Lower bounds for polynomial kernelization

Part 2

Michał Pilipczuk

Institutt for Informatikk, Universitetet i Bergen

August $20^{\rm th}$, 2014

Outline

- **Goal**: how to prove that for some problems polynomial kernels do **not** exist?
- **Part 1**:
  - Introduction of the (cross)-composition framework.
  - Basic example.
- **Part 2:**
  - PPT reductions.
  - Case study of several cross-compositions.
  - Weak compositions.

In the previous episode...

- **Composition**: an algorithm composing many instances into one instance simulating their OR.

In the previous episode...

- **Composition**: an algorithm composing many instances into one instance simulating their OR.
  - The new instance has parameter bounded polynomially in the maximum size of an input instance.

In the previous episode...

- **Composition**: an algorithm composing many instances into one instance simulating their OR.
    - The new instance has parameter bounded polynomially in the maximum size of an input instance.
- Composition+Compression gives OR-Distillation

In the previous episode...

- **Composition**: an algorithm composing many instances into one instance simulating their OR.
  - The new instance has parameter bounded polynomially in the maximum size of an input instance.
- Composition+Compression gives OR-Distillation
- OR-Distillation of an **NP**-hard language contradicts **coNP** $\subseteq$ **NP**/poly.

In the previous episode...

- **Composition**: an algorithm composing many instances into one instance simulating their OR.
  - The new instance has parameter bounded polynomially in the maximum size of an input instance.
- Composition+Compression gives OR-Distillation
- OR-Distillation of an **NP**-hard language contradicts **coNP** $\subseteq$ **NP**$/\mathrm{poly}$.
- **Corollary**: To show no-poly-kernel it suffices to construct a composition algorithm.

In the previous episode...

Cross-composition

An unparameterized problem Q *cross-composes* into a parameterized problem L, if there exists a polynomial equivalence relation $\mathcal{R}$ and an algorithm that, given $\mathcal{R}$-equivalent strings $x_1, x_2, \ldots, x_t$, in time $\mathrm{poly}\left(t + \sum_{i=1}^{t} |x_i|\right)$ produces one instance $(y, k^\star)$ such that

- $(y, k^\star) \in L$ iff $x_i \in Q$ for at least one $i = 1, 2, \ldots, t$,
- $k^\star = \mathrm{poly}\left(\log t + \max_{i=1}^{t} |x_i|\right)$.

Cross-composition theorem

Bodlaender et al.; STACS 2011, SIDMA 2014

If some **NP**-hard problem Q cross-composes into L, then L does not admit a polynomial compression into any language R, unless **NP** $\subseteq$ **coNP**$/\mathrm{poly}$.

PPTs

- **Idea**: Hardness of kernelization can be transferred via reductions, similarly to **NP**-hardness.

PPTs

- **Idea**: Hardness of kernelization can be transferred via reductions, similarly to **NP**-hardness.

Polynomial parameter transformation (PPT)

A *polynomial parameter transformation* from a parameterized problem P to a parameterized problem Q is a polynomial-time algorithm that transforms a given instance (x, k) of P into an equivalent instance (y, k') of Q such that $k' = \mathrm{poly}(k)$.

PPTs: properties

Observation

If problem P PPT-reduces to Q, and P does not admit a polynomial compression algorithm (into any language R), then neither does Q.

PPTs: properties

Observation

If problem P PPT-reduces to Q, and P does not admit a polynomial compression algorithm (into any language R), then neither does Q.

- **Proof**: Compose the PPT-reduction with the assumed compression for Q.

Application 2: Steiner Tree

Steiner Tree

Input: Graph G with designated terminals $T \subseteq V(G)$, and an integer k

Parameter: $k + |T|$

Question: Is there a set $X \subseteq V(G) \setminus T$, such that $|X| \leq k$ and $G[T \cup X]$ is connected?

