U N I V E R S I T Y   O F   B E R G E N

Algorithms Research Group

On Sparsification for Computing Treewidth

Bart M. P. Jansen

September 4th 2013, IPEC 2013, Sophia Antipolis

Outline

Treewidth

Sparsification

Results

- Sparsification lower bound for TREEWIDTH
- Quadratic-vertex kernel upper bound for TREEWIDTH [VC]

Conclusion

Treewidth

- Measure of how “tree-like” a graph is

Treewidth

- Measure of how “tree-like” a graph is
  - Decompose a graph into a **tree decomposition** of small width to reveal its internal structure

Treewidth

- Measure of how “tree-like” a graph is
  - Decompose a graph into a **tree decomposition** of small width to reveal its internal structure
  - Dynamic programming solves many optimization problems when a tree decomposition is known

Treewidth

- Measure of how "tree-like" a graph is
  - Decompose a graph into a **tree decomposition** of small width to reveal its internal structure
  - Dynamic programming solves many optimization problems when a tree decomposition is known
  - First step: **find** good tree decomposition

Treewidth

- Measure of how “tree-like” a graph is
  - Decompose a graph into a **tree decomposition** of small width to reveal its internal structure
  - Dynamic programming solves many optimization problems when a tree decomposition is known
  - First step: **find** good tree decomposition

- TREEWIDTH

  Input: A graph G, an integer k

  Question: Is the treewidth of G at most k?

Treewidth

- Measure of how “tree-like” a graph is
  - Decompose a graph into a **tree decomposition** of small width to reveal its internal structure
  - Dynamic programming solves many optimization problems when a tree decomposition is known
  - First step: **find** good tree decomposition

- TREEWIDTH

  Input: A graph G, an integer k

  Question: Is the treewidth of G at most k?

  - NP-complete [Arnborg et al.’87]

Sparsification

- The task of making a problem instance less dense, without changing its answer

Sparsification

- The task of making a problem instance less dense, without changing its answer
- Density
  - .. of a CNF formula: ratio of clauses to variables
  - .. of a graph: ratio of edges to vertices

Sparsification

- The task of making a problem instance less dense, without changing its answer
- Density
  .. of a CNF formula: ratio of clauses to variables
  .. of a graph: ratio of edges to vertices
- Work on sparsification

Sparsification

- The task of making a problem instance less dense, without changing its answer
- Density

  .. of a CNF formula: ratio of clauses to variables

  .. of a graph: ratio of edges to vertices
- Work on sparsification
  - **Eppstein et al. @ J.ACM'97:**
    Sparsification to speed up dynamic graph algorithms

Sparsification

- The task of making a problem instance less dense, without changing its answer
- Density
  .. of a CNF formula: ratio of clauses to variables
  .. of a graph: ratio of edges to vertices
- Work on sparsification
  - **Eppstein et al. @ J.ACM'97:**
    Sparsification to speed up dynamic graph algorithms
  - **Impagliazzo et al. @ JCSS'01:**
    Subexponential-time *Sparsification Lemma* for SATISFIABILITY

Sparsification

- The task of making a problem instance less dense, without changing its answer
- Density
  .. of a CNF formula: ratio of clauses to variables
  .. of a graph: ratio of edges to vertices
- Work on sparsification
  - **Eppstein et al. @ J.ACM'97:** Sparsification to speed up dynamic graph algorithms
  - **Impagliazzo et al. @ JCSS'01:** Subexponential-time *Sparsification Lemma* for SATISFIABILITY
  - **Dell & van Melkebeek @ STOC'10:** No nontrivial polynomial-time sparsification for *d*-CNF-SAT

Sparsification for Computing Treewidth

- Given a graph G, we want to make it easier to find a good tree decomposition of G

Sparsification for Computing Treewidth

- Given a graph G, we want to make it easier to find a good tree decomposition of G
- Main idea:
  - quickly compute G’ which is “simpler” than G,
  - such that minimum-width decomposition of G’ easily leads to minimum-width decomposition of G

Sparsification for Computing Treewidth

- Given a graph G, we want to make it easier to find a good tree decomposition of G
- Main idea:
  - quickly compute G' which is "simpler" than G,
  - such that minimum-width decomposition of G' easily leads to minimum-width decomposition of G
- Possible ways to make G' provably simpler than G:
  - Upper bound on the density of G'
  - Upper bound on the vertex count of G', in terms of structural measures of G

Parameterized Complexity and Kernelization

- A parameterized problem is a subset $Q \subseteq \Sigma^* \times \mathbb{N}$
  - For an instance $(x,k) \in \Sigma^* \times \mathbb{N}$, we call k the **parameter**

Parameterized Complexity and Kernelization

- A parameterized problem is a subset $\mathbb{Q} \subseteq \Sigma^* \times \mathbb{N}$
  - For an instance (x,k) $\in \Sigma^* \times \mathbb{N}$, we call k the **parameter**
- Let $\mathbb{Q}$, $\mathbb{Q}'$ be parameterized problems and f: $\mathbb{N} \rightarrow \mathbb{N}$

Parameterized Complexity and Kernelization

- A parameterized problem is a subset $\mathbb{Q} \subseteq \Sigma^* \times \mathbb{N}$
  - For an instance (x,k) $\in \Sigma^* \times \mathbb{N}$, we call k the **parameter**
- Let $\mathbb{Q}$, $\mathbb{Q}$' be parameterized problems and f: $\mathbb{N} \rightarrow \mathbb{N}$
- A **generalized kernelization of $\mathbb{Q}$ into $\mathbb{Q}$' with size f** is

Parameterized Complexity and Kernelization

- A parameterized problem is a subset $\mathbb{Q} \subseteq \Sigma^* \times \mathbb{N}$
  - For an instance (x,k) $\in \Sigma^* \times \mathbb{N}$, we call k the **parameter**
- Let $\mathbb{Q}$, $\mathbb{Q}$' be parameterized problems and f: $\mathbb{N} \rightarrow \mathbb{N}$
- A **generalized kernelization of $\mathbb{Q}$ into $\mathbb{Q}$' with size f** is
  - an algorithm that takes (x,k) as input,

Parameterized Complexity and Kernelization

- A parameterized problem is a subset $\mathbb{Q} \subseteq \Sigma^* \times \mathbb{N}$
  - For an instance (x,k) $\in \Sigma^* \times \mathbb{N}$, we call k the **parameter**
- Let $\mathbb{Q}$, $\mathbb{Q}$’ be parameterized problems and f: $\mathbb{N} \rightarrow \mathbb{N}$
- A **generalized kernelization of $\mathbb{Q}$ into $\mathbb{Q}$’ with size f** is
  - an algorithm that takes (x,k) as input,
  - runs in poly(|x| + k) time,

Parameterized Complexity and Kernelization

- A parameterized problem is a subset $\mathbb{Q} \subseteq \Sigma^* \times \mathbb{N}$
  - For an instance (x,k) $\in \Sigma^* \times \mathbb{N}$, we call k the **parameter**
- Let $\mathbb{Q}$, $\mathbb{Q}$' be parameterized problems and f: $\mathbb{N} \rightarrow \mathbb{N}$
- A **generalized kernelization of $\mathbb{Q}$ into $\mathbb{Q}$' with size f** is
  - an algorithm that takes (x,k) as input,
  - runs in poly(|x| + k) time,
  - outputs (x',k') such that:

Parameterized Complexity and Kernelization

- A parameterized problem is a subset $\mathbb{Q} \subseteq \Sigma^* \times \mathbb{N}$
  - For an instance $(x,k) \in \Sigma^* \times \mathbb{N}$, we call k the **parameter**
- Let $\mathbb{Q}$, $\mathbb{Q}'$ be parameterized problems and f: $\mathbb{N} \rightarrow \mathbb{N}$
- A **generalized kernelization of $\mathbb{Q}$ into $\mathbb{Q}'$ with size f** is
  - an algorithm that takes (x,k) as input,
  - runs in poly(|x| + k) time,
  - outputs (x',k') such that:
    - $(x,k) \in \mathbb{Q}$ iff $(x',k') \in \mathbb{Q}'$

Parameterized Complexity and Kernelization

- A parameterized problem is a subset $\mathbb{Q} \subseteq \Sigma^* \times \mathbb{N}$
  - For an instance (x,k) $\in \Sigma^* \times \mathbb{N}$, we call k the **parameter**
- Let $\mathbb{Q}$, $\mathbb{Q}'$ be parameterized problems and f: $\mathbb{N} \rightarrow \mathbb{N}$
- A **generalized kernelization of $\mathbb{Q}$ into $\mathbb{Q}'$ with size f** is
  - an algorithm that takes (x,k) as input,
  - runs in poly(|x| + k) time,
  - outputs (x',k') such that:
    - (x,k) $\in \mathbb{Q}$ iff (x',k') $\in \mathbb{Q}'$
    - |x'|, k' $\leq$ f(k)

Parameterized Complexity and Kernelization

- A parameterized problem is a subset $\mathbb{Q} \subseteq \Sigma^* \times \mathbb{N}$
  - For an instance $(x,k) \in \Sigma^* \times \mathbb{N}$, we call k the **parameter**
- Let $\mathbb{Q}$, $\mathbb{Q}'$ be parameterized problems and f: $\mathbb{N} \rightarrow \mathbb{N}$
- A **generalized kernelization of $\mathbb{Q}$ into $\mathbb{Q}'$ with size f** is
  - an algorithm that takes (x,k) as input,
  - runs in poly(|x| + k) time,
  - outputs (x',k') such that:
    - $(x,k) \in \mathbb{Q}$ iff $(x',k') \in \mathbb{Q}'$
    - $|x'|, k' \leq f(k)$

} Poly-time mapping from $\mathbb{Q}$ to $\mathbb{Q}'$

– preserves the answer

– f(k)-size output bound

Parameterized Complexity and Kernelization

- A parameterized problem is a subset $\mathbb{Q} \subseteq \Sigma^* \times \mathbb{N}$
  - For an instance $(x,k) \in \Sigma^* \times \mathbb{N}$, we call k the **parameter**
- Let $\mathbb{Q}$, $\mathbb{Q}'$ be parameterized problems and f: $\mathbb{N} \rightarrow \mathbb{N}$
- A **generalized kernelization of $\mathbb{Q}$ into $\mathbb{Q}'$ with size f** is
  - an algorithm that takes (x,k) as input,
  - runs in poly(|x| + k) time,
  - outputs (x',k') such that:
    - $(x,k) \in \mathbb{Q}$ iff $(x',k') \in \mathbb{Q}'$
    - $|x'|, k' \leq f(k)$

Poly-time mapping from $\mathbb{Q}$ to $\mathbb{Q}'$

– preserves the answer

– f(k)-size output bound

- It is a kernelization (or **kernel**) if $\mathbb{Q} = \mathbb{Q}'$

Sparsification Analysis using Kernelization

- Based on the parameterization by vertex count:
  - *n*-TREEWIDTH

    Input: $n \in \mathbb{N}$, an n-vertex graph G, an integer k

    Parameter: n

    Question: Is the treewidth of G at most k?

Sparsification Analysis using Kernelization

- Based on the parameterization by vertex count:
  - *n*-TREEWIDTH

    Input: $n \in \mathbb{N}$, an n-vertex graph G, an integer k

    Parameter: n

    Question: Is the treewidth of G at most k?

