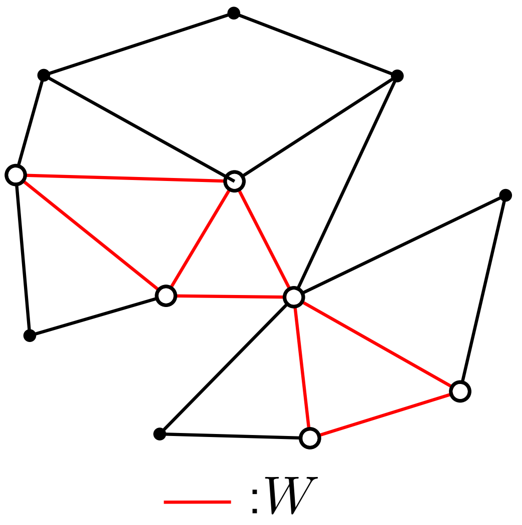




If all H is matched under $M \rightarrow (I, H, M)$ is **Crown Decomposition**

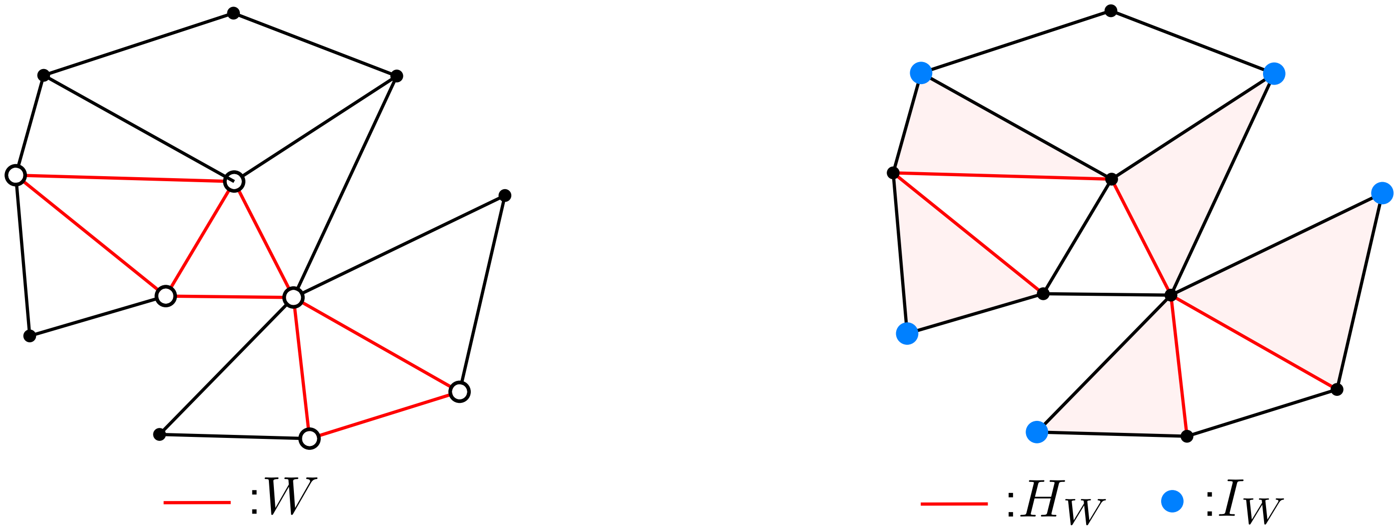
Kernelization: Crown and Core

W : Core of size $O(pk)$



Kernelization: Crown and Core

W : Core of size $O(pk)$

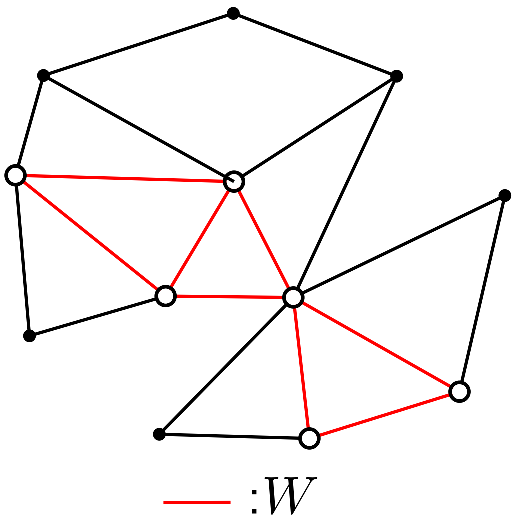


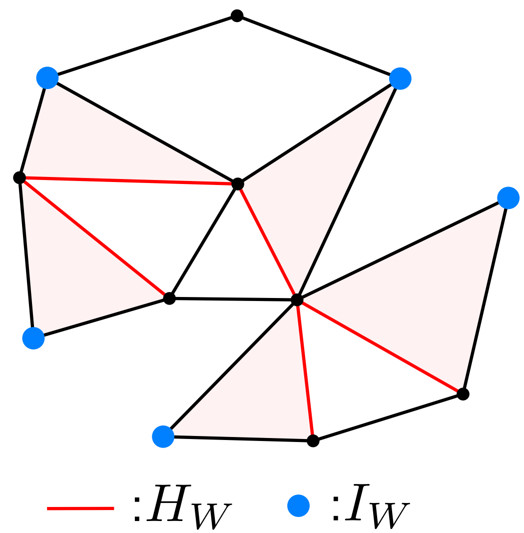
I_W: Vertices of triangle not in W

H_W: Edges form triangle with I_W

Kernelization: Crown and Core

W : Core of size $O(pk)$

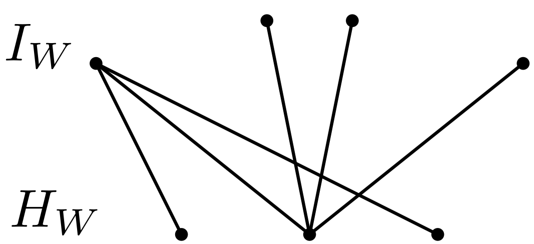




I_W: Vertices of triangle not in W

H_W: Edges form triangle with I_W

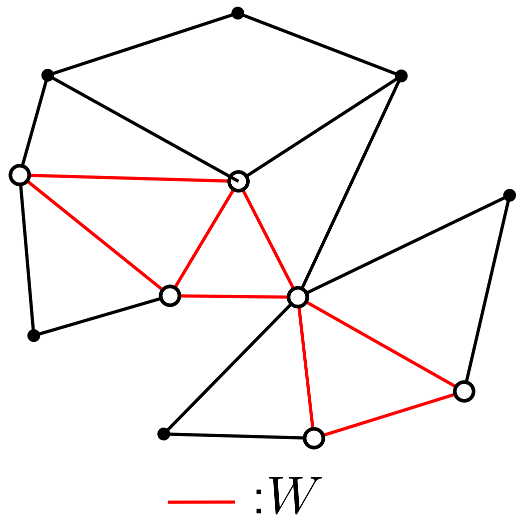
(I_W, H_W) **may not** form a crown.

