Subset Feedback Vertex Set in Chordal and Split Graphs

Geevarghese Philip [1], Varun Rajan [1], Saket Saurabh [2,3], and Prafullkumar Tale [3]

May 27, 2019

[1] Chennai Mathematical Institute, Chennai, India,
[2] University of Bergen, Bergen, Norway
[3] The Institute of Mathematical Sciences, HBNI, Chennai, India

The Problem

FEEDBACK VERTEX SET(FVS) **Parameter:** k
Input: A graph $G = (V, E)$, and an integer k
Question: Does there exist a set $S \subseteq V$ of at most k vertices of G such that the subgraph $G[V \setminus S]$ contains no cycle?

Subset Feedback Vertex Set

SUBSET-FVS **Parameter:** k

Input: A graph $G = (V, E)$, a set of *terminal vertices* $T \subseteq V$, and an integer k

Question: Does there exist a set $S \subseteq V$ of at most k vertices of G such that the subgraph $G[V \setminus S]$ contains no T-cycle?

Subset Feedback Vertex Set

SUBSET-FVS **Parameter:** k

Input: A graph $G = (V, E)$, a set of *terminal vertices* $T \subseteq V$, and an integer k

Question: Does there exist a set $S \subseteq V$ of at most k vertices of G such that the subgraph $G[V \setminus S]$ contains no T-cycle?

T*-cycle* is a cycle which contains at least one vertex from T.

Subset Feedback Vertex Set in General Graphs

2000 Even et al. [3] introduced the problem.

Subset Feedback Vertex Set in General Graphs

2000 Even et al. [3] introduced the problem.

—— Generalizes problems like FVS, VC, and MULTIWAY CUT.

Subset Feedback Vertex Set in General Graphs

2000 Even et al. [3] introduced the problem.

$--$ Generalizes problems like FVS, VC, and MULTIWAY CUT.

2011 Cygan et al. [2] and Kawarabayashi and Kobayashi [6] independently proved that the problem is FPT($2^{\mathcal{O}(k \log k)}$).

Subset Feedback Vertex Set in General Graphs

2000 Even et al. [3] introduced the problem.

—— Generalizes problems like FVS, VC, and Multiway Cut.

2011 Cygan et al. [2] and Kawarabayashi and Kobayashi [6] independently proved that the problem is FPT($2^{\mathcal{O}(k \log k)}$).

2011 Fomin et al. [4]: No $2^{o(k)}$ algorithm under the ETH.

Subset Feedback Vertex Set in General Graphs

2000 Even et al. [3] introduced the problem.

–– Generalizes problems like FVS, VC, and MULTIWAY CUT.

2011 Cygan et al. [2] and Kawarabayashi and Kobayashi [6] independently proved that the problem is FPT($2^{\mathcal{O}(k \log k)}$).

2011 Fomin et al. [4]: No $2^{o(k)}$ algorithm under the ETH.

2014 Wahlström [9] gave algorithm running in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.

Subset Feedback Vertex Set in General Graphs

2000 Even et al. [3] introduced the problem.

—— Generalizes problems like FVS, VC, and MULTIWAY CUT.

2011 Cygan et al. [2] and Kawarabayashi and Kobayashi [6] independently proved that the problem is FPT($2^{\mathcal{O}(k \log k)}$).

2011 Fomin et al. [4]: No $2^{o(k)}$ algorithm under the ETH.

2014 Wahlström [9] gave algorithm running in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.

2015 Lokshtanov et al. [7] presented a different FPT algorithm which has linear dependence on the input size.

Subset Feedback Vertex Set in General Graphs

2000 Even et al. [3] introduced the problem.

—— Generalizes problems like FVS, VC, and MULTIWAY CUT.

2011 Cygan et al. [2] and Kawarabayashi and Kobayashi [6] independently proved that the problem is FPT($2^{\mathcal{O}(k \log k)}$).

2011 Fomin et al. [4]: No $2^{o(k)}$ algorithm under the ETH.

2014 Wahlström [9] gave algorithm running in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.

2015 Lokshtanov et al. [7] presented a different FPT algorithm which has linear dependence on the input size.

2016 Hols and Kratsch [5] obtained a randomized polynomial kernel with $\mathcal{O}(k^9)$ vertices.

Subset Feedback Vertex Set in Chordal and Split Graphs

– A graph is *chordal* if it does not contain induced cycles of length four or larger.

Subset Feedback Vertex Set in Chordal and Split Graphs

- A graph is *chordal* if it does not contain induced cycles of length four or larger.
- A graph is called *split graph* if its vertex set can be partitioned into a clique and an independent set.

Subset Feedback Vertex Set in Chordal and Split Graphs

- A graph is *chordal* if it does not contain induced cycles of length four or larger.
- A graph is called *split graph* if its vertex set can be partitioned into a clique and an independent set.
- Every split graph is chordal.

Subset Feedback Vertex Set in Chordal and Split Graphs

- A graph is *chordal* if it does not contain induced cycles of length four or larger.
- A graph is called *split graph* if its vertex set can be partitioned into a clique and an independent set.
- Every split graph is chordal.

SUBSET-FVS ON CHORDAL GRAPHS **Parameter:** k

Input: A chordal graph $G = (V, E)$, a set of *terminal vertices* $T \subseteq V$, and an integer k

Question: Does there exist a set $S \subseteq V$ of at most k vertices of G such that the subgraph $G[V \setminus S]$ contains no T-cycle?

Subset Feedback Vertex Set in Chordal and Split Graphs

- A graph is *chordal* if it does not contain induced cycles of length four or larger.
- A graph is called *split graph* if its vertex set can be partitioned into a clique and an independent set.
- Every split graph is chordal.

SUBSET-FVS ON CHORDAL GRAPHS **Parameter:** k
Input: A chordal graph $G = (V, E)$, a set of *terminal vertices* $T \subseteq V$, and an integer k
Question: Does there exist a set $S \subseteq V$ of at most k vertices of G such that the subgraph $G[V \setminus S]$ contains no T-cycle?

The problem is NP-Complete even on split graphs [4].

To intersect every T-cycle in a *chordal* graph it is sufficient and necessary to intersect all *T-triangles* in the graph.