- An instance (G,k,n) can be encoded in $\mathcal{O}(n^2)$ bits

Sparsification Analysis using Kernelization

- Based on the parameterization by vertex count:
  - *n*-TREEWIDTH
    Input: $n \in \mathbb{N}$, an n-vertex graph G, an integer k
    Parameter: n
    Question: Is the treewidth of G at most k?
- An instance (G,k,n) can be encoded in $\mathcal{O}(n^2)$ bits
- Most relaxed form of polynomial-time sparsification:
  - generalized kernel for *n*-TREEWIDTH of size $\mathcal{O}(n^{2-\varepsilon})$ for $\varepsilon > 0$

Sparsification Analysis using Kernelization

- Based on the parameterization by vertex count:
  - *n*-TREEWIDTH
    Input: $n \in \mathbb{N}$, an n-vertex graph G, an integer k
    Parameter: n
    Question: Is the treewidth of G at most k?
- An instance (G,k,n) can be encoded in $\mathcal{O}(n^2)$ bits
- Most relaxed form of polynomial-time sparsification:
  - generalized kernel for *n*-TREEWIDTH of size $\mathcal{O}(n^{2-\varepsilon})$ for $\varepsilon > 0$

Theorem. *n*-TREEWIDTH does not have a generalized kernel of bitsize $\mathcal{O}(n^{2-\varepsilon})$, for any $\varepsilon > 0$, unless NP ⊆ coNP/poly

Proof Technique

- Proof using **cross-composition of bounded cost**
  - Introduced in the journal version of the paper on cross-composition [Bodlaender, J, Kratsch ’12]
  - Easier front-end to the complementary witness lemma of Dell & van Melkebeek [STOC’10]

Proof Technique

Corollary [Bodlaender et al.'12].
If there is a polynomial-time algorithm that:
- composes the OR of t^2 similar size-s instances of an NP-hard problem,
- into an instance (G^*, n^*, k^*) of *n*-TREEWIDTH with $n^* \in \mathcal{O}(t \cdot s^{\mathcal{O}(1)})$,

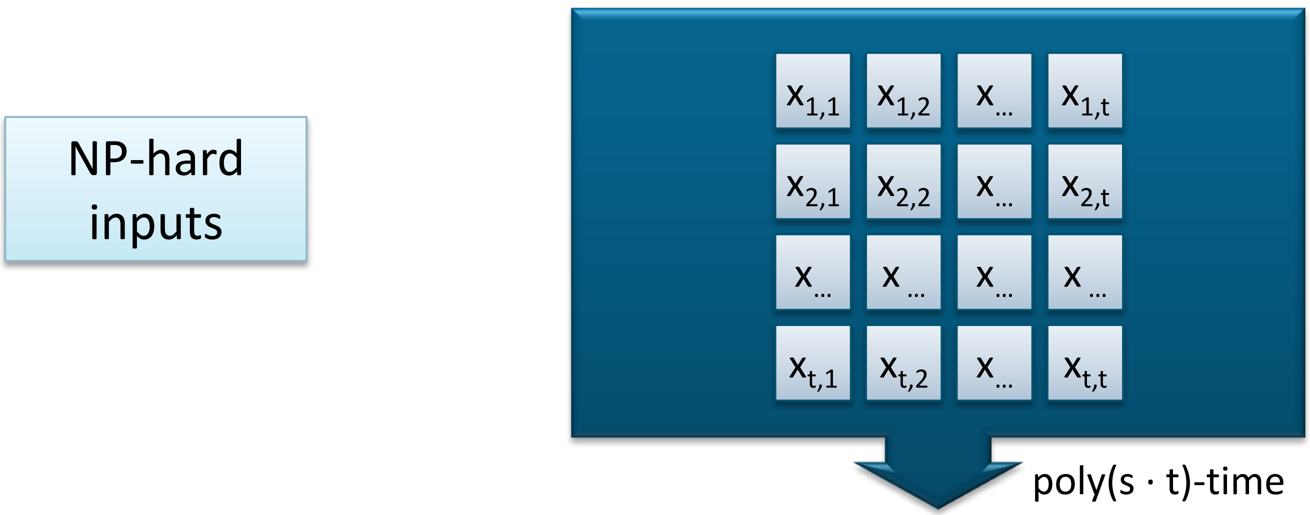
then *n*-TREEWIDTH does not have a generalized kernel of bitsize $\mathcal{O}(n^{2-\varepsilon})$, for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly

Proof Technique

Corollary [Bodlaender et al.'12].
If there is a polynomial-time algorithm that:

- composes the OR of t^2 similar size-s instances of an NP-hard problem,
- into an instance (G^*, n^*, k^*) of *n*-TREEWIDTH with $n^* \in \mathcal{O}(t \cdot s^{\mathcal{O}(1)})$,

then *n*-TREEWIDTH does not have a generalized kernel of bitsize $\mathcal{O}(n^{2-\varepsilon})$, for any $\varepsilon > 0$, unless NP ⊆ coNP/poly

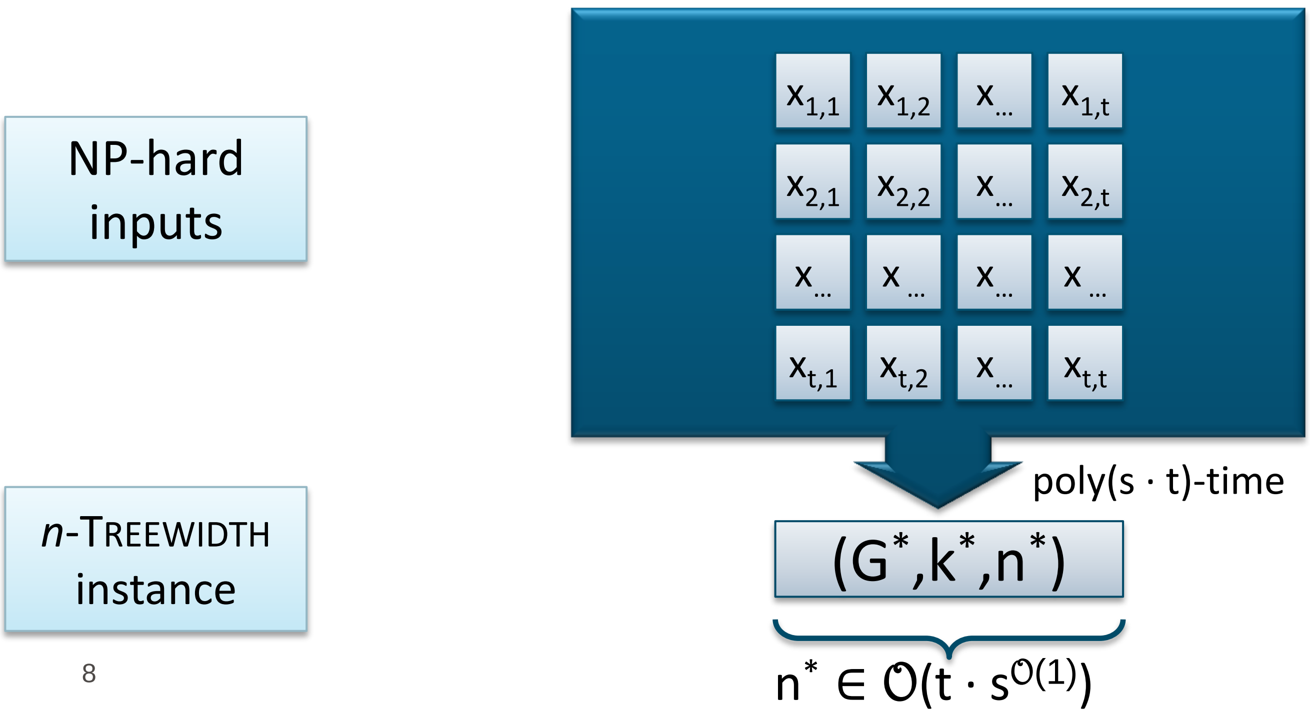


Proof Technique

Corollary [Bodlaender et al.'12].
If there is a polynomial-time algorithm that:

- composes the OR of t^2 similar size-s instances of an NP-hard problem,
- into an instance (G^*, n^*, k^*) of *n*-TREEWIDTH with $n^* \in \mathcal{O}(t \cdot s^{\mathcal{O}(1)})$,

then *n*-TREEWIDTH does not have a generalized kernel of bitsize $\mathcal{O}(n^{2-\varepsilon})$, for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly



Proof Strategy

- Convenient source problem

Proof Strategy

- Convenient source problem
  - COBIPARTITE GRAPH ELIMINATION
  - "Given a restricted type of cobipartite graph, does it have treewidth at most k?"
  - Based on Arnborg et al.'s NP-completeness proof for TREEWIDTH

Proof Strategy

- Convenient source problem
  - COBIPARTITE GRAPH ELIMINATION
  - "Given a restricted type of cobipartite graph, does it have treewidth at most k?"
  - Based on Arnborg et al.'s NP-completeness proof for TREEWIDTH
- Embed t^2 instances into a t × 2 table (Dell & Marx [SODA'12])

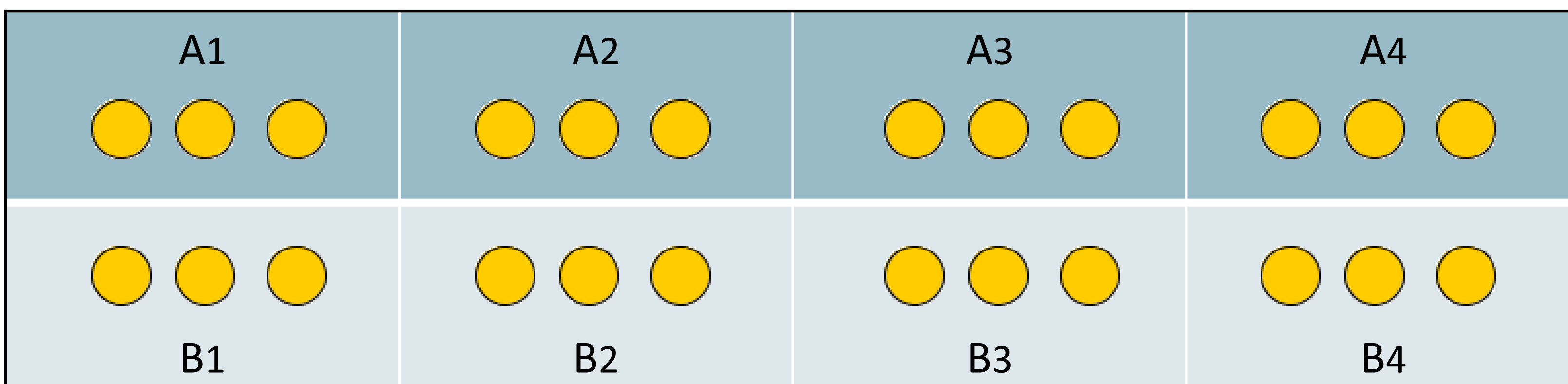
Proof Strategy

- Convenient source problem
  - COBIPARTITE GRAPH ELIMINATION
  - “Given a restricted type of cobipartite graph, does it have treewidth at most k?”
  - Based on Arnborg et al.’s NP-completeness proof for TREEWIDTH
- Embed t^2 instances into a $t \times 2$ table (Dell & Marx [SODA’12])

A1	A2	A3	A4
B1	B2	B3	B4

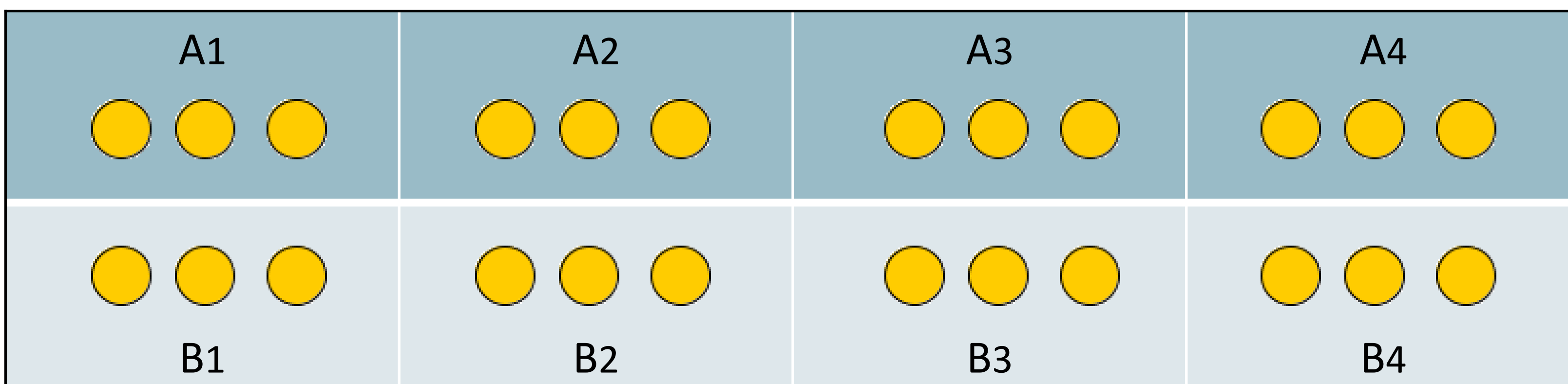
Proof Strategy

- Convenient source problem
  - COBIPARTITE GRAPH ELIMINATION
  - “Given a restricted type of cobipartite graph, does it have treewidth at most k?”
  - Based on Arnborg et al.’s NP-completeness proof for TREEWIDTH
- Embed t^2 instances into a $t \times 2$ table (Dell & Marx [SODA’12])