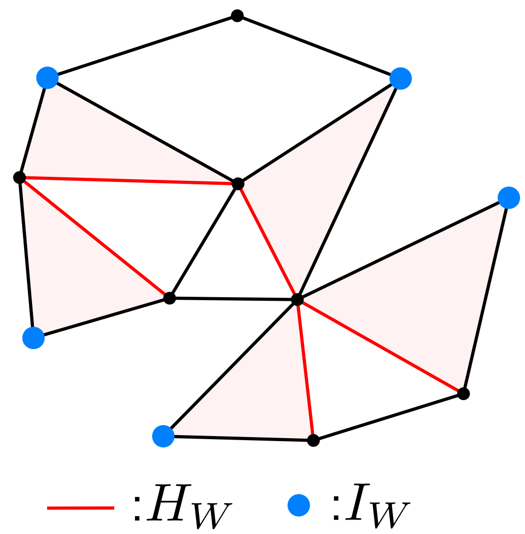


$\rightarrow$ Cannot match all H_W into a matching

Kernelization: Crown and Core

W : Core of size $O(pk)$

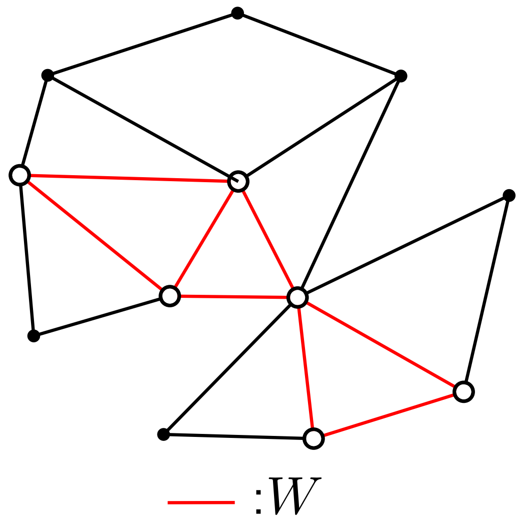


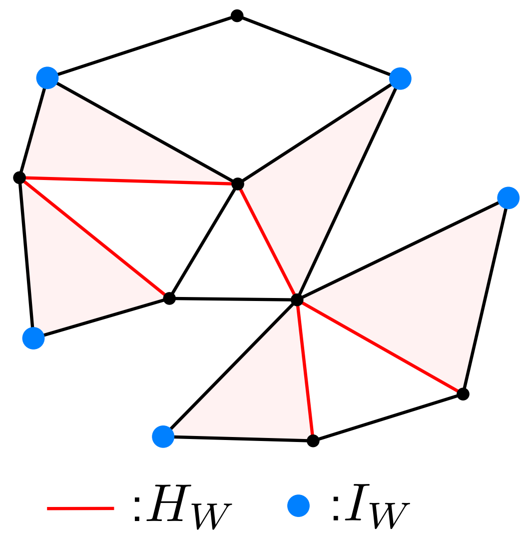


Lemma.:
If $|I_W| > |H_W|$, one can compute a crown (I, H, M) of G with $I \subset I_W$ and $H \subset H_W$.

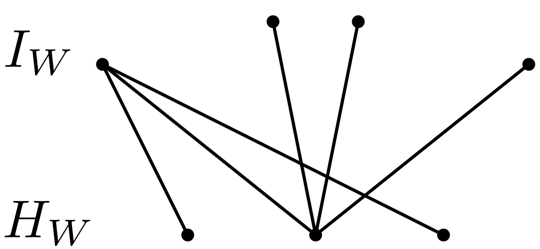
Kernelization: Crown and Core

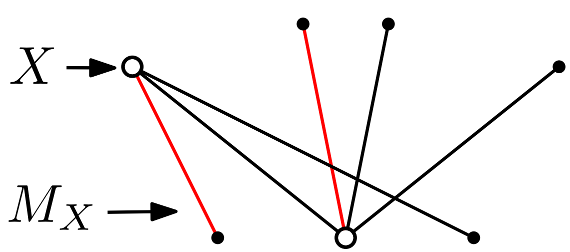
W : Core of size $O(pk)$

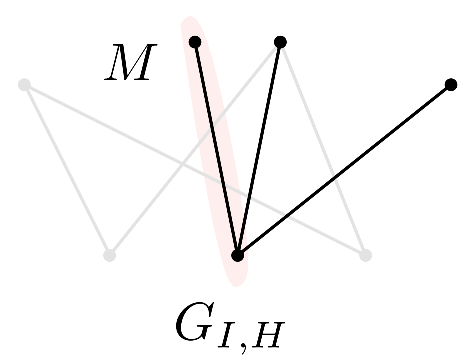




Lemma.:
If $|I_W| > |H_W|$, one can compute a crown (I, H, M) of G with $I \subset I_W$ and $H \subset H_W$.