Application 2: Steiner Tree

Steiner Tree

Input: Graph G with designated terminals $T \subseteq V(G)$, and an integer k
Parameter: $k + |T|$
Question: Is there a set $X \subseteq V(G) \setminus T$, such that $|X| \leq k$ and $G[T \cup X]$ is connected?

- Follows from a PPT from Set Cover par. by $|U|$.

Application 2: Steiner Tree

Steiner Tree

Input: Graph G with designated terminals $T \subseteq V(G)$, and an integer k

Parameter: $k + |T|$

Question: Is there a set $X \subseteq V(G) \setminus T$, such that $|X| \leq k$ and $G[T \cup X]$ is connected?

- Follows from a PPT from Set Cover par. by $|U|$.
- But we will present an alternative approach.

The pivot problem technique

- Introduce an simpler problem P, which is almost trivially compositional.

The pivot problem technique

- Introduce an simpler problem P, which is almost trivially compositional.
- Then design a PPT from P to the target problem.

The pivot problem technique

- Introduce an simpler problem P, which is almost trivially compositional.
- Then design a PPT from P to the target problem.
- Move the weight of the proof to the transformation and the actual definition of P.

The pivot problem technique

- Introduce an simpler problem P, which is almost trivially compositional.
- Then design a PPT from P to the target problem.
- Move the weight of the proof to the transformation and the actual definition of P.
- **Idea**: Extract the essence of the problem.

Colourful Graph Motif

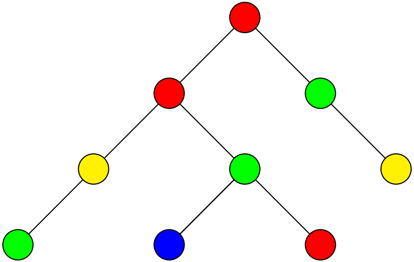
Colourful Graph Motif

Input: Graph G and a colouring function $\mathcal{C} : V(G) \to \{1, 2, \ldots, k\}$

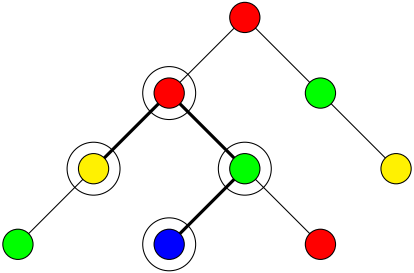
Parameter: k

Question: Does there exists a connected subgraph H of G containing exactly one vertex of each colour?

Colourful Graph Motif — example



Colourful Graph Motif — example



About CGM

- Introduced by Fellows et al. (ICALP 2007, JCSS 2011).

About CGM

- Introduced by Fellows et al. (ICALP 2007, JCSS 2011).
- **NP**-hard even on trees.

About CGM

- Introduced by Fellows et al. (ICALP 2007, JCSS 2011).
- **NP**-hard even on trees.
- Interesting FPT algorithms for various variants using the algebraic approach.

About CGM

- Introduced by Fellows et al. (ICALP 2007, JCSS 2011).
- **NP**-hard even on trees.
- Interesting FPT algorithms for various variants using the algebraic approach.
- Trivial composition algorithm: take the disjoint union of instances, reuse colors.

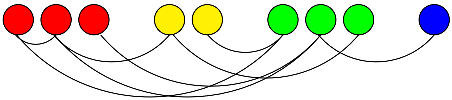
About CGM

- Introduced by Fellows et al. (ICALP 2007, JCSS 2011).
- **NP**-hard even on trees.
- Interesting FPT algorithms for various variants using the algebraic approach.
- Trivial composition algorithm: take the disjoint union of instances, reuse colors.
- Hence, CGM does not have a polykernel unless **coNP** $\subseteq$ **NP**$/\mathrm{poly}$.