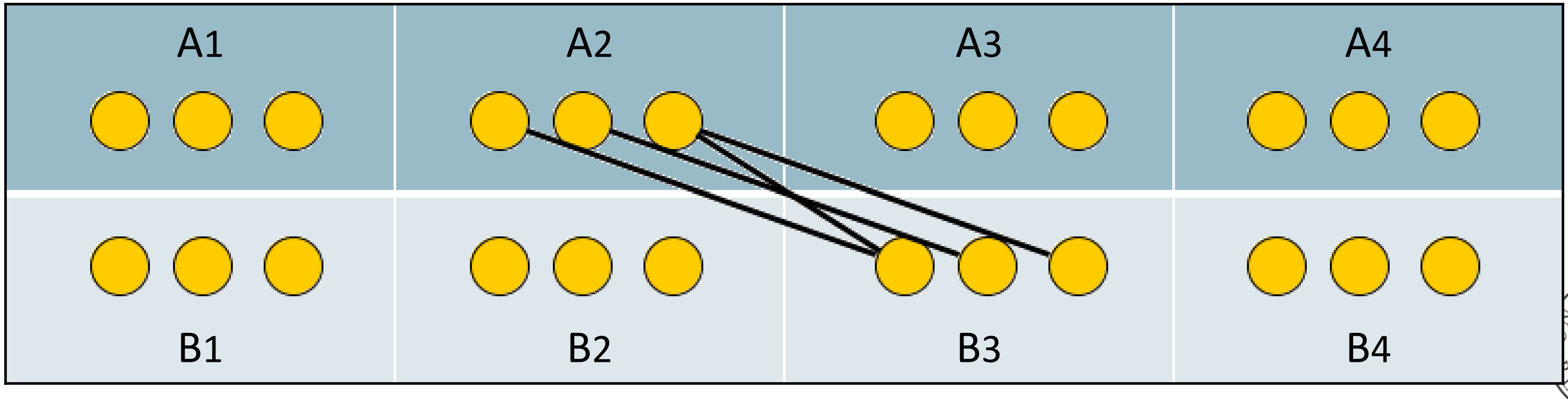
Proof Strategy

- Convenient source problem
  - COBIPARTITE GRAPH ELIMINATION
  - "Given a restricted type of cobipartite graph, does it have treewidth at most k?"
  - Based on Arnborg et al.'s NP-completeness proof for TREEWIDTH
- Embed t^2 instances into a $t \times 2$ table (Dell & Marx [SODA'12])
  - Turn rows into cliques to get a cobipartite graph



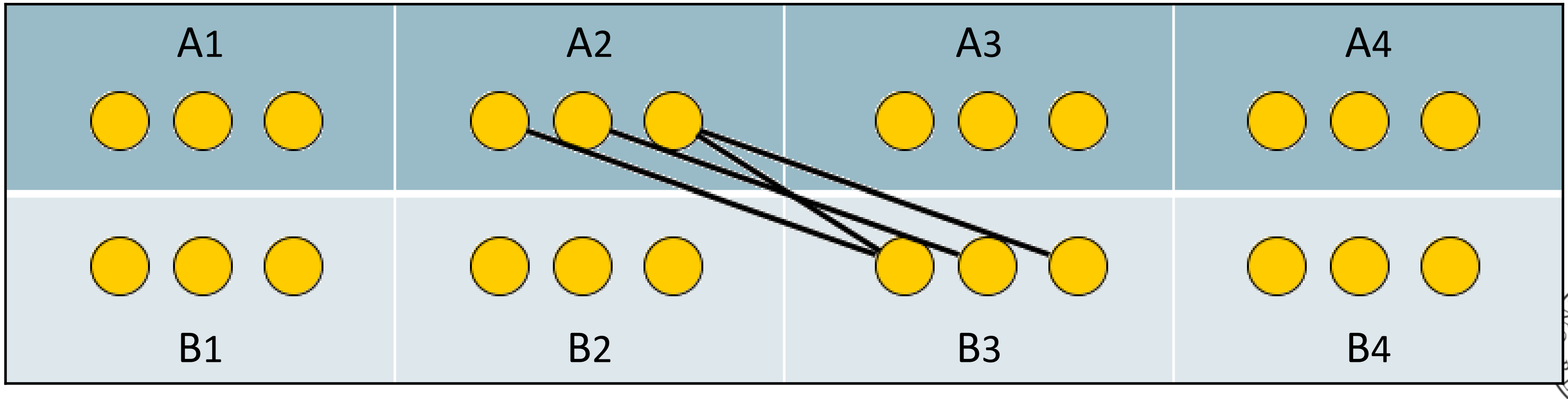
Proof Strategy

- Convenient source problem
  - COBIPARTITE GRAPH ELIMINATION
  - “Given a restricted type of cobipartite graph, does it have treewidth at most k?”
  - Based on Arnborg et al.’s NP-completeness proof for TREEWIDTH
- Embed t^2 instances into a t × 2 table (Dell & Marx [SODA’12])
  - Turn rows into cliques to get a cobipartite graph



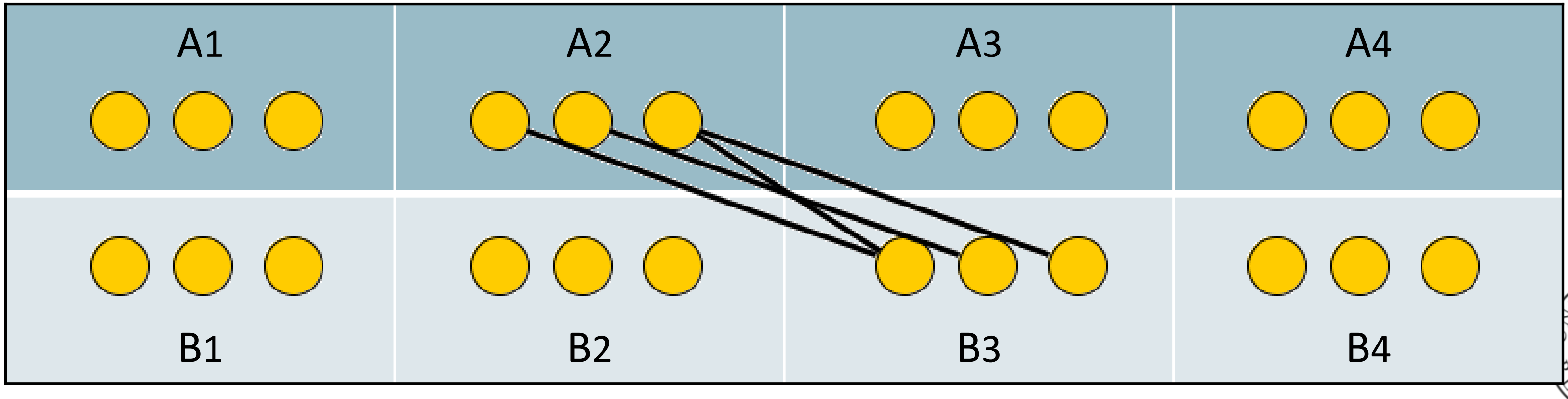
Proof Strategy

- Convenient source problem
  - COBIPARTITE GRAPH ELIMINATION
  - "Given a restricted type of cobipartite graph, does it have treewidth at most k?"
  - Based on Arnborg et al.'s NP-completeness proof for TREEWIDTH
- Embed t^2 instances into a $t \times 2$ table (Dell & Marx [SODA'12])
  - Turn rows into cliques to get a cobipartite graph
- Gadgets enforce an OR-gate through the cobipartite graph



Proof Strategy

- Convenient source problem
  - COBIPARTITE GRAPH ELIMINATION
  - "Given a restricted type of cobipartite graph, does it have treewidth at most k?"
  - Based on Arnborg et al.'s NP-completeness proof for TREEWIDTH
- Embed t^2 instances into a $t \times 2$ table (Dell & Marx [SODA'12])
  - Turn rows into cliques to get a cobipartite graph
- Gadgets enforce an OR-gate through the cobipartite graph
  - 10 pages of proof to make it work



Consequences of the Lower Bound

Consequences of the Lower Bound

Corollary 1. For every $\varepsilon > 0$ and every parameter Π that does not exceed the vertex count, TREEWIDTH $[\Pi]$ does not have a kernel of bitsize $\mathcal{O}(k^{2-\varepsilon})$, unless NP $\subseteq$ coNP/poly

Consequences of the Lower Bound

Corollary 1. For every ε > 0 and every parameter Π that does not exceed the vertex count, TREEWIDTH [Π] does not have a kernel of bitsize $\mathcal{O}(k^{2-\varepsilon})$, unless NP ⊆ coNP/poly

- Applies to parameterizations by Vertex Cover, Feedback Vertex Set, Vertex-Deletion Distance to …

Consequences of the Lower Bound

Corollary 1. For every $\varepsilon > 0$ and every parameter Π that does not exceed the vertex count, TREEWIDTH $[\Pi]$ does not have a kernel of bitsize $\mathcal{O}(k^{2-\varepsilon})$, unless NP $\subseteq$ coNP/poly

- Applies to parameterizations by Vertex Cover, Feedback Vertex Set, Vertex-Deletion Distance to …
- The constructed graph is cobipartite
  - For cobipartite graphs, treewidth equals pathwidth

Consequences of the Lower Bound

Corollary 1. For every ε > 0 and every parameter Π that does not exceed the vertex count, TREEWIDTH [Π] does not have a kernel of bitsize $\mathcal{O}(k^{2-\varepsilon})$, unless NP ⊆ coNP/poly

- Applies to parameterizations by Vertex Cover, Feedback Vertex Set, Vertex-Deletion Distance to …
- The constructed graph is cobipartite
  - For cobipartite graphs, treewidth equals pathwidth

Corollary 2. *n*-PATHWIDTH does not have a generalized kernel of bitsize $\mathcal{O}(n^{2-\varepsilon})$, for any ε > 0, unless NP ⊆ coNP/poly

Treewidth Parameterized by Vertex Cover

- TREEWIDTH [VC]

  Input: A graph G, vertex cover X of G, and an integer k

  Parameter: |X|

  Question: Is the treewidth of G at most k?

Treewidth Parameterized by Vertex Cover

- TREEWIDTH [VC]
  Input: A graph G, vertex cover X of G, and an integer k
  Parameter: |X|
  Question: Is the treewidth of G at most k?

- Lower bound implies:
  - No kernel with **bitsize** $O(|X|^{2-\varepsilon})$ unless NP ⊆ coNP/poly

Treewidth Parameterized by Vertex Cover

- TREEWIDTH [VC]
  Input: A graph G, vertex cover X of G, and an integer k
  Parameter: |X|
  Question: Is the treewidth of G at most k?

- Lower bound implies:
  - No kernel with **bitsize** $\mathcal{O}(|X|^{2-\varepsilon})$ unless NP ⊆ coNP/poly

- Previous-best was a kernel with $\mathcal{O}(|X|^3)$ **vertices**
  - Bodlaender, J, Kratsch [ICALP'11]

Treewidth Parameterized by Vertex Cover

- TREEWIDTH [VC]
  Input: A graph G, vertex cover X of G, and an integer k
  Parameter: |X|
  Question: Is the treewidth of G at most k?
- Lower bound implies:
  - No kernel with **bitsize** $\mathcal{O}(|X|^{2-\varepsilon})$ unless NP ⊆ coNP/poly
- Previous-best was a kernel with $\mathcal{O}(|X|^3)$ **vertices**
  - Bodlaender, J, Kratsch [ICALP'11]
- We improve this to $|X|^2$ **vertices**

Treewidth-Invariant Sets

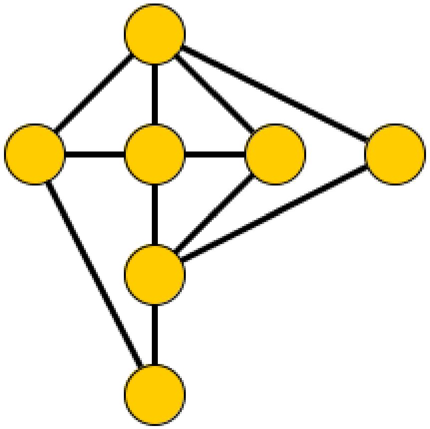
- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth

Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T

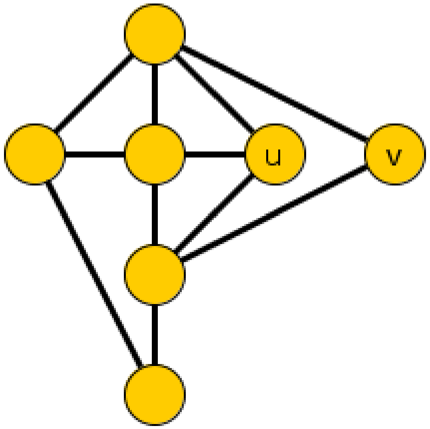
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T



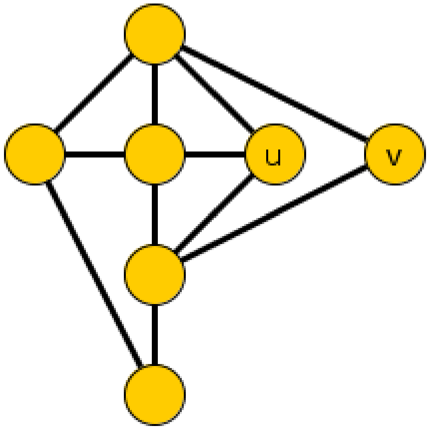
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T