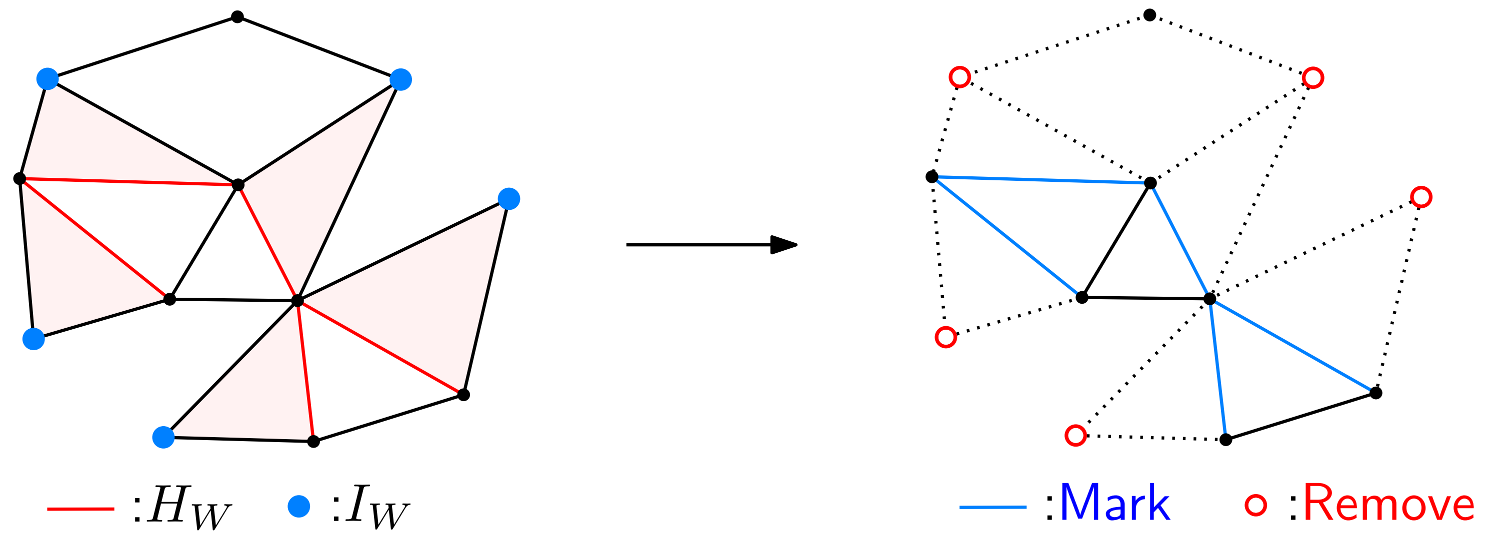


X : Minimum vertex cover, M_X : matching containing X

Kernelization: Crown and Core

W : Core of size $O(pk)$

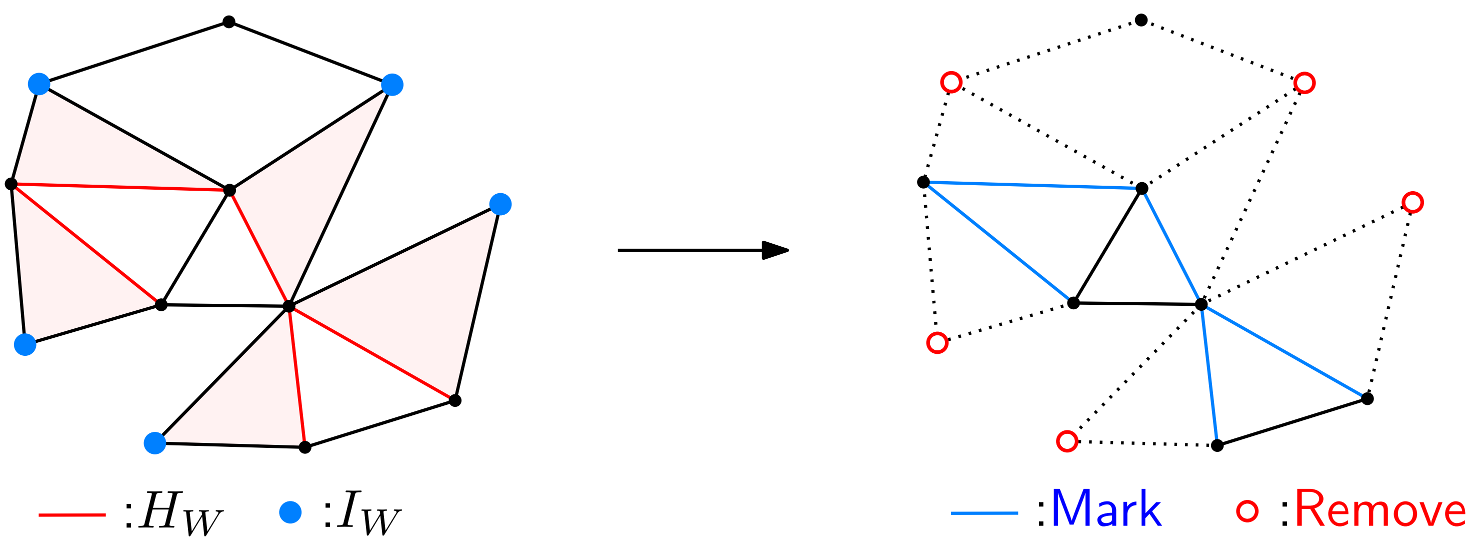
Algorithm: Remove all I_W and mark all H_W



Kernelization: Crown and Core

W : Core of size $O(pk)$

Algorithm: Remove all I_W and mark all H_W



Eventually,

$|\text{Remove}| + |\text{Mark}| > k$ or $I_W \leq H_W$

$I_W \leq H_W \rightarrow |G| = O(|H_W|) = O(W) = O(pk)$

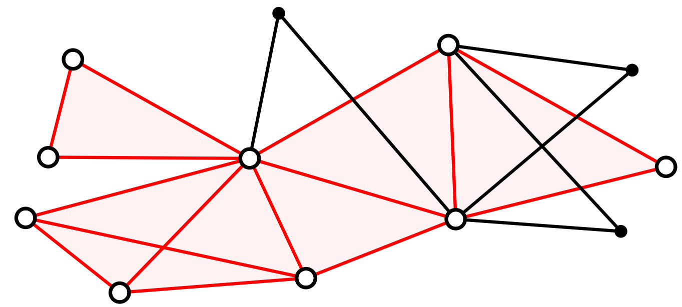
$\rightarrow$ **Small-size kernel.**

Overview: Triangle Hitting Set

(G, k): **yes**-insatnce, p : maximum ply

Step 1. Branching

- $2^{O((k/p)\log k)}$ instances
- Core of size $O(pk)$



Δ :Core

Step 2. Kernelization

- $O(pk)$-size Core $\to$ $O(pk)$-size kernel

Step 3. Treewidth Analysis

- $O(pk)$-size kernel $\to$ $O(p^{1.5}\sqrt{k})$ treewidth

Step 4. Dynamic Programming

- Standard DP

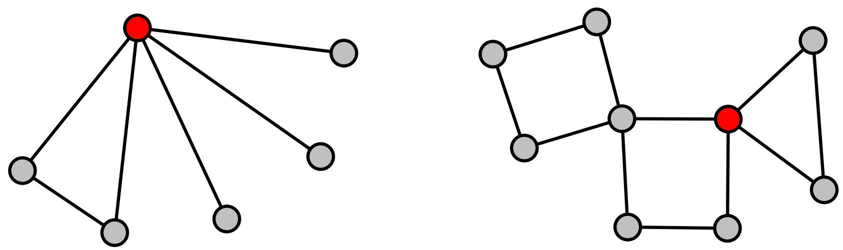
[LPSXZ 23]

Two Cycle Hitting Problems

$G = (V, E)$: Graph, k : integer, S : solution of size k

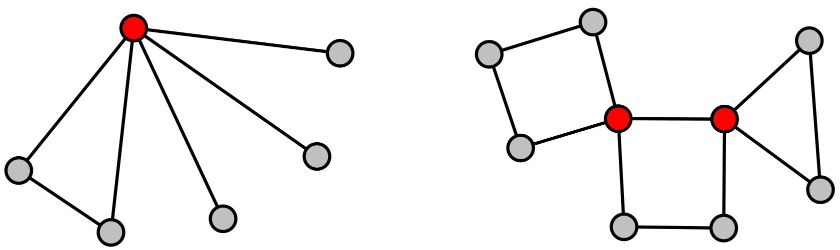
Triangle Hitting Set

$G - S$ is triangle-free



Feedback Vertex Set

$G - S$ is cycle-free



THS : branching $\rightarrow$ kernelizing $\rightarrow$ treewidth

FVS? : branching $\rightarrow$ **cannot kernelize**

Overview: Feedback Vertex Set

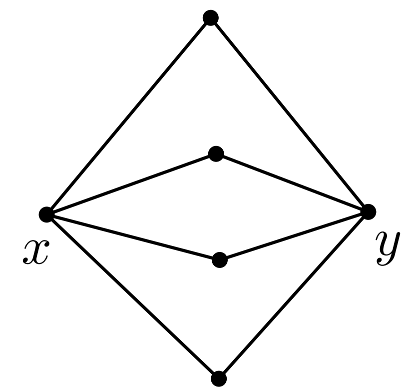
(G, k) : **yes**-instance, p : maximum ply

Step 1. Branching

- $2^{O((k/p)\log k)}$ instances
- Core of size $O(pk)$

Step 2. Cleaning

- Remain one vertex from false twins.