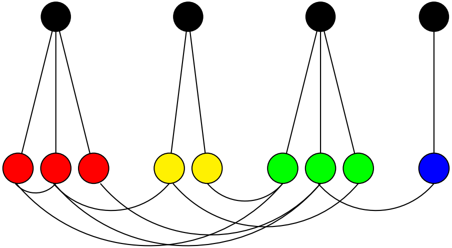
About CGM

- Introduced by Fellows et al. (ICALP 2007, JCSS 2011).
- **NP**-hard even on trees.
- Interesting FPT algorithms for various variants using the algebraic approach.
- Trivial composition algorithm: take the disjoint union of instances, reuse colors.
- Hence, CGM does not have a polykernel unless **coNP** $\subseteq$ **NP**$/\mathrm{poly}$.
- **Now**: PPT-reduction from CGM to ST.

From CGM to ST



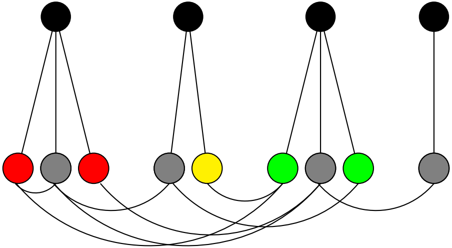
From CGM to ST



Attach a terminal to every colour class

Give budget k for Steiner nodes

From CGM to ST



Attach a terminal to every colour class

Give budget k for Steiner nodes

Wrapping up

- CGM does not admit a polynomial kernel, unless **coNP** $\subseteq$ **NP**$/\mathrm{poly}$.

Wrapping up

- CGM does not admit a polynomial kernel, unless **coNP** $\subseteq$ **NP**$/\mathrm{poly}$.
- CGM PPT-reduces to Steiner Tree par. by $k + |T|$.

Wrapping up

- CGM does not admit a polynomial kernel, unless **coNP** $\subseteq$ **NP**$/\mathrm{poly}$.
- CGM PPT-reduces to Steiner Tree par. by $k + |T|$.
- Hence Steiner Tree par. by $k + |T|$ does not admit a polynomial kernel, unless **coNP** $\subseteq$ **NP**$/\mathrm{poly}$.

Application 3: SET COVER par. by $|U|$

SET COVER

Input: Universe U, a family of subsets $\mathcal{F} \subseteq 2^U$, integer k
Parameter: $|U|$
Question: Is there a subfamily $\mathcal{G} \subseteq \mathcal{F}$, $|\mathcal{G}| \leq k$, such that $\bigcup \mathcal{G} = U$?

Application 3: Set Cover par. by $|U|$

Set Cover

Input: Universe U, a family of subsets $\mathcal{F} \subseteq 2^U$, integer k
Parameter: $|U|$
Question: Is there a subfamily $\mathcal{G} \subseteq \mathcal{F}$, $|\mathcal{G}| \leq k$, such that $\bigcup \mathcal{G} = U$?

- **Convention**: We view it as a bipartite graph with one side (blue) trying to dominate the other one (red).

Application 3: Set Cover par. by $|U|$

Set Cover

Input:	Universe U, a family of subsets $\mathcal{F} \subseteq 2^U$, integer k
Parameter:	$|U|$
Question:	Is there a subfamily $\mathcal{G} \subseteq \mathcal{F}$, $|\mathcal{G}| \leq k$, such that $\bigcup \mathcal{G} = U$?

- **Convention**: We view it as a bipartite graph with one side (blue) trying to dominate the other one (red).
- W.l.o.g. $k \leq |U|$.

Colourful Set Cover

Colourful Set Cover

Input: Universe U and families $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k \subseteq 2^U$
Parameter: $|U| + k$
Question: Is there a family $\mathcal{G}$ containing exactly one set from each family $\mathcal{F}_i$, such that $\bigcup \mathcal{G} = U$?

Equivalence of the problems

- $SC \leq_{PPT} CSC$:

Equivalence of the problems

- $SC \leq_{PPT} CSC$:
  - Put $\mathcal{F}_i = \mathcal{F}$ for every i.