To intersect every T-cycle in a *chordal* graph it is sufficient and necessary to intersect all T-*triangles* in the graph.

SUBSET-FVS IN CHORDAL to 3-HITTING SET.

(*Parameter Preserving Reduction*)

Subset-FVS in Chordal to 3-Hitting Set.

Subset-FVS in Chordal to 3-Hitting Set.

- 3-Hitting Set has a polynomial kernel of size $\mathcal{O}(k^3)$ [1]

$\Rightarrow$ *Polynomial compression* of size $\mathcal{O}(k^3)$

- We improve it to $\mathcal{O}(k^2)$ kernel for split graphs.

∘ 3-Hitting Set is FPT (running time $2.076^k \cdot n^{\mathcal{O}(1)}$ [8])

$\Rightarrow$ FPT algorithm running in time $2.076^k \cdot n^{\mathcal{O}(1)}$

∘ We improve it to $2^k \cdot \mathcal{O}(n+m)$ for chordal graphs.

SUBSET-FVS IN CHORDAL to 3-HITTING SET.

– 3-HITTING SET has a polynomial kernel of size $\mathcal{O}(k^3)$ [1]

⇒ *Polynomial compression* of size $\mathcal{O}(k^3)$

– We improve it to $\mathcal{O}(k^2)$ kernel for split graphs.

∘ 3-HITTING SET is FPT (running time $2.076^k \cdot n^{\mathcal{O}(1)}$ [8])

⇒ FPT algorithm running in time $2.076^k \cdot n^{\mathcal{O}(1)}$

∘ We improve it to $2^k \cdot \mathcal{O}(n+m)$ for chordal graphs.

SUBSET-FVS IN CHORDAL to 3-HITTING SET.

– 3-HITTING SET has a polynomial kernel of size $\mathcal{O}(k^3)$ [1]

$\Rightarrow$ *Polynomial compression* of size $\mathcal{O}(k^3)$

– We improve it to $\mathcal{O}(k^2)$ kernel for split graphs.

◦ 3-HITTING SET is FPT (running time $2.076^k \cdot n^{\mathcal{O}(1)}$ [8])

$\Rightarrow$ FPT algorithm running in time $2.076^k \cdot n^{\mathcal{O}(1)}$

◦ We improve it to $2^k \cdot \mathcal{O}(n+m)$ for chordal graphs.

SUBSET-FVS IN CHORDAL to 3-HITTING SET.

– 3-HITTING SET has a polynomial kernel of size $\mathcal{O}(k^3)$ [1]

$\Rightarrow$ *Polynomial compression* of size $\mathcal{O}(k^3)$

– We improve it to $\mathcal{O}(k^2)$ kernel for split graphs.

∘ 3-HITTING SET is FPT (running time $2.076^k \cdot n^{\mathcal{O}(1)}$ [8])

$\Rightarrow$ FPT algorithm running in time $2.076^k \cdot n^{\mathcal{O}(1)}$

∘ We improve it to $2^k \cdot \mathcal{O}(n+m)$ for chordal graphs.

Our Results

Theorem

SUBSET-FVS IN SPLIT *admits a kernel with* $\mathcal{O}(k^2)$ *vertices and* $\mathcal{O}(k^2)$ *edges.*

No polynomial kernel of size $\mathcal{O}(k^{2-\epsilon})$ bits, unless NP $\subseteq$ coNP/*poly*.

Theorem

SUBSET-FVS IN CHORDAL *admits an* FPT *algorithm with running time* $\mathcal{O}(2^k(n+m))$.

No $2^{o(k)}$ algorithm under ETH [4].

Our Results

Theorem

Subset-FVS in Split *admits a kernel with $\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^2)$ edges.*

No polynomial kernel of size $\mathcal{O}(k^{2-\epsilon})$ bits, unless NP $\subseteq$ coNP/*poly*.

Theorem

Subset-FVS in Chordal *admits an* FPT *algorithm with running time $\mathcal{O}(2^k(n+m))$.*

No $2^{o(k)}$ algorithm under ETH [4].

Kernel : Overview

Overview of Kernelization

(K, I) is a split partition ($K-$ clique and $I-$ independent set)

Step 1: Reduce the input to an instance $(G; T; k)$ where the terminal set T is exactly the independent set I

Step 2: If $v \in K$ has at least $k+1$ neighbours in I then either include v in a solution or delete an edge incident with v;
Each $v \in K$ has at most k neighbours in I.

Step 3: Bound the number of vertices in K by $10k$;
An instance with $\mathcal{O}(k^2)$ vertices in I.

Overview of Kernelization

(K, I) is a split partition ($K-$ clique and $I-$ independent set)

Step 1: Reduce the input to an instance $(G; T; k)$ where the terminal set T is exactly the independent set I

Step 2: If $v \in K$ has at least $k + 1$ neighbours in I then either include v in a solution or delete an edge incident with v;

Each $v \in K$ has at most k neighbours in I.

Step 3: Bound the number of vertices in K by $10k$;

An instance with $\mathcal{O}(k^2)$ vertices in I.

Overview of Kernelization

(K, I) is a split partition ($K-$ clique and $I-$ independent set)

Step 1: Reduce the input to an instance $(G; T; k)$ where the terminal set T is exactly the independent set I

Step 2: If $v \in K$ has at least $k+1$ neighbours in I then either include v in a solution or delete an edge incident with v;
Each $v \in K$ has at most k neighbours in I.

Step 3: Bound the number of vertices in K by $10k$;
An instance with $O(k^2)$ vertices in I.

Overview of Kernelization

(K, I) is a split partition ($K-$ clique and $I-$ independent set)

Step 1: Reduce the input to an instance $(G; T; k)$ where the terminal set T is exactly the independent set I

Step 2: If $v \in K$ has at least $k+1$ neighbours in I then either include v in a solution or delete an edge incident with v;
Each $v \in K$ has at most k neighbours in I.

Step 3: Bound the number of vertices in K by $10k$;
An instance with $\mathcal{O}(k^2)$ vertices in I.

Overview of Kernelization

(K, I) is a split partition ($K-$ clique and $I-$ independent set)

Step 1: Reduce the input to an instance $(G; T; k)$ where the terminal set T is exactly the independent set I

Step 2: If $v \in K$ has at least $k+1$ neighbours in I then either include v in a solution or delete an edge incident with v;
Each $v \in K$ has at most k neighbours in I.