x, y: false twins

Step 3. Treewidth Analysis

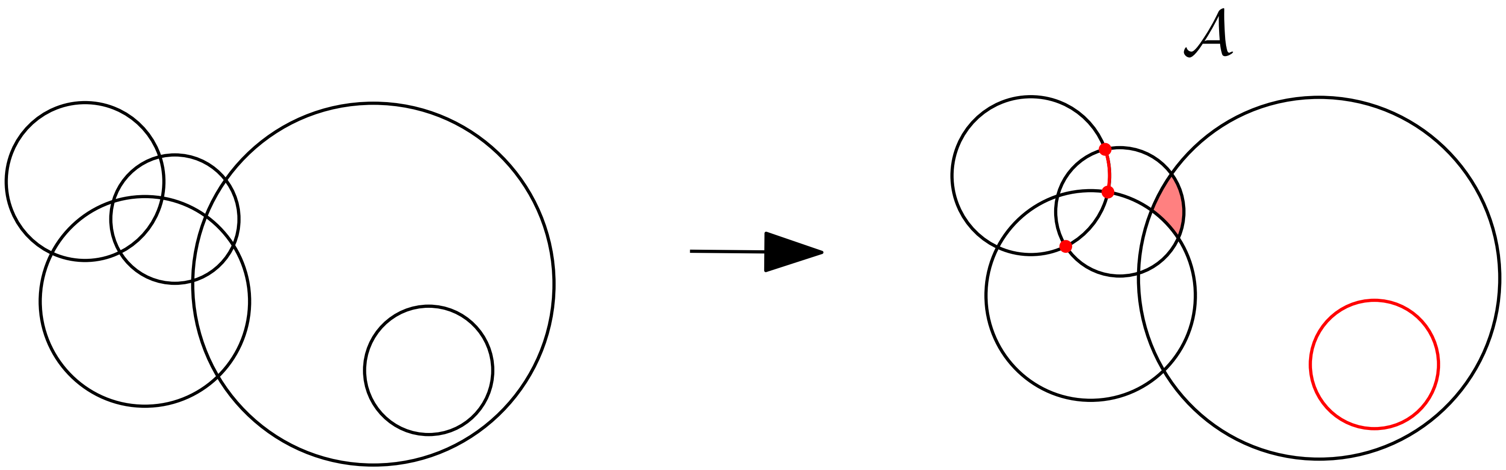
- Classify vertivecs by additively weighted high-order Voronoi diagram.
- Not all classes affect the treewidth. $\rightarrow$ treewidth is $O(p^4\sqrt{k})$.

Step 4. Dynamic Programming

- Standard DP.

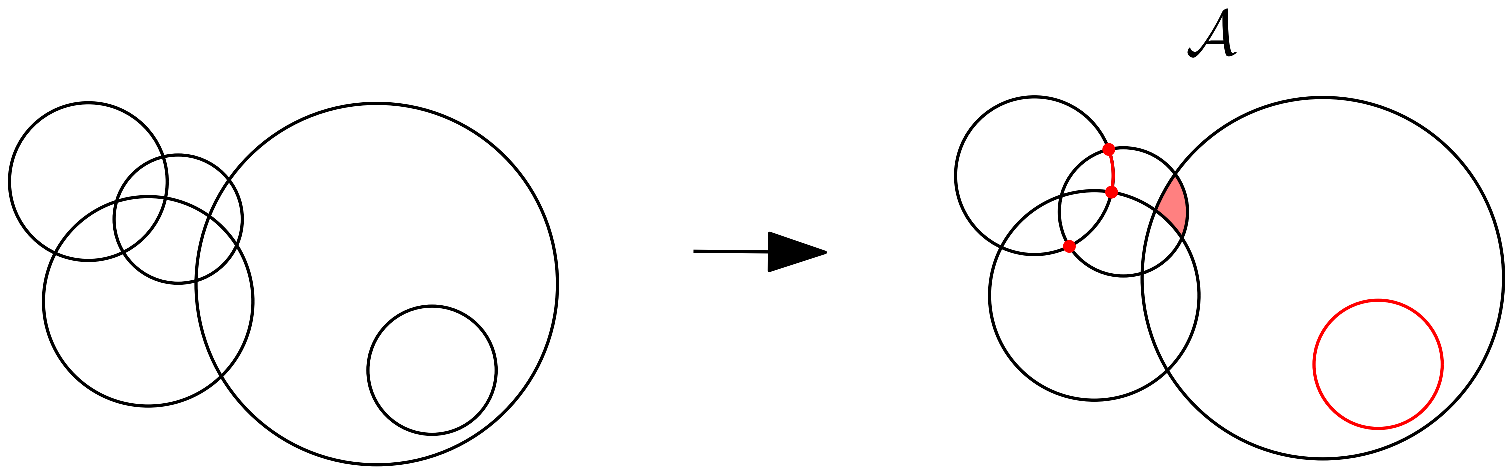
Treewidth Analysis: Classification

Arrangement Graph $\mathcal{A}$: geometric information of the disk graph

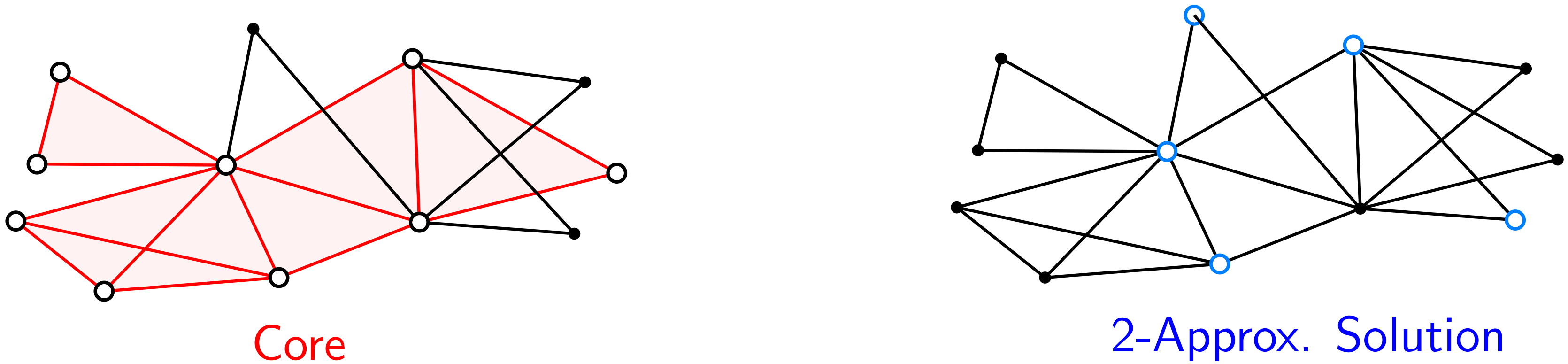


Treewidth Analysis: Classification

Arrangement Graph $\mathcal{A}$: geometric information of the disk graph



Compute arrangement graph of **subset** F of G.