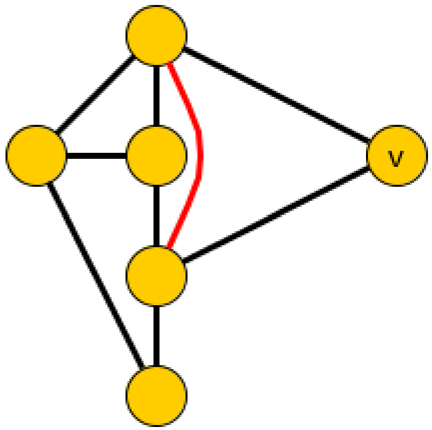
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)



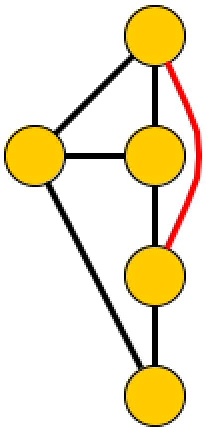
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)



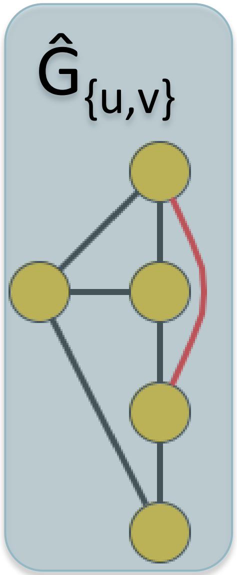
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)



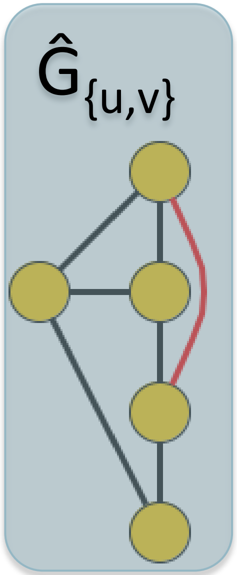
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)



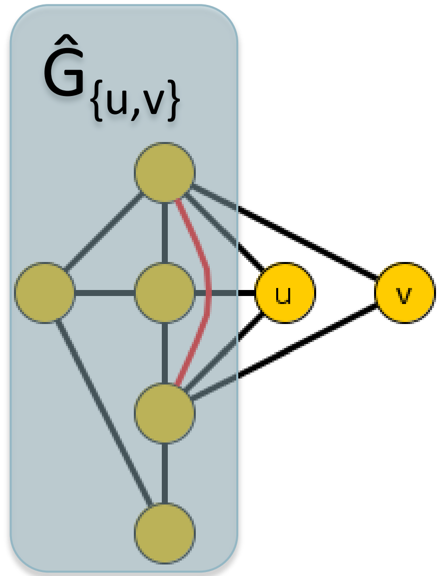
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of G – {z} for every z ∈ T, then T is a **treewidth-invariant set** in G



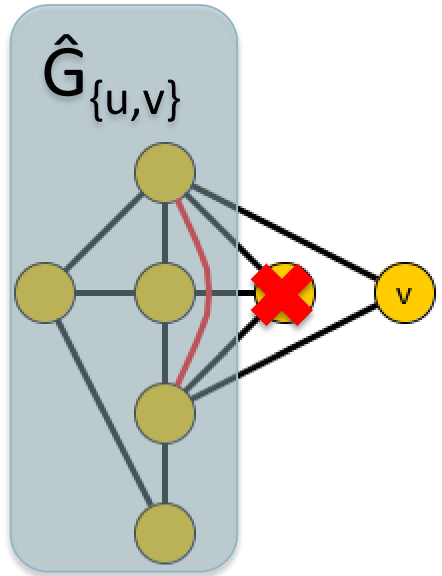
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of G – {z} for every z ∈ T, then T is a **treewidth-invariant set** in G



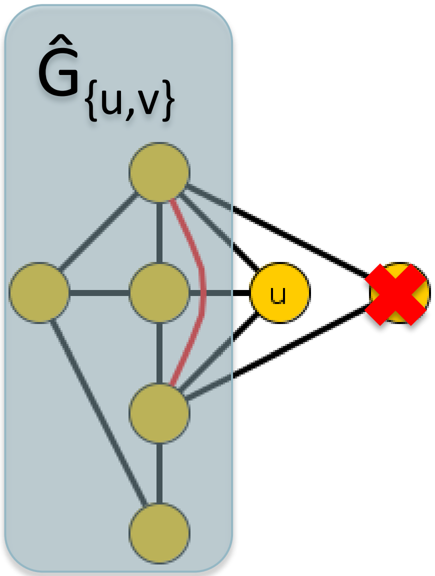
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of G – {z} for every z ∈ T, then T is a **treewidth-invariant set** in G



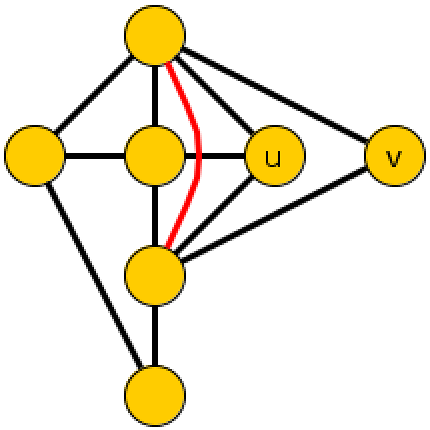
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of $G - \{z\}$ for every $z \in T$, then T is a **treewidth-invariant set** in G



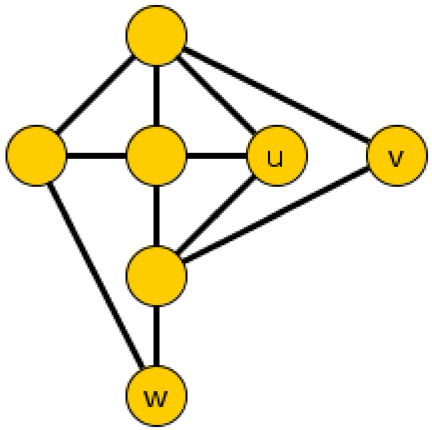
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of $G - \{z\}$ for every $z \in T$, then T is a **treewidth-invariant set** in G



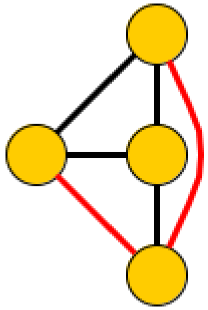
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of G – {z} for every z ∈ T, then T is a **treewidth-invariant set** in G



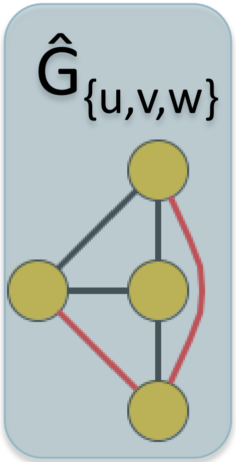
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of G – {z} for every z ∈ T, then T is a **treewidth-invariant set** in G



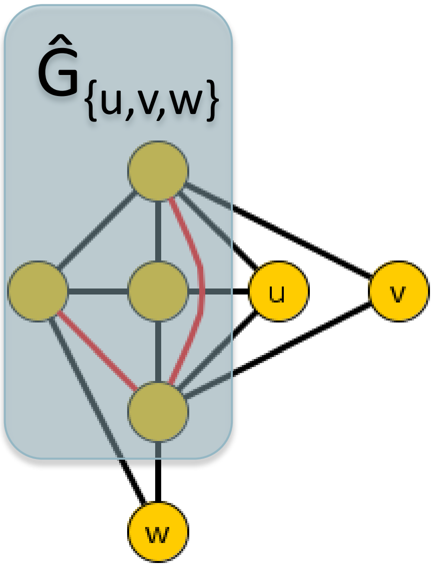
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of G – {z} for every z ∈ T, then T is a **treewidth-invariant set** in G



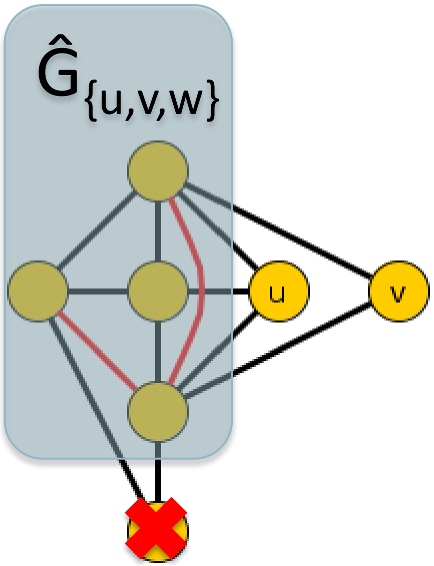
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of G – {z} for every z ∈ T, then T is a **treewidth-invariant set** in G



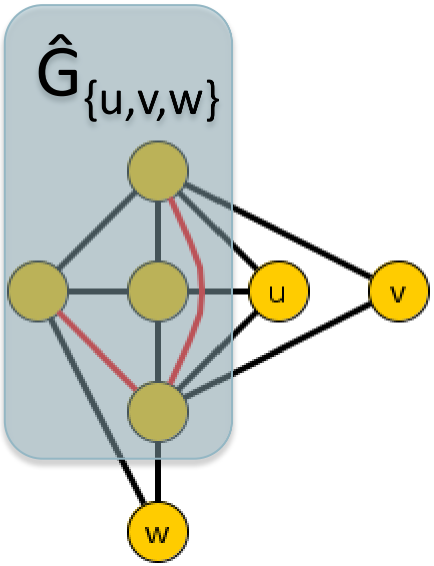
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of G – {z} for every z ∈ T, then T is a **treewidth-invariant set** in G



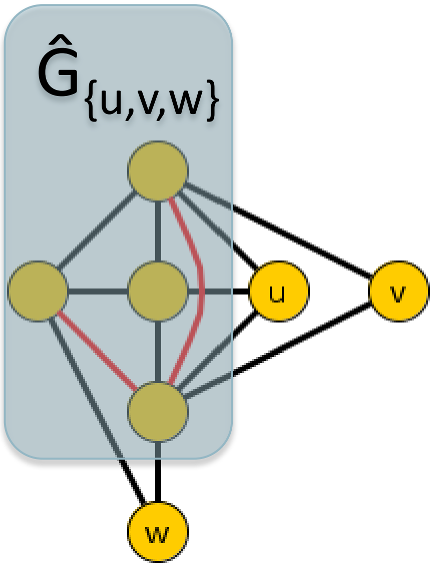
Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of G – {z} for every z ∈ T, then T is a **treewidth-invariant set** in G
- Let $\Delta(T) := \max_{v \in T} \deg(v)$



Treewidth-Invariant Sets

- Kernel is based on **treewidth-invariant sets**
  - Vertex set whose elimination has a predictable effect on treewidth
- Consider a graph G with an independent set T
  - Let $\hat{G}_T$ be the result of eliminating T from G, one vertex at a time (order does not matter)
  - If $\hat{G}_T$ is a minor of G – {z} for every z ∈ T, then T is a **treewidth-invariant set** in G
- Let $\Delta(T) := \max_{v \in T} \deg(v)$



Lemma. If T is a treewidth-invariant set in G, then $\textsc{tw}(G) = \max\{\textsc{tw}(\hat{G}_T), \Delta(T)\}$

Reduction Based on Treewidth-Invariant Sets

- Consider an instance (G,k) of TREEWIDTH with a treewidth-invariant set T:

Reduction Based on Treewidth-Invariant Sets

- Consider an instance (G,k) of TREEWIDTH with a treewidth-invariant set T:
  - If $\Delta(T) \geq k + 1$: output NO

Reduction Based on Treewidth-Invariant Sets

- Consider an instance (G,k) of TREEWIDTH with a treewidth-invariant set T:
  - If $\Delta(T) \geq k + 1$: output NO
  - Else reduce to $\hat{G}_T$

Reduction Based on Treewidth-Invariant Sets

- Consider an instance (G,k) of TREEWIDTH with a treewidth-invariant set T:
  - If $\Delta(T) \geq k + 1$: output NO
  - Else reduce to $\hat{G}_T$
- Invariance lemma shows that $\mathrm{TW}(G) \leq k$ iff $\mathrm{TW}(\hat{G}_T) \leq k$, reduction is safe

Reduction Based on Treewidth-Invariant Sets

- Consider an instance (G,k) of TREEWIDTH with a treewidth-invariant set T:
  - If $\Delta(T) \geq k + 1$: output NO
  - Else reduce to $\hat{G}_T$
- Invariance lemma shows that $\text{TW}(G) \leq k$ iff $\text{TW}(\hat{G}_T) \leq k$, reduction is safe
- Remaining issue: how to find treewidth-invariant sets?
  - Seems hard in general
  - NP-complete to test if a given set is treewidth-invariant!