Step 3: Bound the number of vertices in K by $10k$;
An instance with $\mathcal{O}(k^2)$ vertices in I.

Overview of Kernelization

(K, I) is a split partition ($K-$ clique and $I-$ independent set)

Step 1: Reduce the input to an instance $(G; T; k)$ where the terminal set T is exactly the independent set I

Step 2: If $v \in K$ has at least $k + 1$ neighbours in I then either include v in a solution or delete an edge incident with v;
Each $v \in K$ has at most k neighbours in I.

Step 3: Bound the number of vertices in K by $10k$;
An instance with $\mathcal{O}(k^2)$ vertices in I.

Kernel : Step 1

Simple Reduction Rules

Simple Reduction Rules

– Delete isolated vertices.

Simple Reduction Rules

- Delete isolated vertices.
- Delete a *non-terminal* vertex which is *not* adjacent to a terminal vertex.

Simple Reduction Rules

- Delete isolated vertices.
- Delete a *non-terminal* vertex which is *not* adjacent to a terminal vertex.
- Delete a cut edge.

Simple Reduction Rules

- Delete isolated vertices.
- Delete a *non-terminal* vertex which is *not* adjacent to a terminal vertex.
- Delete a cut edge.
- If there is a terminal vertex t on the clique side and clique side is $\geq k+3$ then include t in solution.

After Simplification

1. Each vertex in G has degree at least two.
2. Every vertex in G is part of some T-triangle.
3. Let (K, I) be the split partition of G. Then $T = I$ and every vertex in K has a neighbour in I.

After Simplification

1. Each vertex in G has degree at least two.
2. Every vertex in G is part of some T-triangle.
3. Let (K, I) be the split partition of G. Then $T = I$ and every vertex in K has a neighbour in I.

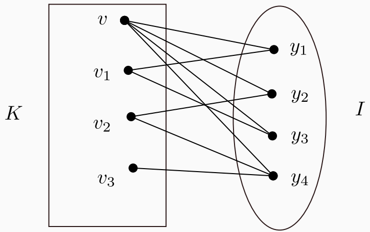
It is enough to bound

- the number of adjacent vertices in independent set for each vertex in clique-side, and
- the size of clique-side.

Kernel : Step 2 (Reducing neighbours in Ind-Set side)

For a vertex $v \in K$ on the clique side define

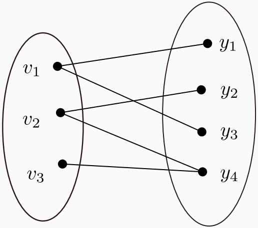
- First Nbrs: the set of neighbours on the independent side I.
  $N_1(v) = N(v) \cap I$
- Second Nbrs: the second neighbourhood "going via I".
  $N_2(v) = N(N_1(v)) \setminus \{v\}$.



$N_1(v) = \{y_1, y_2, y_3, y_4\}$

$N_2(v) = \{v_1, v_2, v_3\}$

Bipartite graph $(B(v))$ *corresponding to* vertex $v \in K$: Graph obtained from $G[N_1(v) \cup N_2(v)]$ by deleting every edge with both its endvertices in $N_2(v)$.



$B(v)$

Let ℓ be the size of largest matching in $B(v)$

Let ℓ be the size of largest matching in $B(v)$

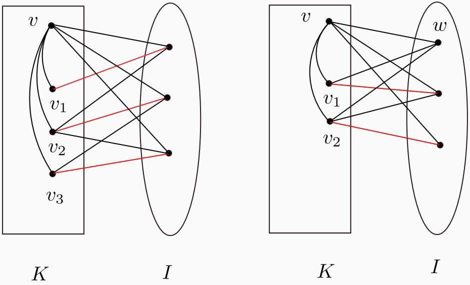
- if $\ell \geq k+1$ then include v in solution and delete it.

Let ℓ be the size of largest matching in $B(v)$

- if $\ell \geq k+1$ then include v in solution and delete it.
- if $\ell \leq k$ then delete a particular edge incident on v

Let ℓ be the size of largest matching in $B(v)$

- if $\ell \geq k+1$ then include v in solution and delete it.
- if $\ell \leq k$ then delete a particular edge incident on v



- Too many vertex disjoint triangles intersecting at v.
- Matching edges are enough to store information regarding T-cycles using vw.

Maximum Matching is $\geq k + 1$

Reduction Rule

If there is a vertex v on the clique side K of graph G s.t. the bipartite graph $B(v)$ has a matching of size at least $k + 1$ then include vertex v in solution.

Maximum Matching is $\geq k+1$

Reduction Rule

If there is a vertex v on the clique side K of graph G s.t. the bipartite graph $B(v)$ has a matching of size at least $k+1$ then include vertex v in solution.

At least $k+1$ many T-triangles which intersect only at v

Maximum Matching is $\geq k + 1$

Reduction Rule

If there is a vertex v on the clique side K of graph G s.t. the bipartite graph $B(v)$ has a matching of size at least $k + 1$ then include vertex v in solution.

At least $k + 1$ many T-triangles which intersect only at v

We need Expansion Lemma to handle second case.

Expansion Lemma and its Matching version

t-expansion

$G(P, Q)$ – a bipartite graph; t – a positive integer

A set of edges $M \subseteq E(G)$ is called a t-expansion of X into Y if

t-expansion

$G(P, Q)$ – a bipartite graph; t – a positive integer

A set of edges $M \subseteq E(G)$ is called a t-expansion of X into Y if

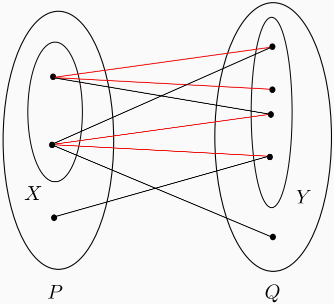
- every vertex of X is incident with exactly t edges of M, and
- the number of vertices in Y which are incident with at least one edge in M is exactly $t|P|$.

t-expansion

$G(P, Q)$ – a bipartite graph; t – a positive integer

A set of edges $M \subseteq E(G)$ is called a t-expansion of X into Y if

- every vertex of X is incident with exactly t edges of M, and
- the number of vertices in Y which are incident with at least one edge in M is exactly $t|P|$.