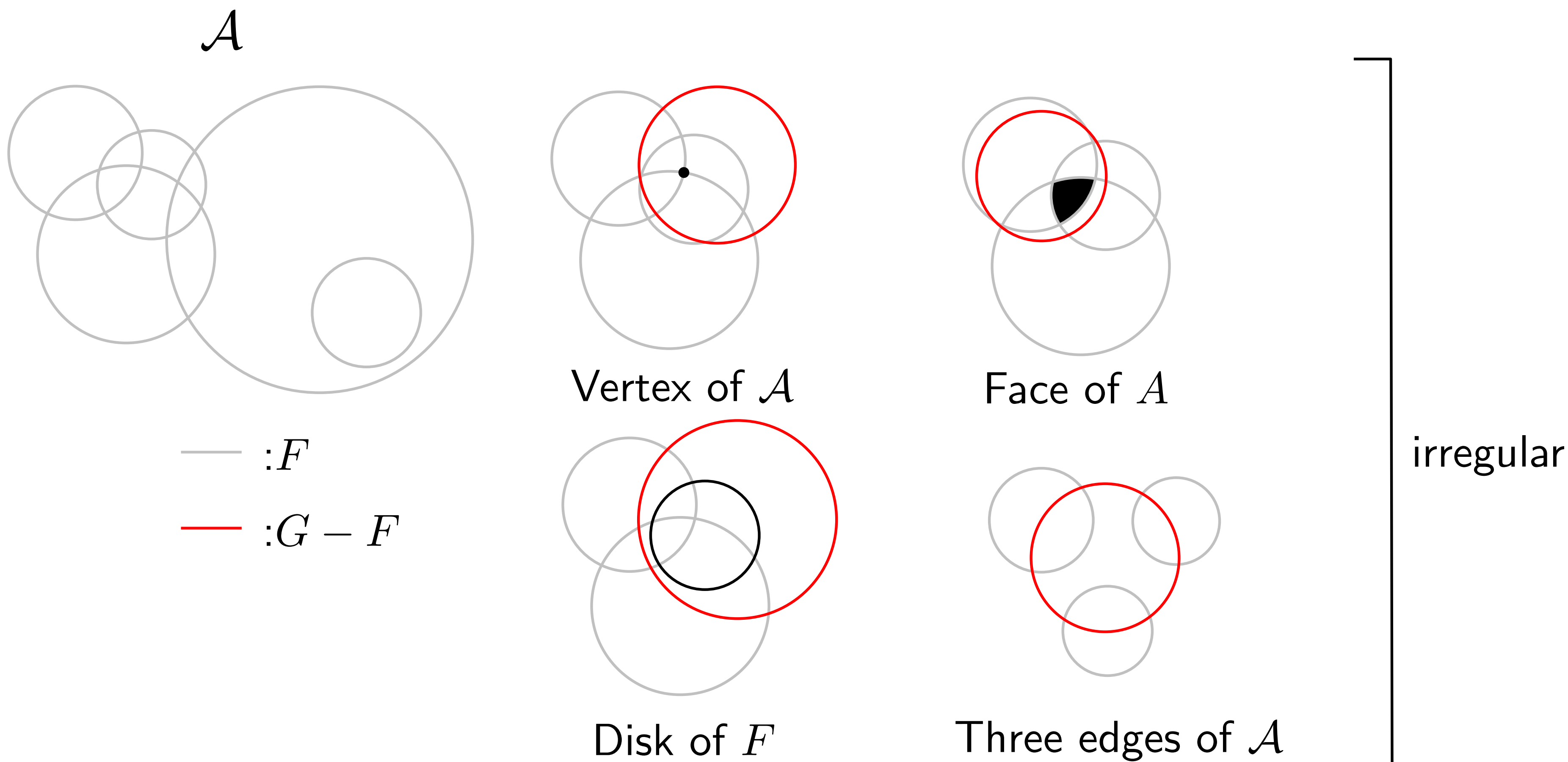


F : Core $\cup$ 2-Approx

- $G - F$ is cycle-free

Treewidth Analysis: Classification

Classify $G - F$ into irregular and regular:



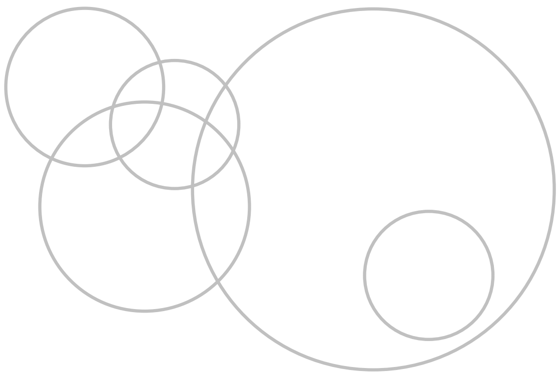
p : maximum ply

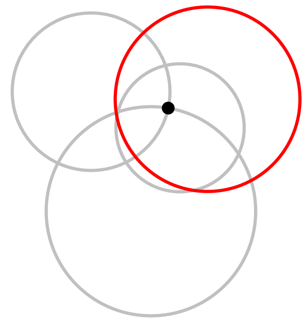
- Complexity of $\mathcal{A}$$= O(p|F|)$
- |Irregular|$=O(p|F|)$

Treewidth Analysis: Classification

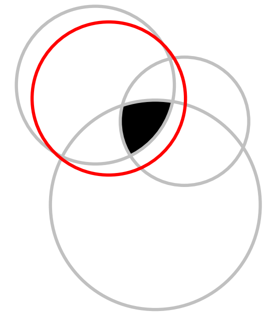
Classify $G - F$ into irregular and regular:

$\mathcal{A}$





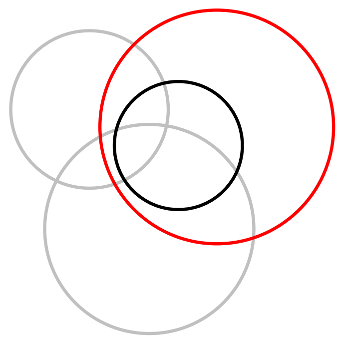
Vertex of $\mathcal{A}$



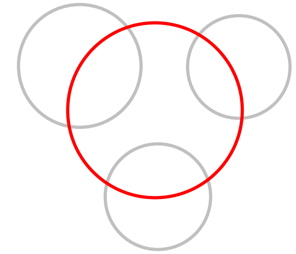
Face of $\mathcal{A}$

$:F$

$:G - F$

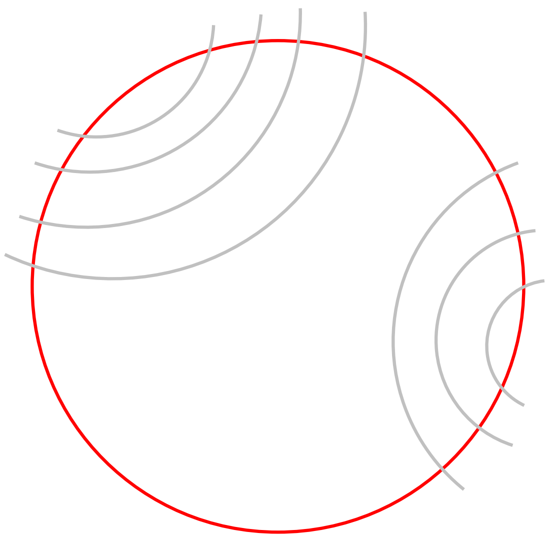


Disk of F



Three edges of $\mathcal{A}$

irregular



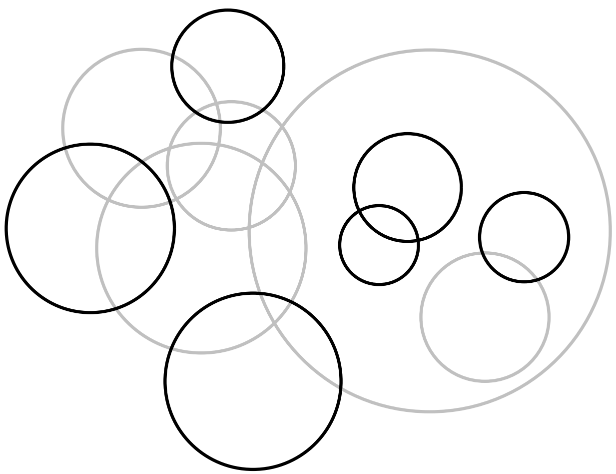
Regular

$\rightarrow$ neighbors in F form two cliques

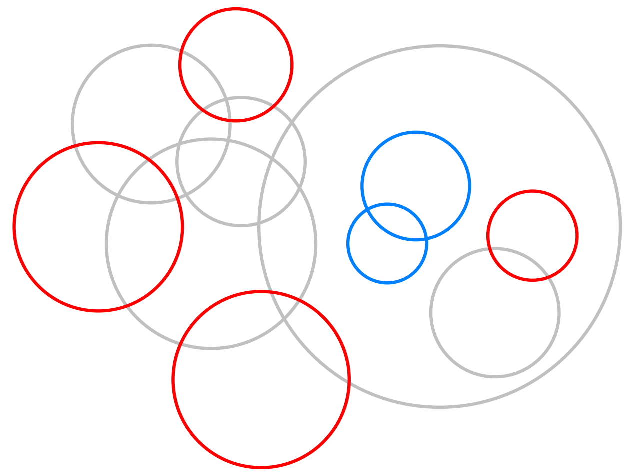
$\rightarrow$ at most $2p$ neighbors

Treewidth Analysis: Classification

Classify $G - F$ into deep and shallow:



— :$G - F$



Deep: All neighbors are F

Shallow: neighbor of $G - F$

Relation to Treewidth:

- |Deep and Regular| is **small** (small treewidth)
- |Shallow and Regular| is large but **irrelevant to treewidth**

Treewidth Analysis: Deep and Regular

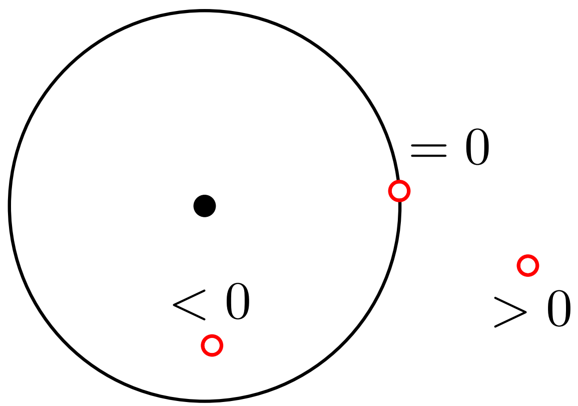
Lemma. The number of deep and regular vertices is $O(p^2|F|)$.

Treewidth Analysis: Deep and Regular

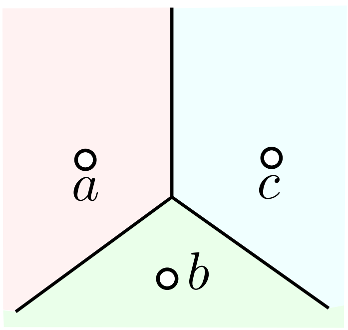
Lemma. The number of deep and regular vertices is $O(p^2|F|)$.

Additively weighted order-r Voronoi diagram:

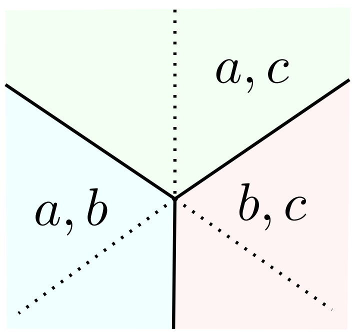
- Subdivision of the plane
- Points in a region has same r-nearest sites



Additively Weighted



Order-1 VD

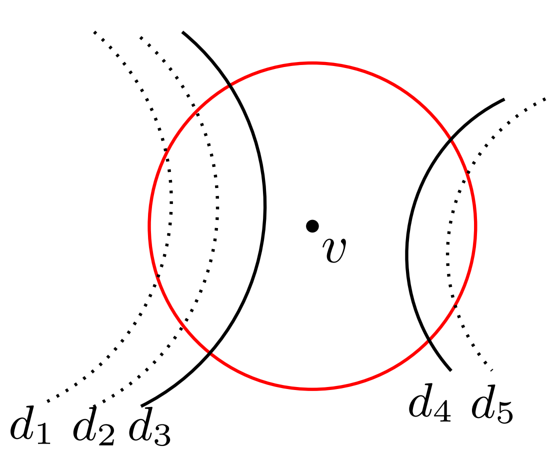


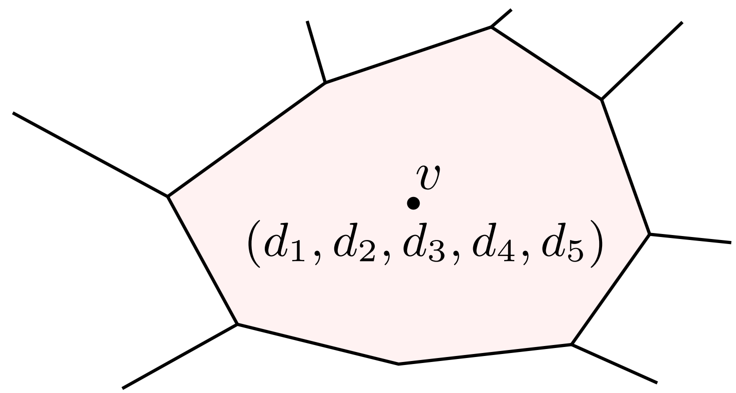
Order-2 VD

Treewidth Analysis: Deep and Regular

Lemma. The number of deep and regular vertices is $O(p^2|F|)$.

Additively weighted order-r Voronoi diagram:



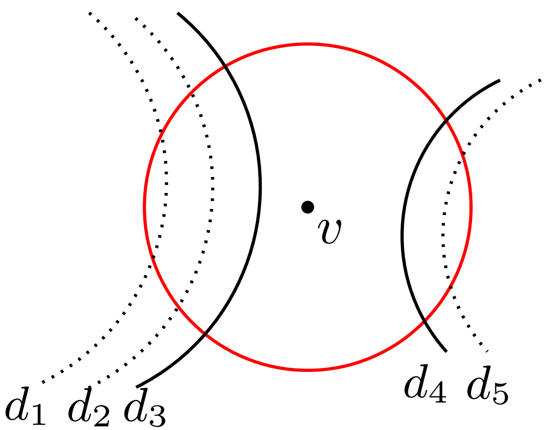


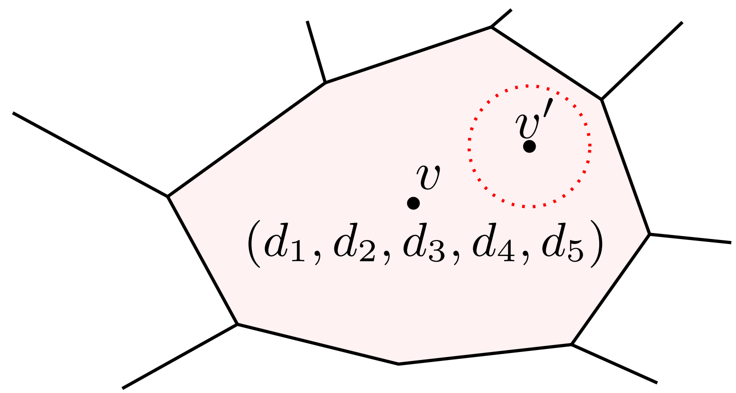
Order-5 Voronoi diagram.

Treewidth Analysis: Deep and Regular

Lemma. The number of deep and regular vertices is $O(p^2|F|)$.

Additively weighted order-r Voronoi diagram:





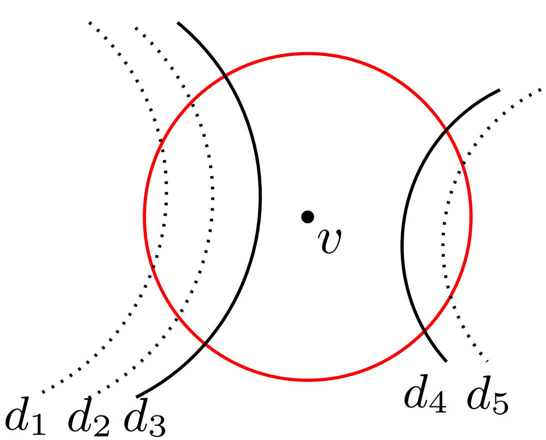
Order-5 Voronoi diagram.