Reduction Based on Treewidth-Invariant Sets

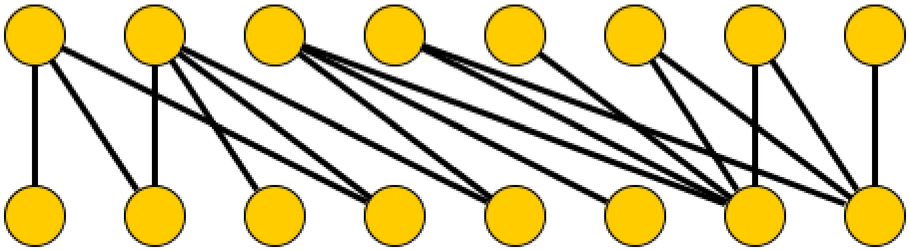
- Consider an instance (G,k) of TREEWIDTH with a treewidth-invariant set T:
  - If $\Delta(T) \geq k + 1$: output NO
  - Else reduce to $\hat{G}_T$
- Invariance lemm… ov… $v(\hat{G}_T) \leq k$, reduction is safe
- Remaining is… …eewidth-invariant sets?
  - Seems …ard in gene…
  - NP-complete to t… …et is treewidth-invariant!

q-Expansion Lemma

- Let H be a bipartite graph with partite sets A and B, and $q \in \mathbb{N}$

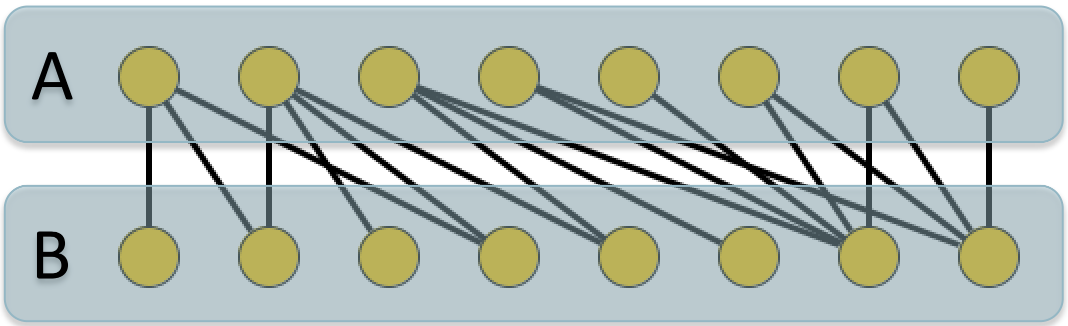
q-Expansion Lemma

- Let H be a bipartite graph with partite sets A and B, and $q \in \mathbb{N}$



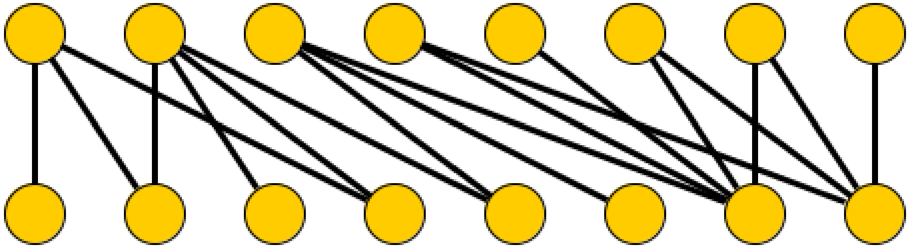
q-Expansion Lemma

- Let H be a bipartite graph with partite sets A and B, and $q \in \mathbb{N}$



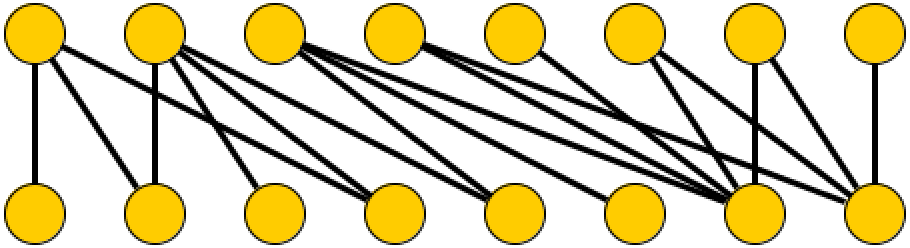
q-Expansion Lemma

- Let H be a bipartite graph with partite sets A and B, and $q \in \mathbb{N}$



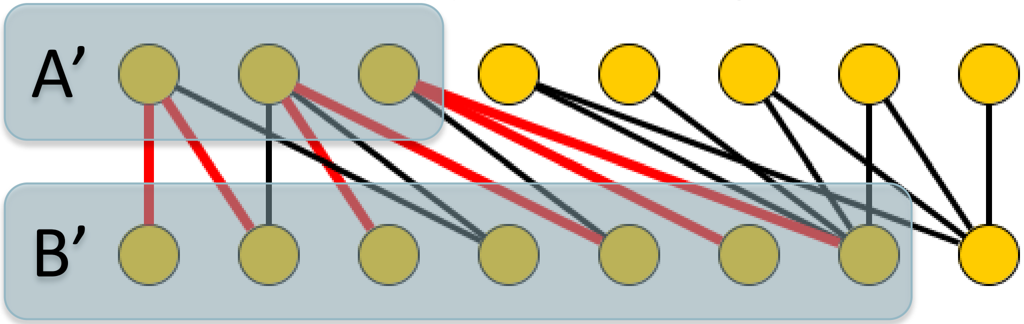
q-Expansion Lemma

- Let H be a bipartite graph with partite sets A and B, and q ∈ ℕ
- A subset A' ⊆ A is **saturated by q-stars into B'** if we can assign to each v ∈ A' a set of q private neighbors from B'



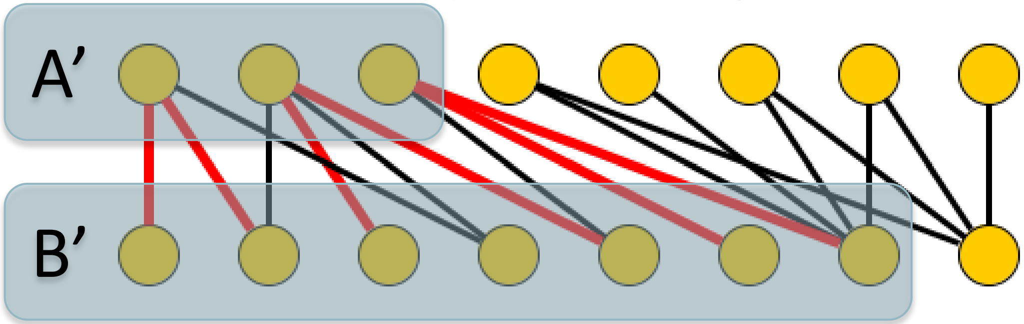
q-Expansion Lemma

- Let H be a bipartite graph with partite sets A and B, and q ∈ ℕ
- A subset A' ⊆ A is **saturated by q-stars into B'** if we can assign to each v ∈ A' a set of q private neighbors from B'



q-Expansion Lemma

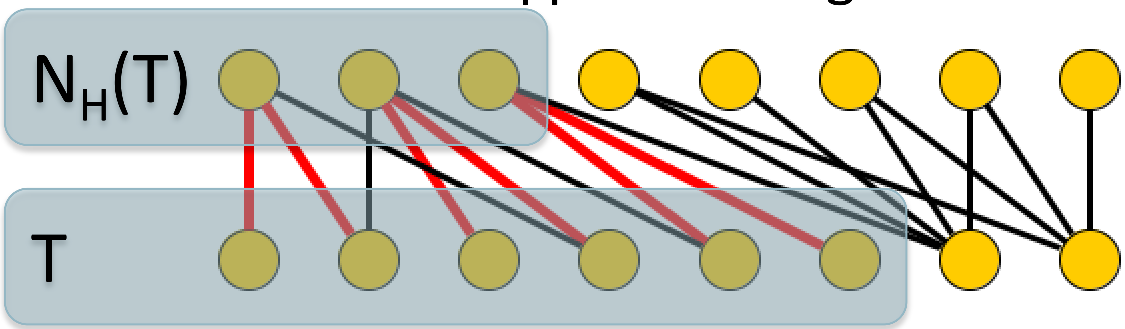
- Let H be a bipartite graph with partite sets A and B, and $q \in \mathbb{N}$
- A subset $A' \subseteq A$ is **saturated by q-stars into B'** if we can assign to each $v \in A'$ a set of q private neighbors from B'



q-Expansion Lemma [Fomin et al.@STACS'11]
Let m be the size of a maximum matching H. If $|B| > m \cdot q$, then there is a nonempty set $T \subseteq B$ such that $N_H(T)$ is saturated by q-stars into T. It can be found in polynomial time.

q-Expansion Lemma

- Let H be a bipartite graph with partite sets A and B, and $q \in \mathbb{N}$
- A subset $A' \subseteq A$ is **saturated by q-stars into B'** if we can assign to each $v \in A'$ a set of q private neighbors from B'



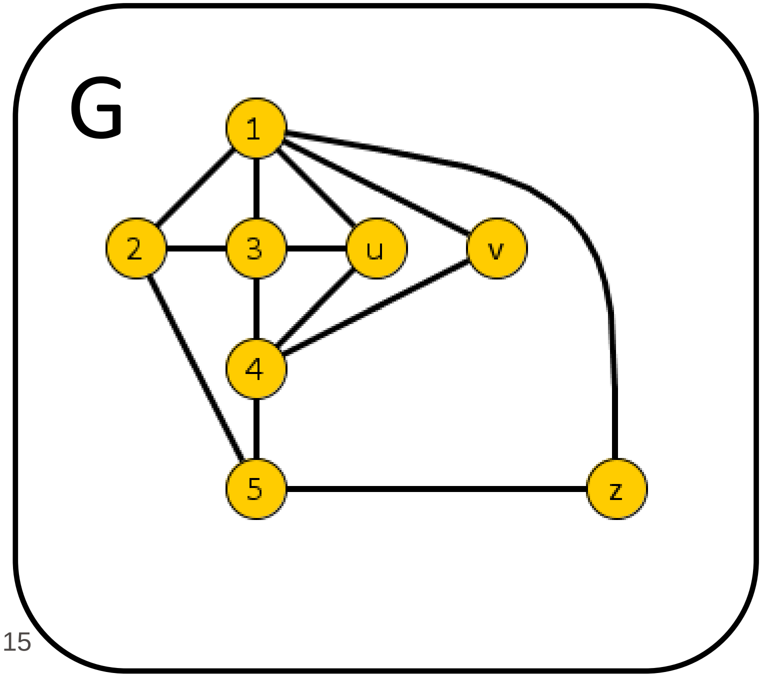
q-Expansion Lemma [Fomin et al.@STACS'11]
Let m be the size of a maximum matching H. If $|B| > m \cdot q$, then there is a nonempty set $T \subseteq B$ such that $N_H(T)$ is saturated by q-stars into T. It can be found in polynomial time.