Expansion Lemma

$G(P,Q)$ – a bipartite graph; t – a positive integer s.t.

Expansion Lemma

$G(P, Q)$ – a bipartite graph; t – a positive integer s.t.

- $|Q| \geq t|P|$
- there are no isolated vertices in Q.

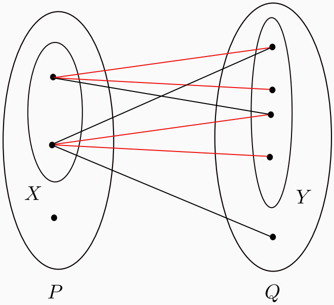
Expansion Lemma

$G(P, Q)$ – a bipartite graph; t – a positive integer s.t.

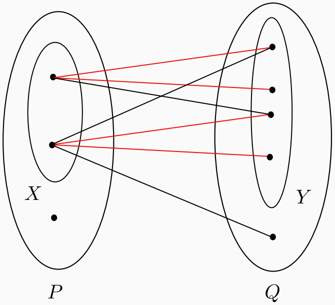
- $|Q| \geq t|P|$
- there are no isolated vertices in Q.

Then there exist nonempty sets $X \subseteq P$ and $Y \subseteq Q$ s.t.

- X has a t-expansion into Y, and
- no vertex in Y has a neighbour outside X.



- X has a t-expansion into Y, and
- no vertex in Y has a neighbour outside X.



- X has a t-expansion into Y, and
- no vertex in Y has a neighbour outside X.

X and Y can be found in poly-time.

Expansion Lemma (Matching Version)

$G(P, Q)$ – a bipartite graph; t – a positive integer;
ℓ – the size of a maximum matching in G s.t.

Expansion Lemma (Matching Version)

$G(P, Q)$ – a bipartite graph; t – a positive integer;
ℓ – the size of a maximum matching in G s.t.

- $|Q| \geq t\ell$
- there are no isolated vertices in Q.

Expansion Lemma (Matching Version)

$G(P, Q)$ – a bipartite graph; t – a positive integer;
ℓ – the size of a maximum matching in G s.t.

- $|Q| \geq t\ell$
- there are no isolated vertices in Q.

Then there exist nonempty sets $X \subseteq P$ and $Y \subseteq Q$ s.t.

- X has a t-expansion into Y, and
- no vertex in Y has a neighbour outside X.

Expansion Lemma (Matching Version)

$G(P, Q)$ – a bipartite graph; t – a positive integer;
ℓ – the size of a maximum matching in G s.t.

- $|Q| \geq t\ell$
- there are no isolated vertices in Q.

Then there exist nonempty sets $X \subseteq P$ and $Y \subseteq Q$ s.t.

- X has a t-expansion into Y, and
- no vertex in Y has a neighbour outside X.

X and Y can be found in poly-time.

Additional Property of X, Y

If there exists X, Y then we can find sets X', Y' s.t.

Additional Property of X, Y

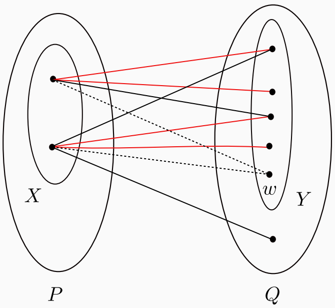
If there exists X, Y then we can find sets X', Y' s.t.

- X' has a t-expansion into Y',
- no vertex in Y' has a neighbour outside X', and
- $\exists$ a vertex $w \in Y'$ s.t. edges in t-expansion **do not** saturate w.

Additional Property of X, Y

If there exists X, Y then we can find sets X', Y' s.t.

- X' has a t-expansion into Y',
- no vertex in Y' has a neighbour outside X', and
- $\exists$ a vertex $w \in Y'$ s.t. edges in t-expansion **do not** saturate w.



$v \in K$ on the clique side of G s.t.

$v \in K$ on the clique side of G s.t.

- has more than k neighbours in the independent side I
- the size of maximum matching in $B(v)$ is at most k

$v \in K$ on the clique side of G s.t.

- has more than k neighbours in the independent side I
- the size of maximum matching in $B(v)$ is at most k

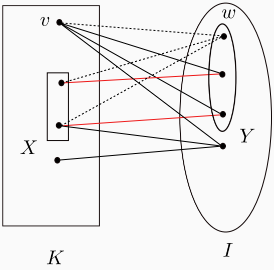
Then we can find X, Y and $w \in Y$ s.t.

$v \in K$ on the clique side of G s.t.

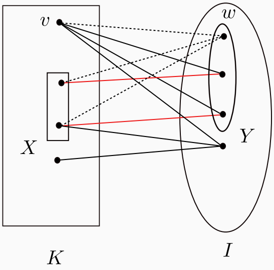
- has more than k neighbours in the independent side I
- the size of maximum matching in $B(v)$ is at most k

Then we can find X, Y and $w \in Y$ s.t.

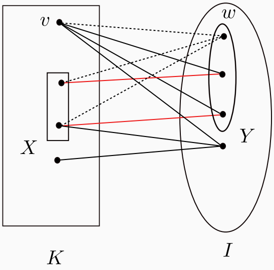
- there is a matching M between X and Y saturating X
- M does *not* saturate w, and
- $N_G(Y) = X \cup \{v\}$.



v
w
X
Y
K
I



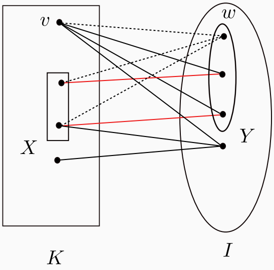
Matching edges are enough to store information about T-cycles containing edge vw.



Matching edges are enough to store information about T-cycles containing edge vw.

Case (A): Solution picks v.

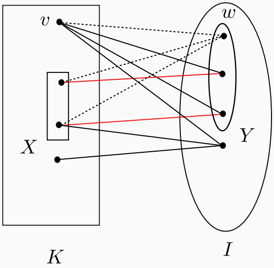
All T-cycles containing edge vw are killed.



Matching edges are enough to store information about T-cycles containing edge vw.