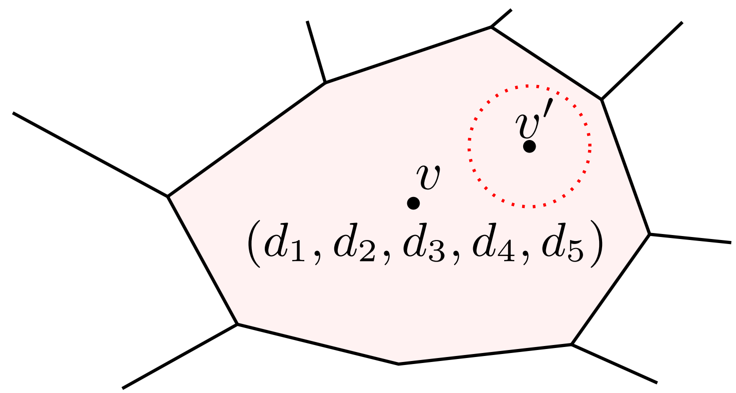
If $|N(v)| = |N(v')| = 5$,
v, v' in same region $\rightarrow$ **false twin**

Treewidth Analysis: Deep and Regular

Lemma. The number of deep and regular vertices is $O(p^2|F|)$.

Additively weighted order-r Voronoi diagram:





Order-5 Voronoi diagram.

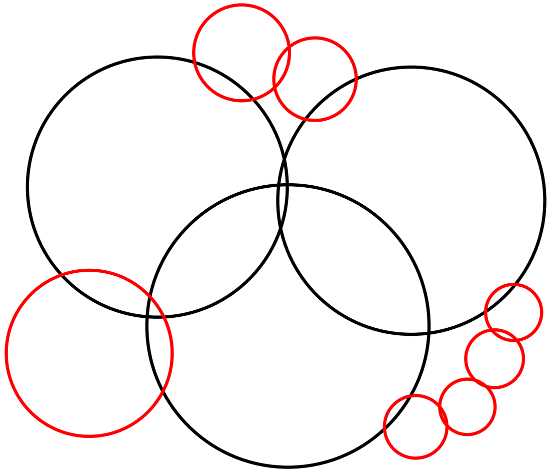
If $|N(v)| = |N(v')| = 5$,
v, v' in same region $\rightarrow$ **false twin**

|Deep|$=\sum_k$ |order-k $-$ VD|

$= O(p^2|F|)$ [Rosenberger 91]

Treewidth Analysis: Shallow and Regular

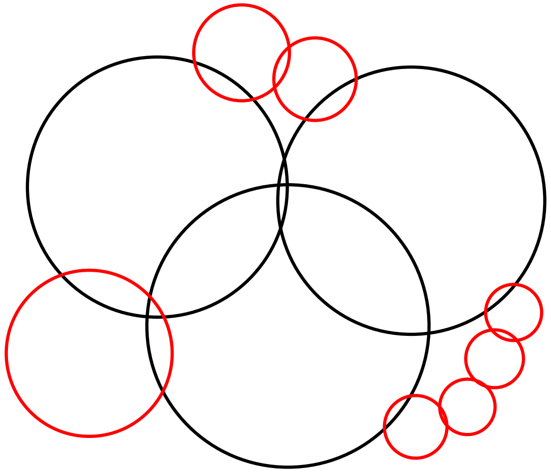
(Sketch) Shallow and Regular vertices irrelevant to the treewidth



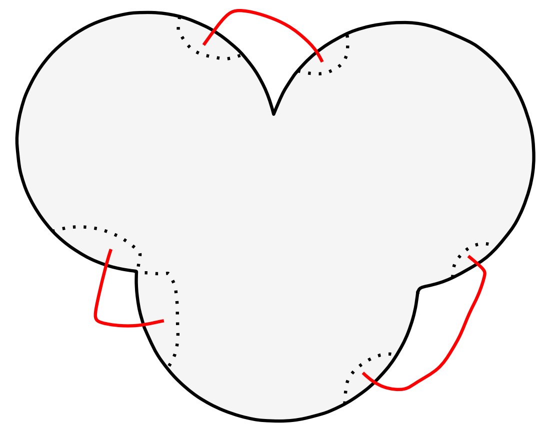
—— : F + Deep + Irregular
—— : Shallow and regular

Treewidth Analysis: Shallow and Regular

(Sketch) Shallow and Regular vertices irrelevant to the treewidth



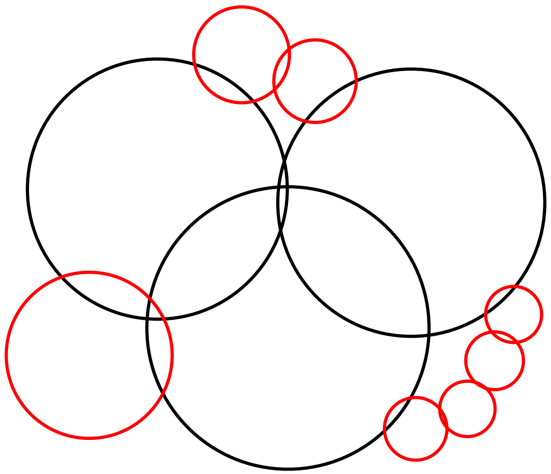
: F + Deep + Irregular
: Shallow and regular