Finding Treewidth-Invariant Sets

- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$

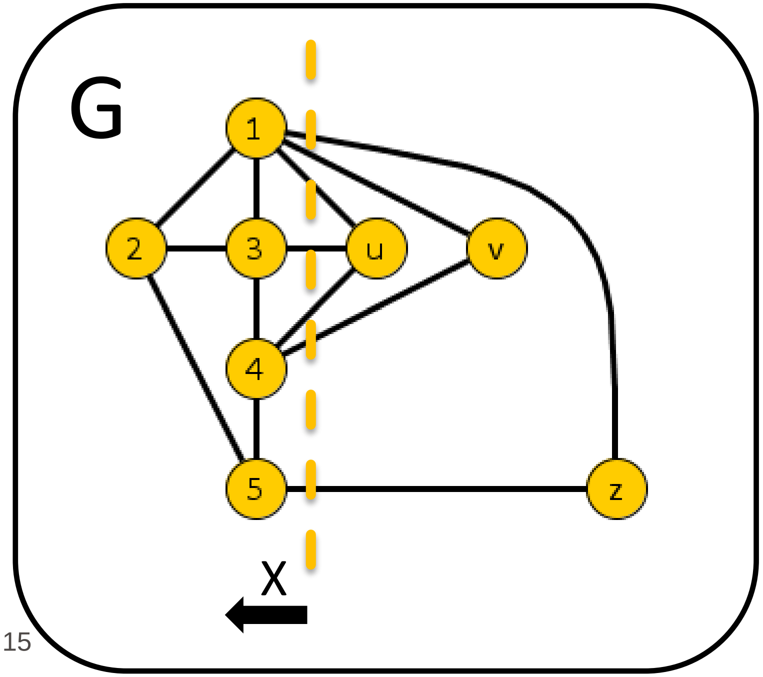
Finding Treewidth-Invariant Sets

- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$



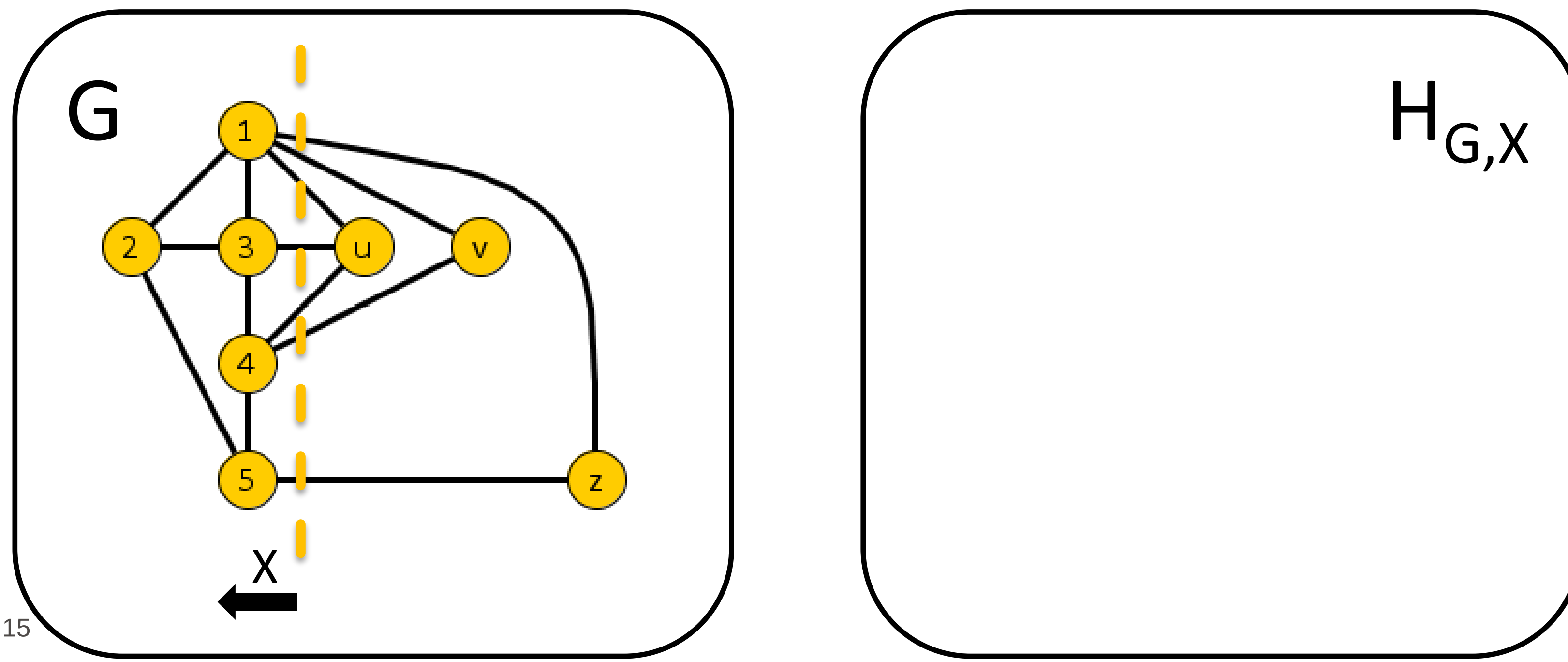
Finding Treewidth-Invariant Sets

- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$



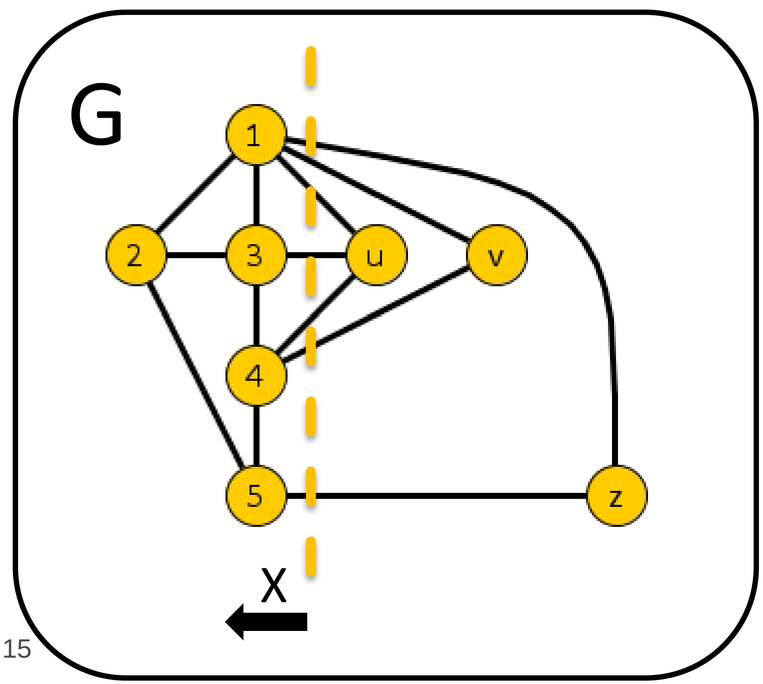
Finding Treewidth-Invariant Sets

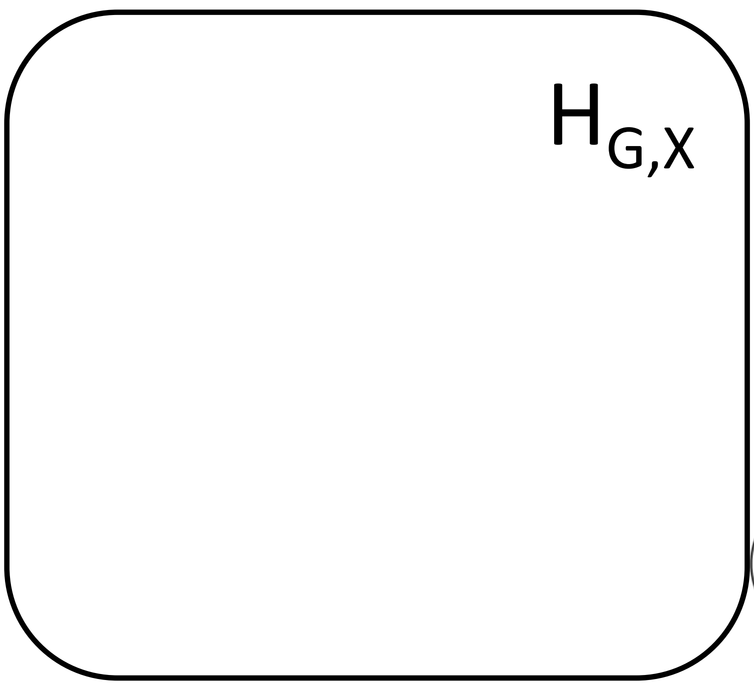
- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$



Finding Treewidth-Invariant Sets

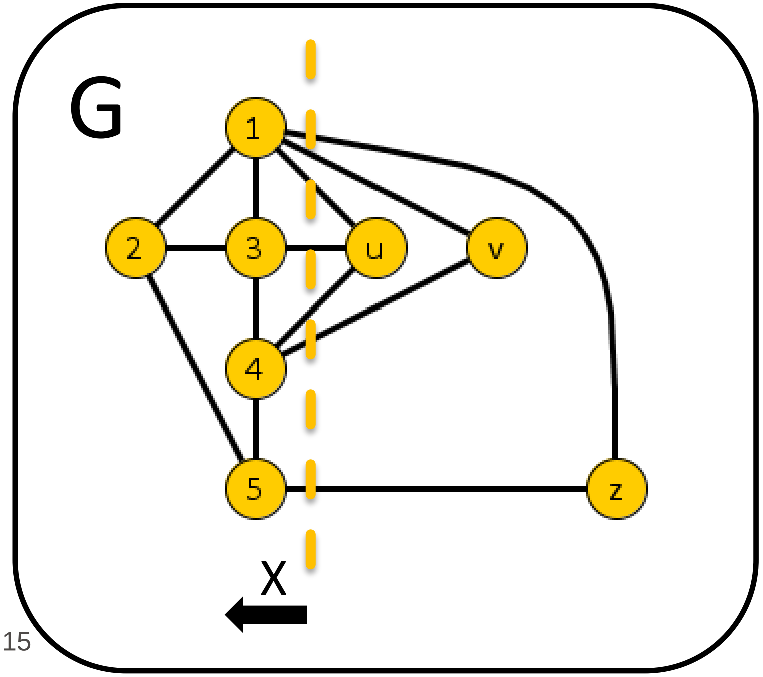
- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$
  - One side corresponds to **non-edges** in G[X]

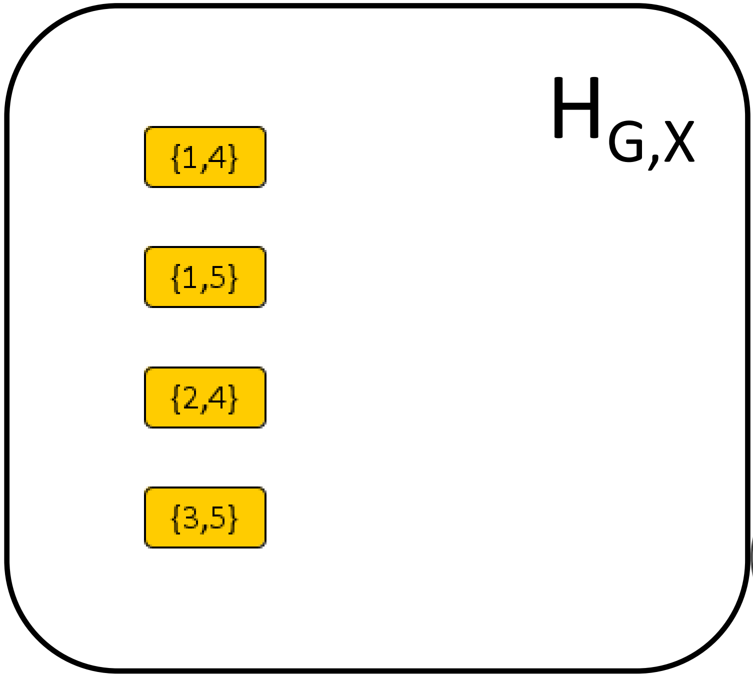




Finding Treewidth-Invariant Sets

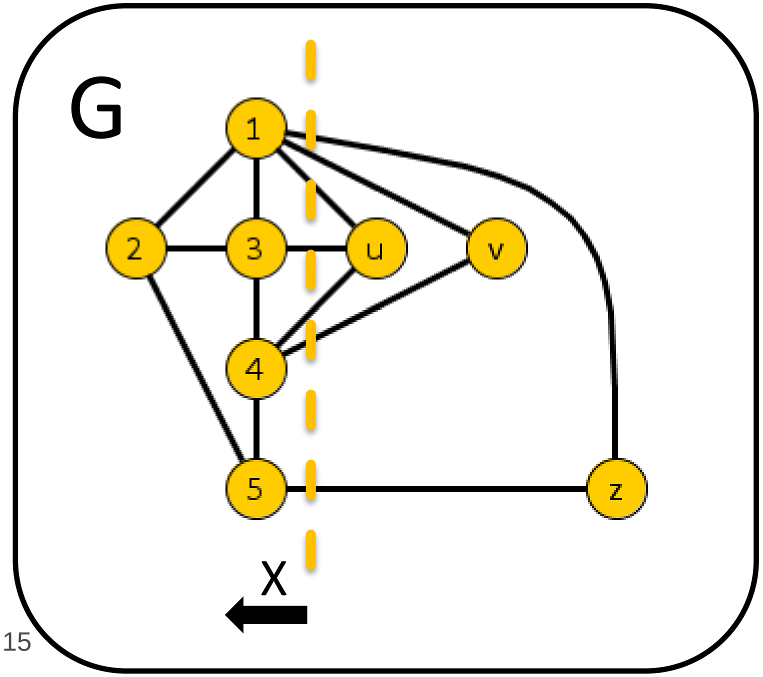
- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$
  - One side corresponds to **non-edges** in G[X]