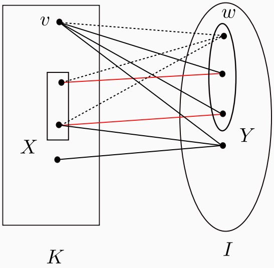
Case (B): Solution does not pick v and pick all vertices in X.

All T-cycles containing edge vw are killed.



Matching edges are enough to store information about T-cycles containing edge vw.

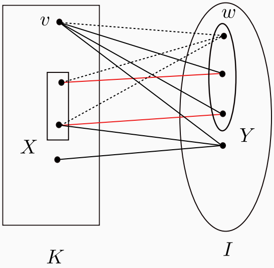
Case (C): Solution does not pick v and pick vertices in $X \cup Y$.



Matching edges are enough to store information about T-cycles containing edge vw.

Case (C): Solution does not pick v and pick vertices in $X \cup Y$.

By Expansion Lemma, Y is adjacent with only $X \cup \{v\}$.



Matching edges are enough to store information about T-cycles containing edge vw.

Case (C): Solution does not pick v and pick vertices in $X \cup Y$.

By Expansion Lemma, Y is adjacent with only $X \cup \{v\}$.

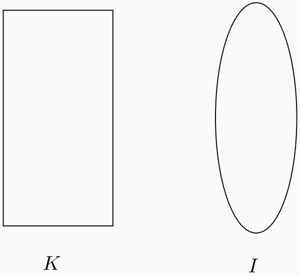
Modify solution: Remove Y; Add remaining vertices in X.

Kernel : Step 3 (Bounding the Size of the Clique Side)

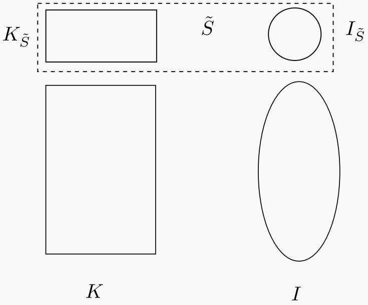
– A simple 3-factor approximation algorithm to compute $\tilde{S}$

- A simple 3-factor approximation algorithm to compute $\tilde{S}$
- If $|\tilde{S}| > 3k$ then return I_{NO}.

- A simple 3-factor approximation algorithm to compute $\tilde{S}$
- If $|\tilde{S}| > 3k$ then return I_{NO}.

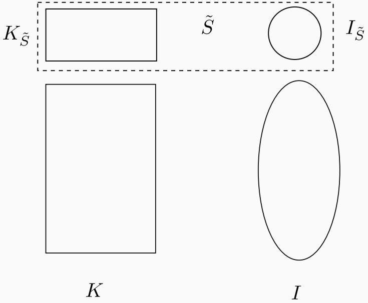


- A simple 3-factor approximation algorithm to compute $\tilde{S}$
- If $|\tilde{S}| > 3k$ then return I_{NO}.

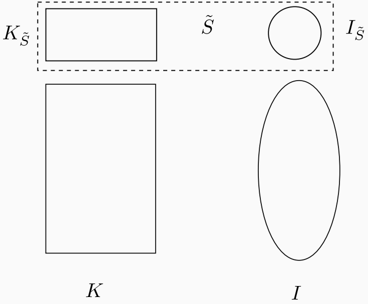


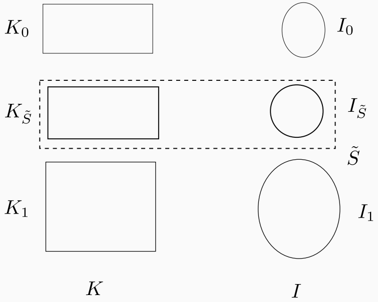
- $K_{\tilde{S}}$: set of clique-side vertices included in $\tilde{S}$.
  $K_{\tilde{S}} = K \cap \tilde{S}$.

- A simple 3-factor approximation algorithm to compute $\tilde{S}$
- If $|\tilde{S}| > 3k$ then return I_{NO}.

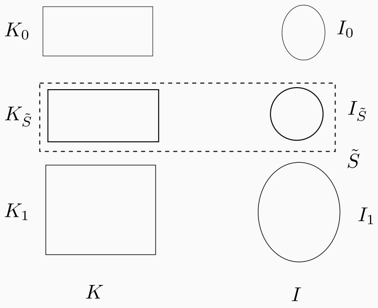


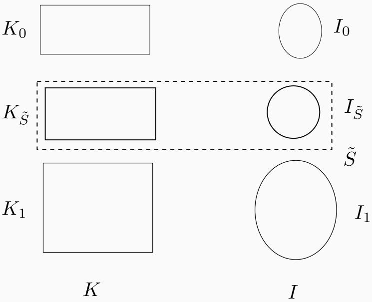
- $K_{\tilde{S}}$: set of clique-side vertices included in $\tilde{S}$.
  $K_{\tilde{S}} = K \cap \tilde{S}$.
- $I_{\tilde{S}}$: set of independent-side vertices included in $\tilde{S}$.
  $I_{\tilde{S}} = I \cap \tilde{S}$.


$K_{\tilde{S}}$
$\tilde{S}$
$I_{\tilde{S}}$
K
I

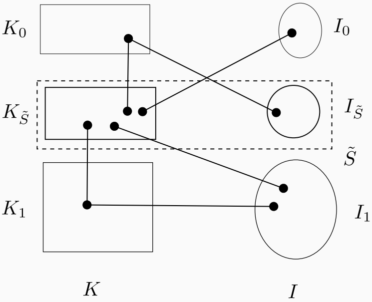


- K_0 : set of clique-side vertices not in $\tilde{S}$ whose neighbourhoods in the independent-side I are all contained in $I_{\tilde{S}}$.
  $K_0 = \{u \in (K \setminus K_{\tilde{S}})\ ;\ N(u) \cap I \subseteq I_{\tilde{S}}\};$
- I_0 : set of independent-side vertices not in $\tilde{S}$ whose neighbourhoods are all contained in $K_{\tilde{S}}$.
  $I_0 = \{v \in I \setminus I_{\tilde{S}}\ ;\ N(v) \subseteq K_{\tilde{S}}\}$

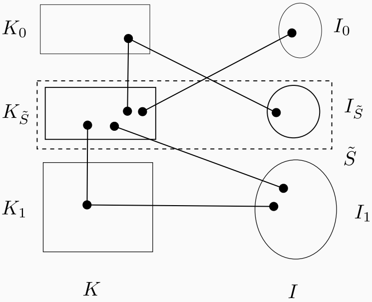

K_0
I_0
$K_{\tilde{S}}$
$I_{\tilde{S}}$
$\tilde{S}$
K_1
I_1
K
I



- K_1 : Remaining vertices in K.
  $K_1 = K \setminus (K_{\tilde{S}} \cup K_0)$.
- I_1 : Remaining vertices in I.
  $I_1 = I \setminus (I_{\tilde{S}} \cup I_0)$.