Connect boundary regions

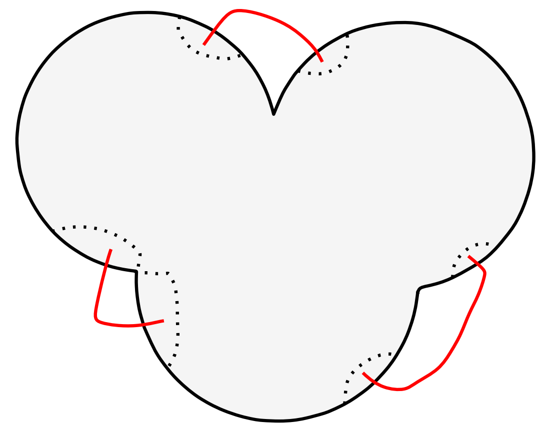
Treewidth Analysis: Shallow and Regular

(Sketch) Shallow and Regular vertices irrelevant to the treewidth

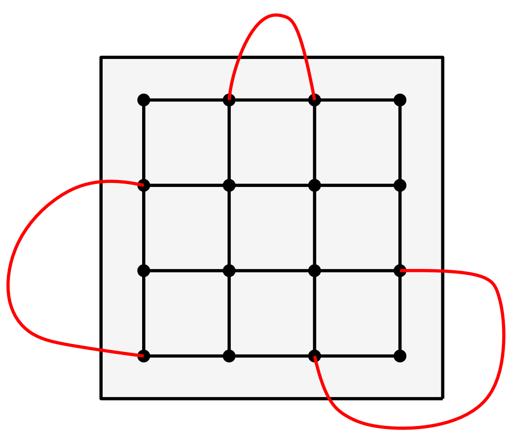


—— : F + Deep + Irregular

—— : Shallow and regular



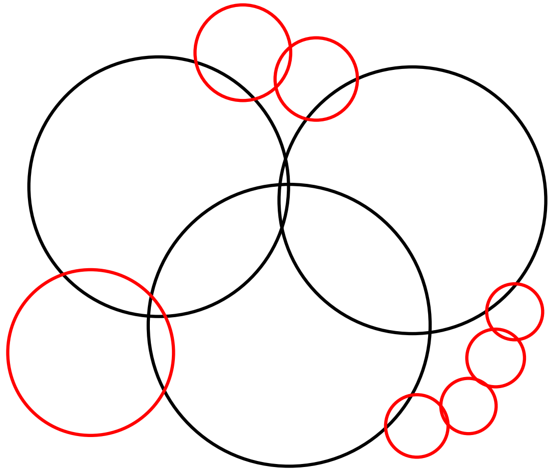
Connect boundary regions



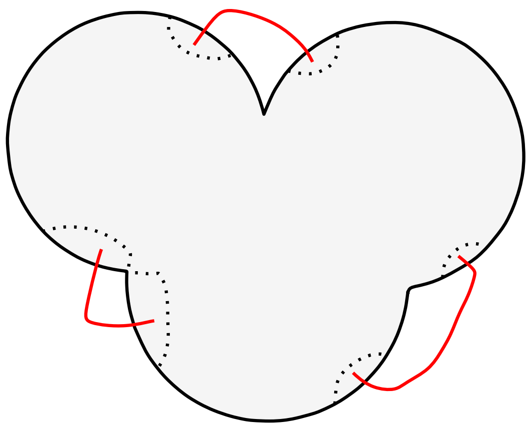
Sub-structure

Treewidth Analysis: Shallow and Regular

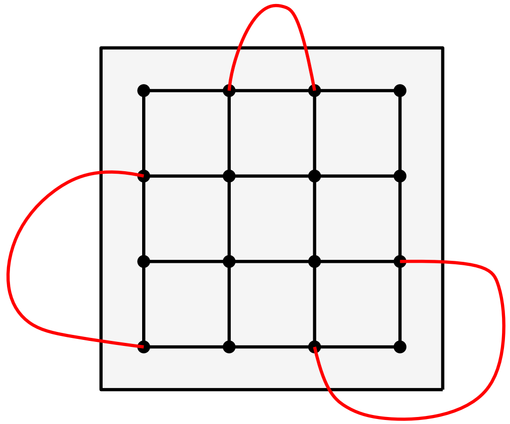
(Sketch) Shallow and Regular vertices irrelevant to the treewidth



—— : F + Deep + Irregular
—— : Shallow and regular



Connect boundary regions

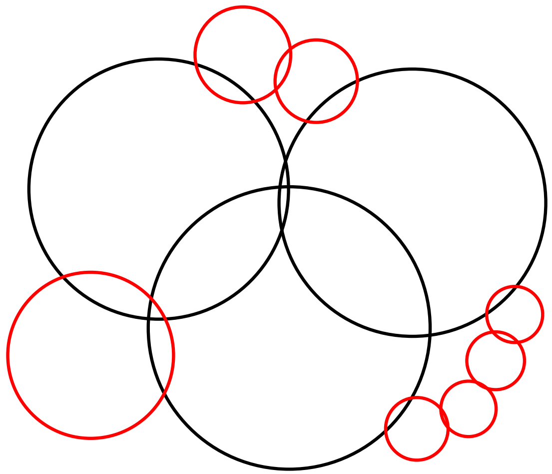


Sub-structure

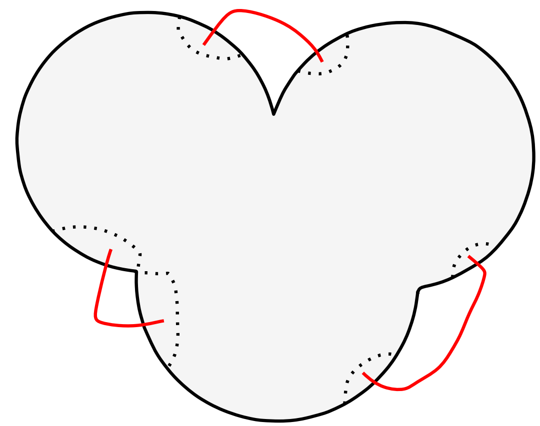
- $t \times t$ grid $\rightarrow$ treewidth t
- $\exists$Red paths $\rightarrow$ still bounded

Treewidth Analysis: Shallow and Regular

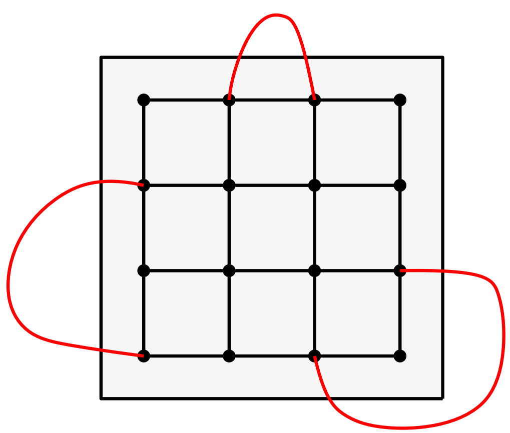
(Sketch) Shallow and Regular vertices irrelevant to the treewidth



—— : F + Deep + Irregular
—— : Shallow and regular



Connect boundary regions



Sub-structure

- $t \times t$ grid $\rightarrow$ treewidth t
- $\exists$Red paths $\rightarrow$ still bounded
- |F+De+Irr|$=O(p^2k)\rightarrow$ treewidth $O(p^4\sqrt{k})$ [LSPXZ 23]

Summary

Progress on **subexponential-time** algorithms for Cycle Hitting problems on **DG**s.

Triangle Hitting Set : $2^{O(k^{9/10}\log k)}n^{O(1)}$ time $\rightarrow 2^{O(k^{4/5}\log k)}n^{O(1)}$ time
Feedback Vertex Set : $2^{O(k^{13/14}\log k)}n^{O(1)}$ time $\rightarrow 2^{O(k^{9/10}\log k)}n^{O(1)}$ time

Summary

Progress on **subexponential-time** algorithms for Cycle Hitting problems on **DG**s.

Triangle Hitting Set : $2^{O(k^{9/10} \log k)} n^{O(1)}$ time $\rightarrow 2^{O(k^{4/5} \log k)} n^{O(1)}$ time
Feedback Vertex Set : $2^{O(k^{13/14} \log k)} n^{O(1)}$ time $\rightarrow 2^{O(k^{9/10} \log k)} n^{O(1)}$ time

If we are aware of the geometric representation,

- $2^{O(k^{2/3} \log k)} n^{O(1)}$-time for Triangle Hitting Set
- $2^{O(k^{7/8} \log k)} n^{O(1)}$-time for Feedback Vertex Set
- $2^{O(k^{15/16} \log k)} n^{O(1)}$-time for Odd Cycle Transversal

Summary

Progress on **subexponential-time** algorithms for Cycle Hitting problems on **DG**s.

Triangle Hitting Set : $2^{O(k^{9/10}\log k)}n^{O(1)}$ time $\rightarrow 2^{O(k^{4/5}\log k)}n^{O(1)}$ time
Feedback Vertex Set : $2^{O(k^{13/14}\log k)}n^{O(1)}$ time $\rightarrow 2^{O(k^{9/10}\log k)}n^{O(1)}$ time

If we are aware of the geometric representation,

- $2^{O(k^{2/3}\log k)}n^{O(1)}$-time for Triangle Hitting Set
- $2^{O(k^{7/8}\log k)}n^{O(1)}$-time for Feedback Vertex Set
- $2^{O(k^{15/16}\log k)}n^{O(1)}$-time for Odd Cycle Transversal

Future Works.

- Does our analysis work for other NP-hard problems?
- Does disk graph admit ETH-tight algorithms as Unit disk graph does?

  ** $2^{O(\sqrt{k})}n^{O(1)}$-time algorithm

Questions?