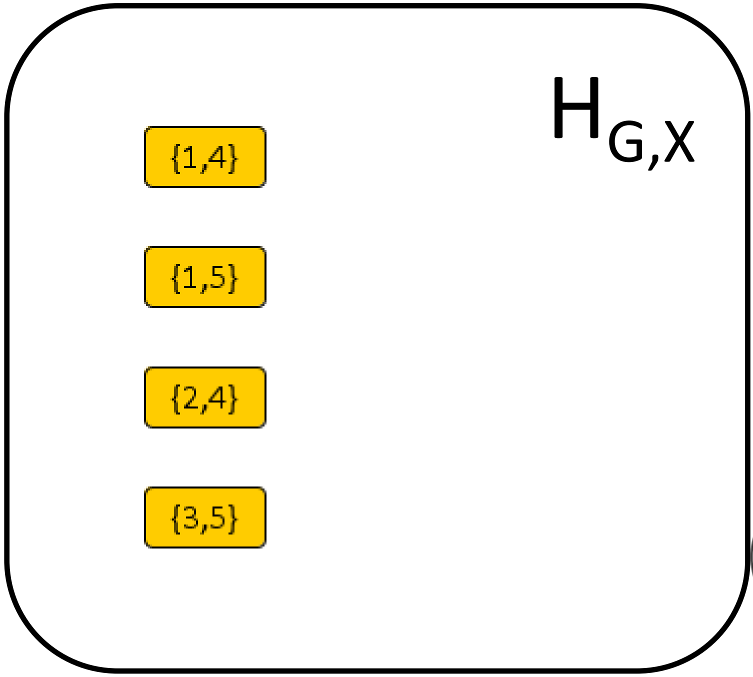




Finding Treewidth-Invariant Sets

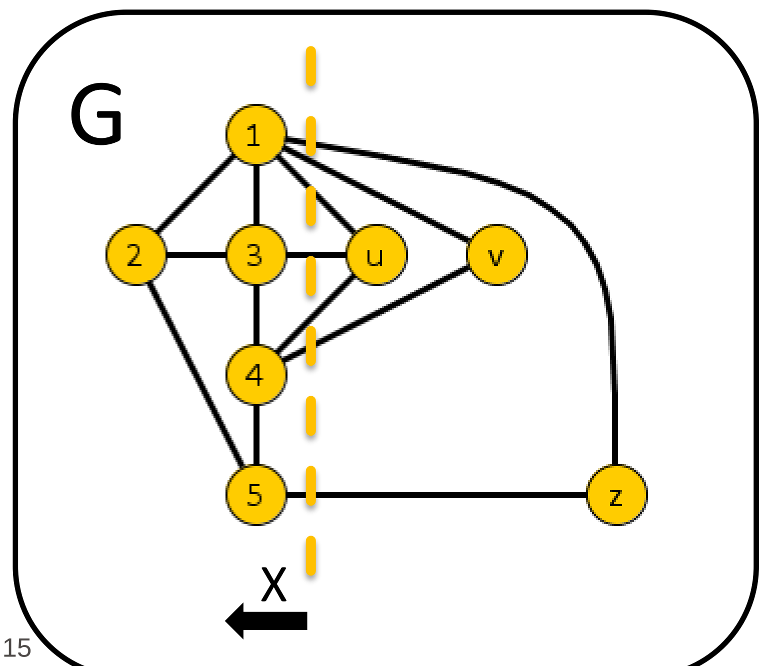
- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$
  - One side corresponds to **non-edges** in G[X]
  - One side consists of the **independent set** V(G) – X

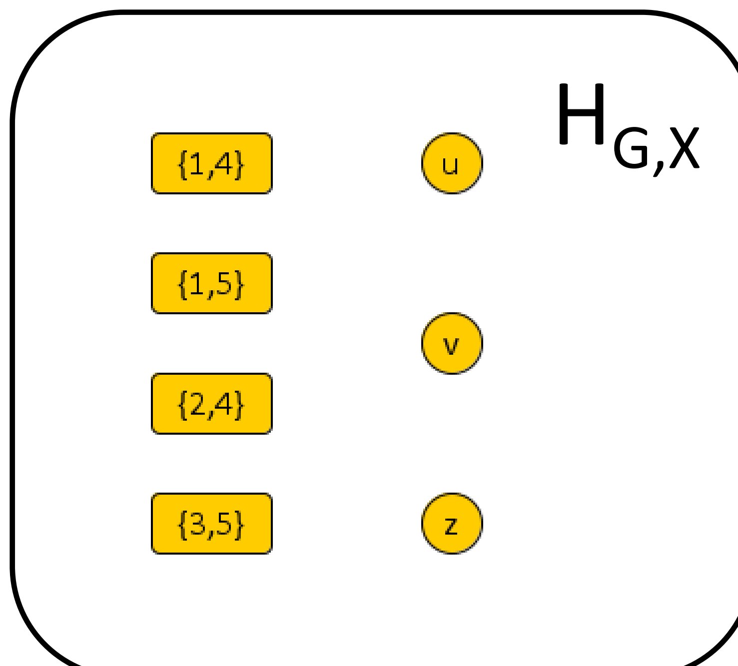




Finding Treewidth-Invariant Sets

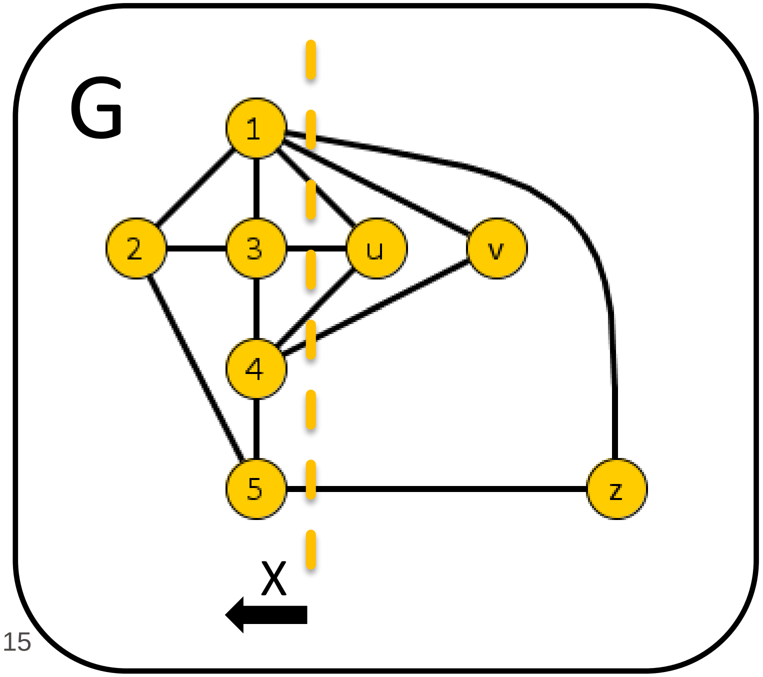
- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$
  - One side corresponds to **non-edges** in G[X]
  - One side consists of the **independent set** V(G) – X

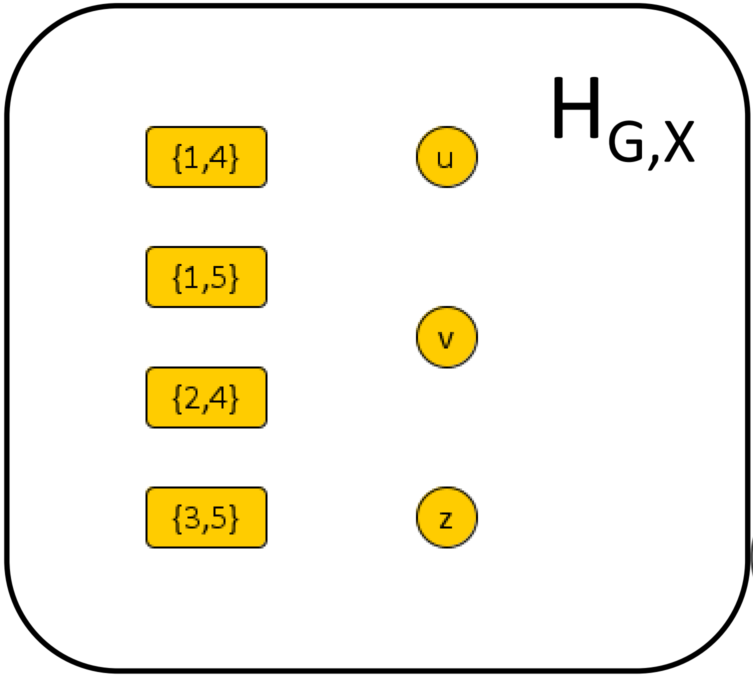




Finding Treewidth-Invariant Sets

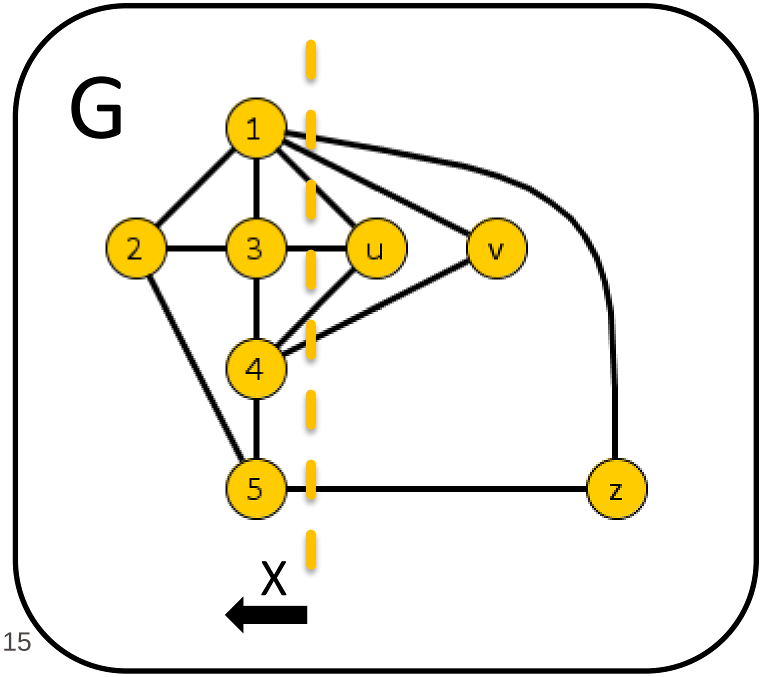
- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$
  - One side corresponds to **non-edges** in G[X]
  - One side consists of the **independent set** V(G) – X
  - $H_{G,X}$ has an edge between non-edge {p,q} and v ∈ V(G) – X, if v is adjacent to **both** p and q in G

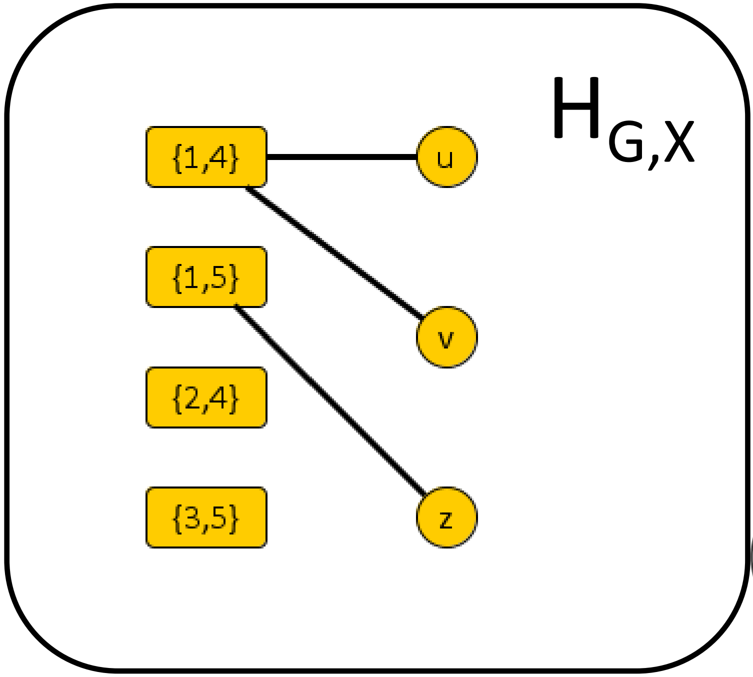




Finding Treewidth-Invariant Sets

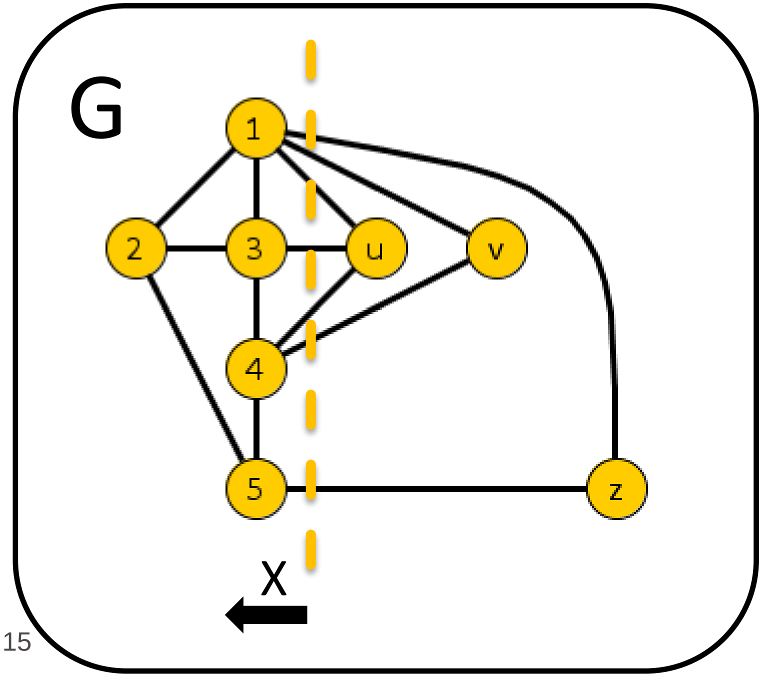
- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$
  - One side corresponds to **non-edges** in G[X]
  - One side consists of the **independent set** V(G) – X
  - $H_{G,X}$ has an edge between non-edge {p,q} and v ∈ V(G) – X, if v is adjacent to **both** p and q in G