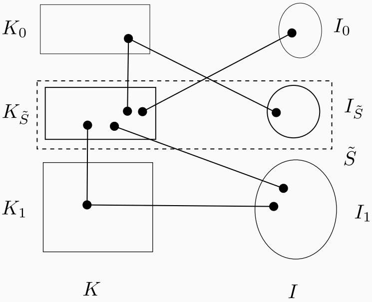


Some observations:



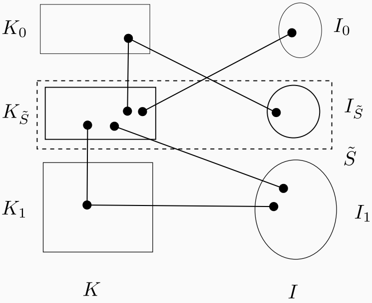
Some observations:

- $|K_{\tilde{S}}| \leq 2k$ and $|I_{\tilde{S}}| \leq k$.

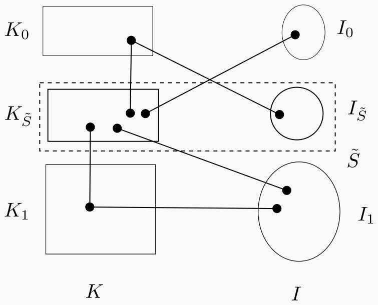


Some observations:

- $|K_{\tilde{S}}| \leq 2k$ and $|I_{\tilde{S}}| \leq k$.
- Each vertex in K_1 has (i) no neighbour in I_0 and (ii) at least one neighbour in I_1.

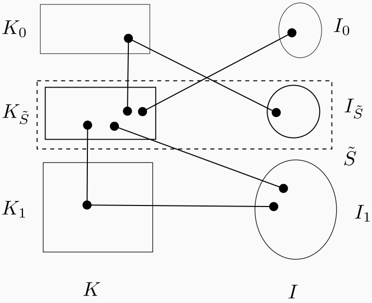


Some observations:



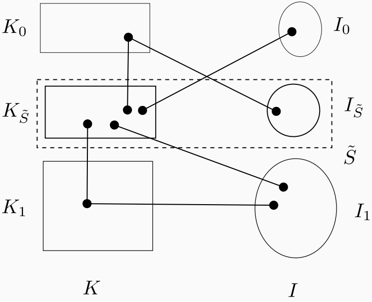
Some observations:

- Each vertex in I_1 has exactly one neighbour in K_1.



Some observations:

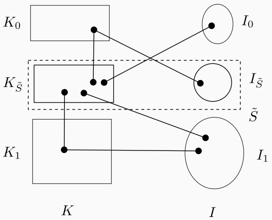
- Each vertex in I_1 has exactly one neighbour in K_1.
- The bipartite graph obtained from $G[K_1 \cup I_1]$ by deleting all the edges in $G[K_1]$ is a forest where each connected component is a star.

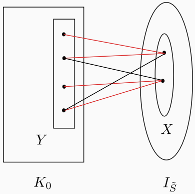


We bound

- K_0 using 2-expansion on graph across K_0 and $I_{\tilde{S}}$.
- K_1 using 2-expansion on graph across K_1 and $\tilde{S}$

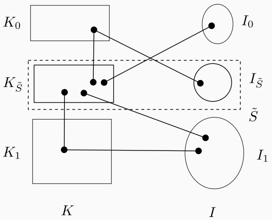
Bounding $|K_0|$

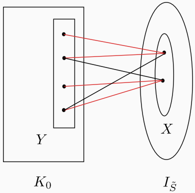




2-expansion across $I_{\tilde{S}}$ and K_0.

Bounding $|K_0|$

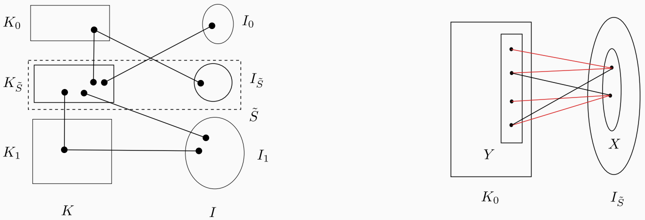




2-expansion across $I_{\tilde{S}}$ and K_0.

Solution intersects T-triangles in $X \cup Y$.

Bounding $|K_0|$

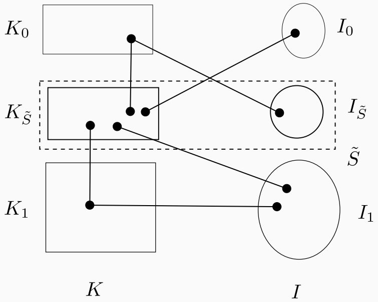


2-expansion across $I_{\tilde{S}}$ and K_0.

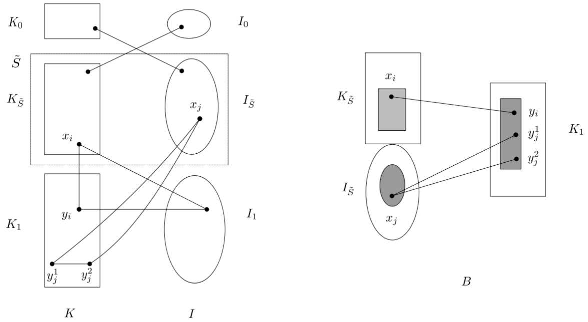
Solution intersects T-triangles in $X \cup Y$.