Equivalence of the problems

- $SC \leq_{PPT} CSC$:
  - Put $\mathcal{F}_i = \mathcal{F}$ for every i.
- $CSC \leq_{PPT} SC$:

Equivalence of the problems

- $SC \leq_{PPT} CSC$:
  - Put $\mathcal{F}_i = \mathcal{F}$ for every i.
- $CSC \leq_{PPT} SC$:
  - Add k elements $e_1, e_2, \ldots, e_k$; include e_i in every set from $\mathcal{F}_i$.

Equivalence of the problems

- $SC \leq_{PPT} CSC$:
  - Put $\mathcal{F}_i = \mathcal{F}$ for every i.
- $CSC \leq_{PPT} SC$:
  - Add k elements $e_1, e_2, \ldots, e_k$; include e_i in every set from $\mathcal{F}_i$.
  - Then take $\mathcal{F} = \bigcup \mathcal{F}_i$.

Equivalence of the problems

- $SC \leq_{PPT} CSC$:
  - Put $\mathcal{F}_i = \mathcal{F}$ for every i.
- $CSC \leq_{PPT} SC$:
  - Add k elements $e_1, e_2, \ldots, e_k$; include e_i in every set from $\mathcal{F}_i$.
  - Then take $\mathcal{F} = \bigcup \mathcal{F}_i$.
- We will cross-compose COLOURFUL SET COVER into itself.

Equivalence of the problems

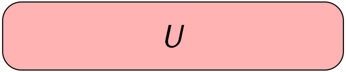
- $SC \leq_{PPT} CSC$:
  - Put $\mathcal{F}_i = \mathcal{F}$ for every i.
- $CSC \leq_{PPT} SC$:
  - Add k elements $e_1, e_2, \ldots, e_k$; include e_i in every set from $\mathcal{F}_i$.
  - Then take $\mathcal{F} = \bigcup \mathcal{F}_i$.
- We will cross-compose Colourful Set Cover into itself.
- **Assumption**: the same universe U, the same k, and t being a power of 2.

Cross-composing into Colourful Set Cover

Input: Instances $(U, (\mathcal{F}^i_j)_{1 \le j \le k})$

Output: Instance $(U^*, (\mathcal{F}^*_j)_{1 \le j \le k})$

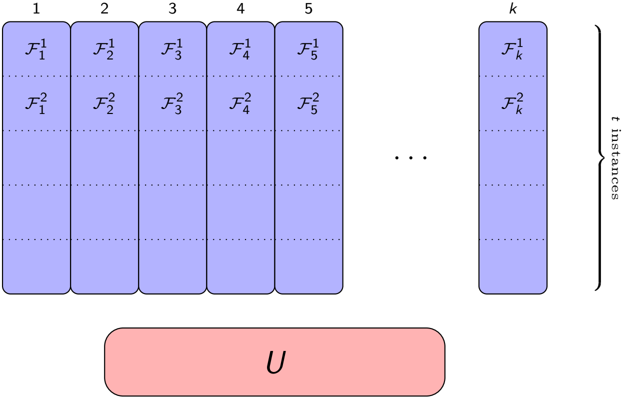
Cross-composing into COLOURFUL SET COVER



Input: Instances $(U, (\mathcal{F}_j^i)_{1 \leq j \leq k})$

Output: Instance $(U^*, (\mathcal{F}_j^*)_{1 \leq j \leq k})$

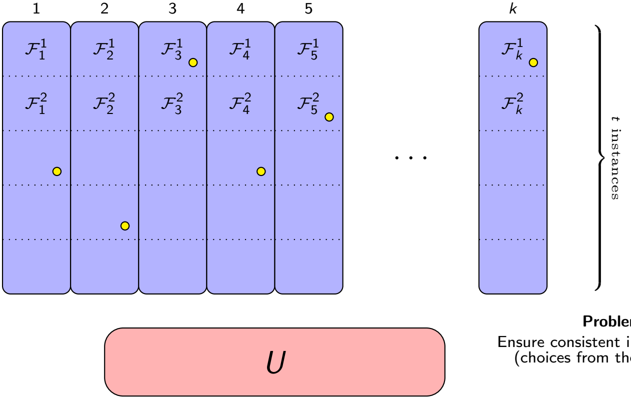
Cross-composing into Colourful Set Cover