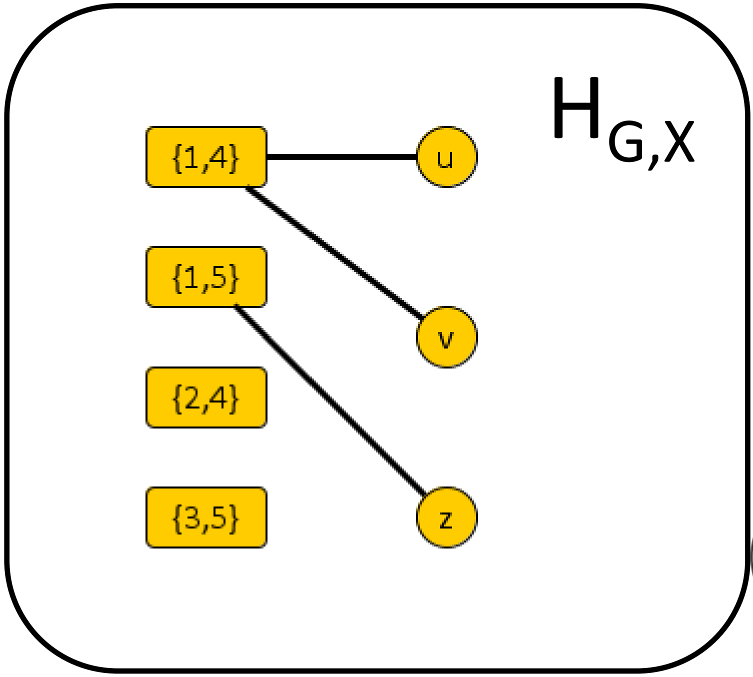




Finding Treewidth-Invariant Sets

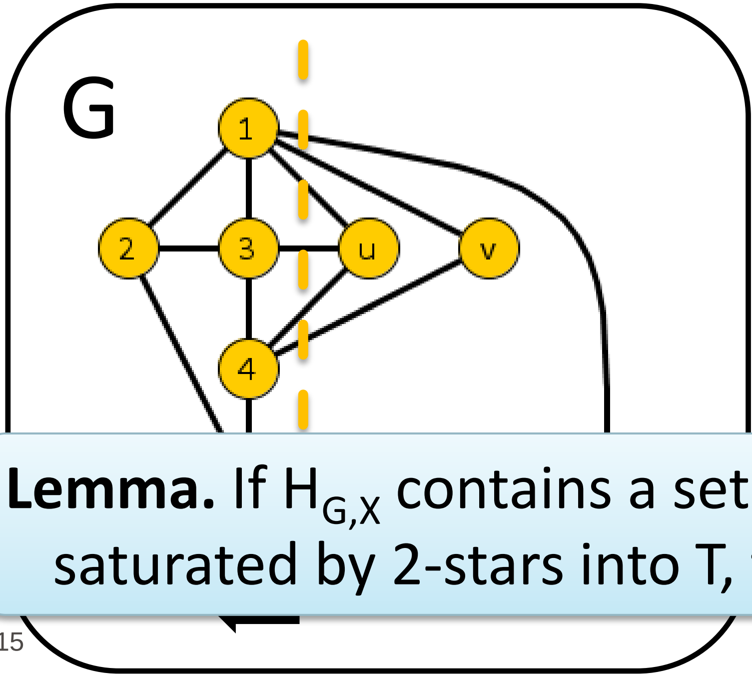
- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$
  - One side corresponds to **non-edges** in G[X]
  - One side consists of the **independent set** V(G) – X
  - $H_{G,X}$ has an edge between non-edge {p,q} and v ∈ V(G) – X, if v is adjacent to **both** p and q in G
    - Contracting v into p or q creates the edge {p,q}

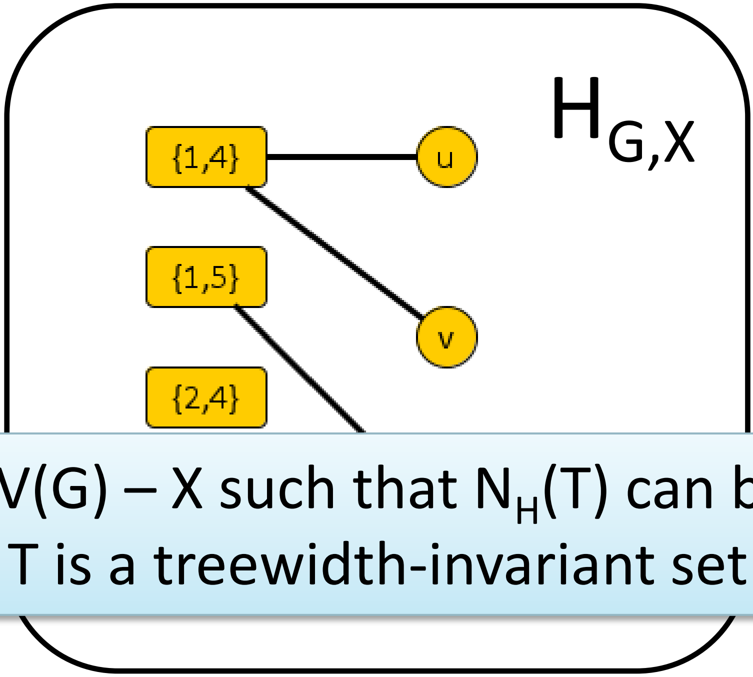




Finding Treewidth-Invariant Sets

- Given a graph G with vertex cover X, form a bipartite non-edge connection graph $H_{G,X}$
  - One side corresponds to **non-edges** in G[X]
  - One side consists of the **independent set** V(G) – X
  - $H_{G,X}$ has an edge between non-edge {p,q} and v ∈ V(G) – X, if v is adjacent to **both** p and q in G
    - Contracting v into p or q creates the edge {p,q}





Lemma. If $H_{G,X}$ contains a set T ⊆ V(G) – X such that $N_H(T)$ can be saturated by 2-stars into T, then T is a treewidth-invariant set

Kernel Size Bound

- Maximum matching in $H_{G,X}$ has size at most $\binom{|X|}{2}$
  - Non-edge side has at most this many vertices

Kernel Size Bound

- Maximum matching in $H_{G,X}$ has size at most $\binom{|X|}{2}$
  - Non-edge side has at most this many vertices

q-Expansion Lemma [Fomin et al.@STACS'11]
Let m be the size of a maximum matching H. If $|B| > m \cdot q$, then there is a nonempty set $T \subseteq B$ such that $N_H(T)$ is saturated by q-stars into T. It can be found in polynomial time.

Kernel Size Bound

- Maximum matching in $H_{G,X}$ has size at most $\binom{|X|}{2}$
  - Non-edge side has at most this many vertices

q-Expansion Lemma [Fomin et al.@STACS'11]
Let m be the size of a maximum matching H. If $|B| > m \cdot q$, then there is a nonempty set $T \subseteq B$ such that $N_H(T)$ is saturated by q-stars into T. It can be found in polynomial time.

- If $|\bar{X}| > 2\binom{|X|}{2}$, find a 2-expansion in poly-time and reduce
- Reduced instances have $\leq |X| + 2\binom{|X|}{2} = |X|^2$ vertices
  - Encoding into $\mathcal{O}(|X|^3)$ bits

Kernel Size Bound

- Maximum matching in $H_{G,X}$ has size at most $\binom{|X|}{2}$
  - Non-edge side has at most this many vertices

q-Expansion Lemma [Fomin et al.@STACS'11]
Let m be the size of a maximum matching H. If $|B| > m \cdot q$, then there is a nonempty set $T \subseteq B$ such that $N_H(T)$ is saturated by q-stars into T. It can be found in polynomial time.

- If $|\bar{X}| > 2\binom{|X|}{2}$, find a 2-expansion in poly-time and reduce

- Reduced instances have $\leq |X| + 2\binom{|X|}{2} = |X|^2$ vertices
  - Encoding into $\mathcal{O}(|X|^3)$ bits

Theorem. TREEWIDTH [VC] has a kernel with $|X|^2$ vertices that can be encoded in $\mathcal{O}(|X|^3)$ bits

Conclusion

- Main contributions

1. No nontrivial polynomial-time sparsification for TREEWIDTH and PATHWIDTH, unless NP ⊆ coNP/poly

2. TREEWIDTH [VC] has a kernel with $|X|^2$ vertices

Conclusion

- Main contributions

1. No nontrivial polynomial-time sparsification for TREEWIDTH and PATHWIDTH, unless NP $\subseteq$ coNP/poly

2. TREEWIDTH [VC] has a kernel with $|X|^2$ vertices

- Open problems

1. Are there graphs whose edge-count is superquadratic in their vertex cover number, which do not have treewidth-invariant sets?

2. Which problems admit nontrivial polynomial-time sparsification?

3. Does TREEWIDTH [VC] have a kernel of bitsize $\mathcal{O}(|X|^2)$?

4. Does PATHWIDTH [VC] have a kernel with $\mathcal{O}(|X|^2)$ vertices?

Thank you!

UNIVERSITY OF BERGEN

Algorithms Research Group

Invariance Property

- Let $\Delta(T) := \max_{v \in T} \deg(v)$

Invariance Property

- Let $\Delta(T) := \max_{v \in T} \deg(v)$

Lemma. If T is a treewidth-invariant set in G, then $\textsc{tw}(G) = \max\{\textsc{tw}(\hat{G}_T), \Delta(T)\}$

Invariance Property

- Let $\Delta(T) := \max_{v \in T} \deg(v)$

Lemma. If T is a treewidth-invariant set in G, then $\textsc{tw}(G) = \max\{\textsc{tw}(\hat{G}_T), \Delta(T)\}$

- Proof.

Invariance Property

- Let $\Delta(T) := \max_{v \in T} \deg(v)$

Lemma. If T is a treewidth-invariant set in G, then $\textsc{tw}(G) = \max\{\textsc{tw}(\hat{G}_T), \Delta(T)\}$

- Proof.
  - **(≥)** G contains $\hat{G}_T$ and a $(\Delta(T)+1)$-clique as a minor

Invariance Property

- Let $\Delta(T) := \max_{v \in T} \deg(v)$

Lemma. If T is a treewidth-invariant set in G,
then $\textsc{tw}(G) = \max\{\textsc{tw}(\hat{G}_T), \Delta(T)\}$

- Proof.
  - **(≥)** G contains $\hat{G}_T$ and a $(\Delta(T)+1)$-clique as a minor
  - **(≤)** Consider a tree decomposition $\mathcal{T}$ of $\hat{G}_T$
    - For every $v \in T$, $N_G(v)$ exists in $\hat{G}_T$ and forms a clique there
    - So $\mathcal{T}$ has a bag containing $N_G(v)$

Invariance Property

- Let $\Delta(T) := \max_{v \in T} \deg(v)$

Lemma. If T is a treewidth-invariant set in G,
then $\mathrm{TW}(G) = \max\{\mathrm{TW}(\hat{G}_T), \Delta(T)\}$

- Proof.
  - **(≥)** G contains $\hat{G}_T$ and a $(\Delta(T)+1)$-clique as a minor
  - **(≤)** Consider a tree decomposition $\mathcal{T}$ of $\hat{G}_T$
    - For every $v \in T$, $N_G(v)$ exists in $\hat{G}_T$ and forms a clique there
    - So $\mathcal{T}$ has a bag containing $N_G(v)$
    - Append a new bag with $N_G(v) \cup \{v\}$, of size $\leq \Delta(T) + 1$
    - Update independently for each $v \in T$