Since K_0 (and hence Y) interact with only $I_{\tilde{S}}$ (and hence only X), it is safe to pick all vertices in X to kill T-triangles in $X \cup Y$.

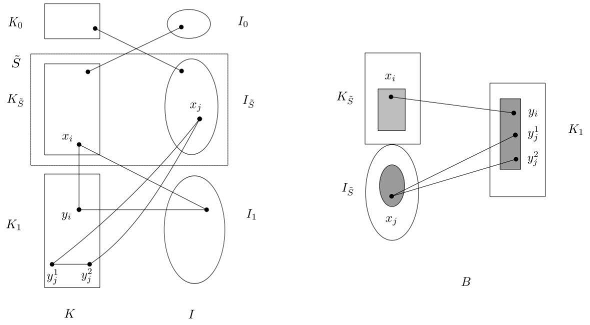
Bounding $|K_1|$



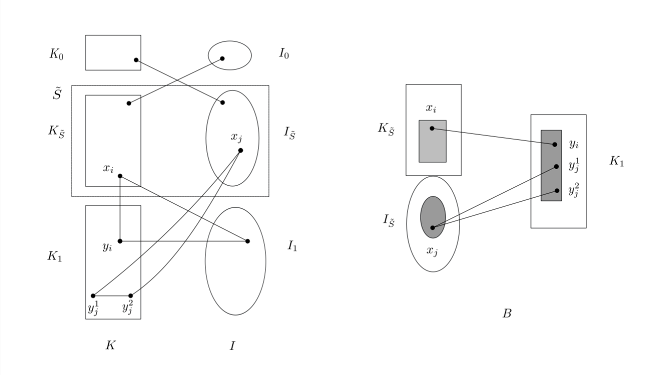
Construct auxilary bipartite graphs B across $\tilde{S}$ and K_1.



K_0
I_0
$\tilde{S}$
$K_{\tilde{S}}$
x_j
$I_{\tilde{S}}$
x_i
I_1
y_i
K_1
y_j^1
y_j^2
K
I
x_i
$K_{\tilde{S}}$
y_i
y_j^1
K_1
y_j^2
$I_{\tilde{S}}$
x_j
B

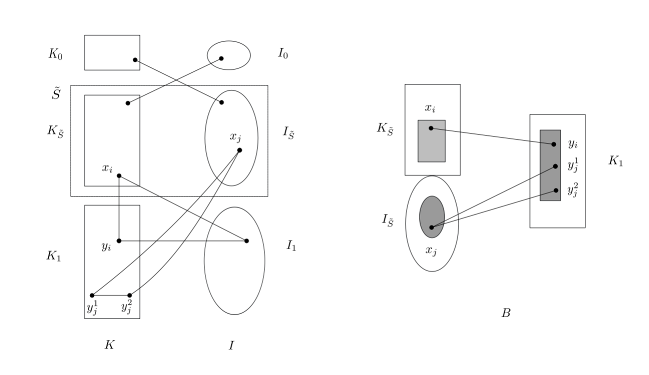


2-Expansion across $\tilde{S}$ and K.

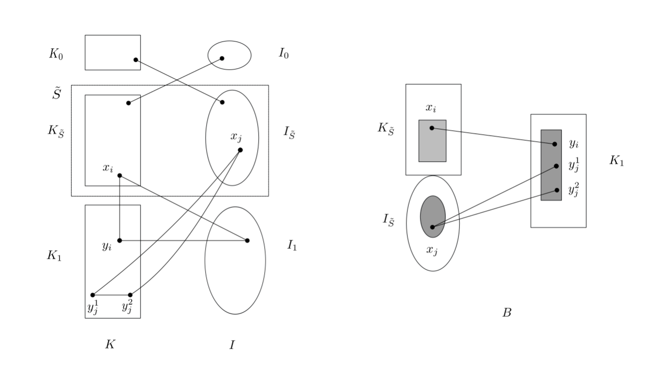


2-Expansion across $\tilde{S}$ and K.

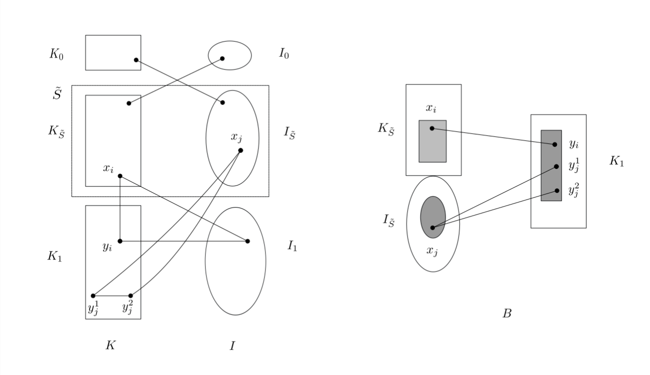
Shaded regions in $\tilde{S}$ and K_1 represent sets X, Y, respectively.



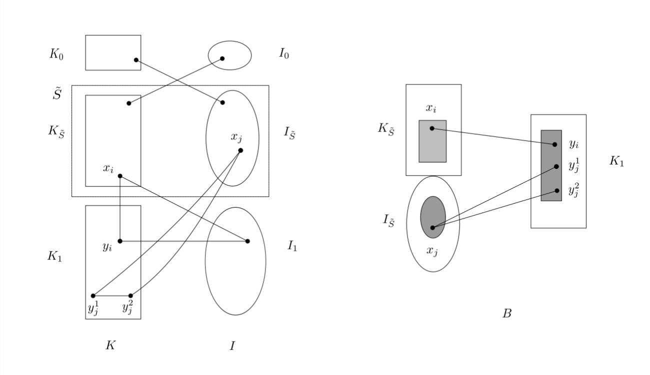
Each vertex $x_i \in (X \cap K_{\tilde{S}})$ is a part of T-triangle with an edge in M.



Each vertex $x_j \in (X \cap I_{\tilde{S}})$ is part of T-triangle with two edges in M.



We get $|X|$ many pairwise vertex-disjoint T-triangles.



We get $|X|$ many pairwise vertex-disjoint T-triangles.

Let S be an optimum solution. We modify this to exclude all vertices in Y.