Input: Instances $(U, (\mathcal{F}_j^i)_{1 \le j \le k})$

Output: Instance $(U^*, (\mathcal{F}_j^*)_{1 \le j \le k})$

Cross-composing into Colourful Set Cover

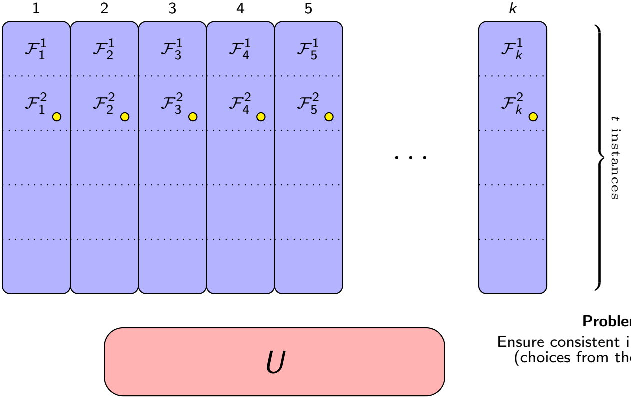


Problem:
Ensure consistent instance choice
(choices from the same row)

Input: Instances $(U, (\mathcal{F}_j^i)_{1 \leq j \leq k})$

Output: Instance $(U^*, (\mathcal{F}_j^*)_{1 \leq j \leq k})$

Cross-composing into Colourful Set Cover

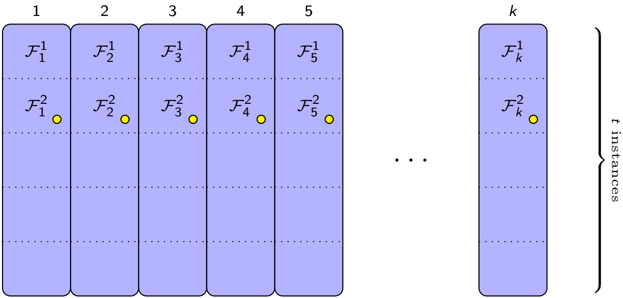


Problem:
Ensure consistent instance choice
(choices from the same row)

Input: Instances $(U, (\mathcal{F}_j^i)_{1 \leq j \leq k})$

Output: Instance $(U^*, (\mathcal{F}_j^*)_{1 \leq j \leq k})$

Cross-composing into COLOURFUL SET COVER



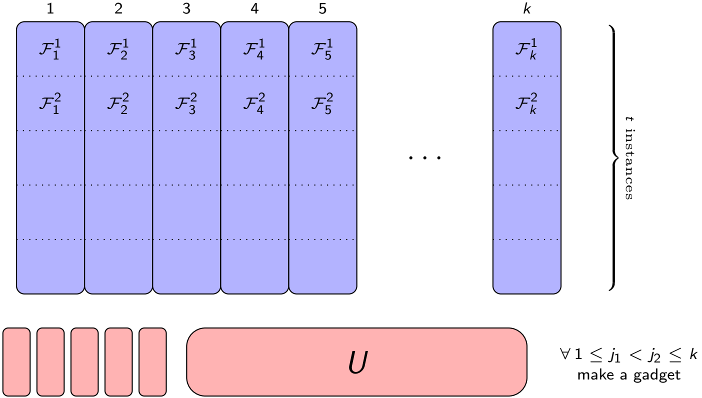
Problem:
Ensure consistent instance choice
(choices from the same row)