Theorem

Subset-FVS in Split *admits a kernel with* $\mathcal{O}(k^2)$ *vertices and* $\mathcal{O}(k^2)$ *edges.*

Theorem

Subset-FVS in Split admits a kernel with $\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^2)$ edges.

- Step 1 ensures
  - No isolated vertex in G.
  - Every vertex in I is adjacent with some vertex in K

Theorem

Subset-FVS in Split *admits a kernel with* $\mathcal{O}(k^2)$ *vertices and* $\mathcal{O}(k^2)$ *edges.*

- Step 1 ensures
  - No isolated vertex in G.
  - Every vertex in I is adjacent with some vertex in K
- Step 2 ensures that any vertex in K is adjacent with at most $k+1$ vertices in I.

Theorem

Subset-FVS in Split admits a kernel with $\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^2)$ edges.

- Step 1 ensures
  - No isolated vertex in G.
  - Every vertex in I is adjacent with some vertex in K
- Step 2 ensures that any vertex in K is adjacent with at most $k+1$ vertices in I.
- Step 3 ensures that size of K is at most $10k$.

$\Rightarrow$ Size bound.

Theorem

Subset-FVS in Split *admits a kernel with* $\mathcal{O}(k^2)$ *vertices and* $\mathcal{O}(k^2)$ *edges.*

- Step 1 ensures
  - No isolated vertex in G.
  - Every vertex in I is adjacent with some vertex in K
- Step 2 ensures that any vertex in K is adjacent with at most $k+1$ vertices in I.
- Step 3 ensures that size of K is at most $10k$.

$\Rightarrow$ Size bound.

- We only use expansion lemma which can be applied in poly-time.

Conclusion and Open Questions

- A kernel of size $\mathcal{O}(k^2)$ with $\mathcal{O}(k)$ and $\mathcal{O}(k^2)$ vertices on the clique and independent set sides respectively.

Conclusion and Open Questions

- A kernel of size $\mathcal{O}(k^2)$ with $\mathcal{O}(k)$ and $\mathcal{O}(k^2)$ vertices on the clique and independent set sides respectively.
- Though size of kernel is optimum, can we bound the number of vertices on the independent side by $\mathcal{O}(k^{2-\epsilon})$?

Conclusion and Open Questions

- A kernel of size $\mathcal{O}(k^2)$ with $\mathcal{O}(k)$ and $\mathcal{O}(k^2)$ vertices on the clique and independent set sides respectively.
- Though size of kernel is optimum, can we bound the number of vertices on the independent side by $\mathcal{O}(k^{2-\epsilon})$?
- Can we obtain quadratic kernel for chordal graphs?

Conclusion and Open Questions

- A kernel of size $\mathcal{O}(k^2)$ with $\mathcal{O}(k)$ and $\mathcal{O}(k^2)$ vertices on the clique and independent set sides respectively.
- Though size of kernel is optimum, can we bound the number of vertices on the independent side by $\mathcal{O}(k^{2-\epsilon})$?
- Can we obtain quadratic kernel for chordal graphs?
- An algorithm running in $\mathcal{O}^*(2^k)$ to solve Subset FVS in Chordal.

Conclusion and Open Questions

- A kernel of size $\mathcal{O}(k^2)$ with $\mathcal{O}(k)$ and $\mathcal{O}(k^2)$ vertices on the clique and independent set sides respectively.
- Though size of kernel is optimum, can we bound the number of vertices on the independent side by $\mathcal{O}(k^{2-\epsilon})$?
- Can we obtain quadratic kernel for chordal graphs?
- An algorithm running in $\mathcal{O}^*(2^k)$ to solve SUBSET FVS IN CHORDAL.
- Under ETH, sub-exponential FPT algorithms for this problem are ruled out. Is it possible to obtain an algorithm with a smaller base in the running time?

Conclusion and Open Questions

- A kernel of size $\mathcal{O}(k^2)$ with $\mathcal{O}(k)$ and $\mathcal{O}(k^2)$ vertices on the clique and independent set sides respectively.
- Though size of kernel is optimum, can we bound the number of vertices on the independent side by $\mathcal{O}(k^{2-\epsilon})$?
- Can we obtain quadratic kernel for chordal graphs?
- An algorithm running in $\mathcal{O}^*(2^k)$ to solve SUBSET FVS IN CHORDAL.
- Under ETH, sub-exponential FPT algorithms for this problem are ruled out. Is it possible to obtain an algorithm with a smaller base in the running time?
- Interesting to investigate other implicit hitting set problems from graph theory and obtain better kernel and FPT results than the ones guaranteed by HITTING SET.

Thank you!

References i

Faisal N. Abu-Khzam.
A kernelization algorithm for d-hitting set.
J. Comput. Syst. Sci., 76(7):524–531, 2010.

Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk.
Subset feedback vertex set is fixed-parameter tractable.
SIAM Journal on Discrete Mathematics, 27(1):290–309, 2013.

Guy Even, Joseph Naor, and Leonid Zosin.
An 8-approximation algorithm for the subset feedback vertex set problem.
SIAM Journal on Computing, 30(4):1231–1252, 2000.

Fedor V Fomin, Pinar Heggernes, Dieter Kratsch, Charis Papadopoulos, and Yngve Villanger.
Enumerating minimal subset feedback vertex sets.
Algorithmica, 69(1):216–231, 2014.

Eva-Maria C Hols and Stefan Kratsch.
A randomized polynomial kernel for subset feedback vertex set.
Theory of Computing Systems, 62(1):63–92, 2018.

Ken-ichi Kawarabayashi and Yusuke Kobayashi.
Fixed-parameter tractability for the subset feedback set problem and the s-cycle packing problem.
Journal of Combinatorial Theory, Series B, 102(4):1020–1034, 2012.

Daniel Lokshtanov, MS Ramanujan, and Saket Saurabh.
Linear time parameterized algorithms for subset feedback vertex set.
ACM Transactions on Algorithms (TALG), 14(1):7, 2018.

Magnus Wahlström.
***Algorithms, measures and upper bounds for satisfiability and related problems*.**
PhD thesis, Department of Computer and Information Science, Linköpings universitet, 2007.

Magnus Wahlström.
Half-integrality, LP-branching and FPT algorithms.
In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1762–1781. SIAM, 2014.