Solution:
Equality gadgets

Input: Instances $(U, (\mathcal{F}_j^i)_{1 \le j \le k})$

Output: Instance $(U^*, (\mathcal{F}_j^*)_{1 \le j \le k})$

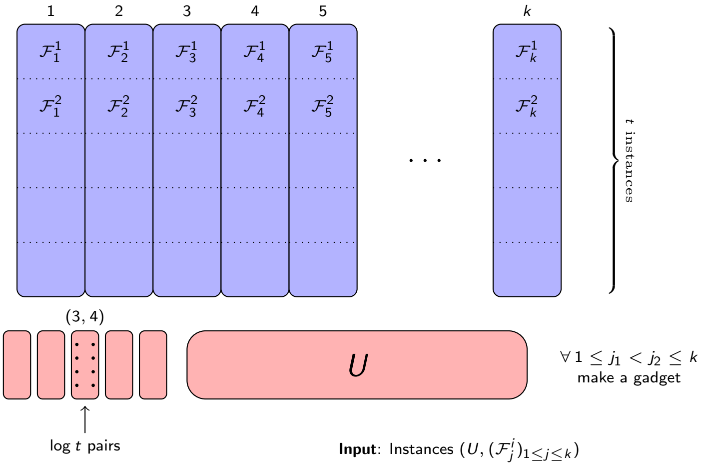
Cross-composing into Colourful Set Cover



Input: Instances $(U, (\mathcal{F}_j^i)_{1 \leq j \leq k})$

Output: Instance $(U^*, (\mathcal{F}_j^*)_{1 \leq j \leq k})$

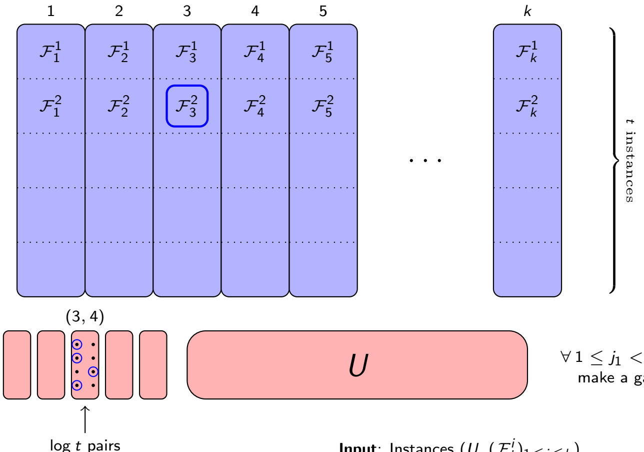
Cross-composing into Colourful Set Cover



Input: Instances $(U, (\mathcal{F}_j^i)_{1 \leq j \leq k})$

Output: Instance $(U^*, (\mathcal{F}_j^*)_{1 \leq j \leq k})$

Cross-composing into Colourful Set Cover



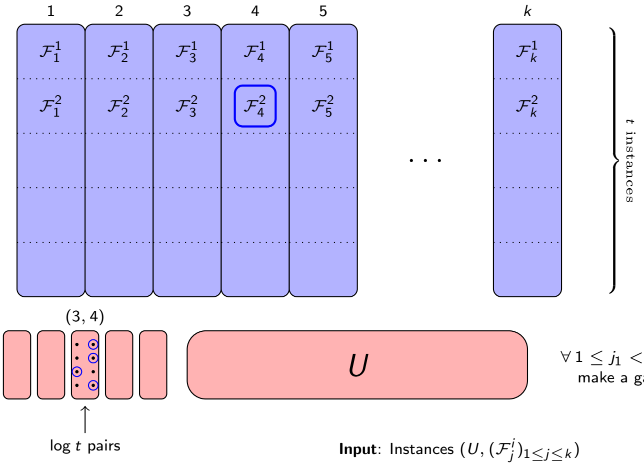
$\forall_i$ add $\mathrm{bin}(i)$ to sets from $\mathcal{F}_3^i$

$\forall\, 1 \le j_1 < j_2 \le k$ make a gadget

Input: Instances $(U, (\mathcal{F}_j^i)_{1 \le j \le k})$

Output: Instance $(U^*, (\mathcal{F}_j^*)_{1 \le j \le k})$

Cross-composing into COLOURFUL SET COVER



$\forall\, 1 \le j_1 < j_2 \le k$
make a gadget

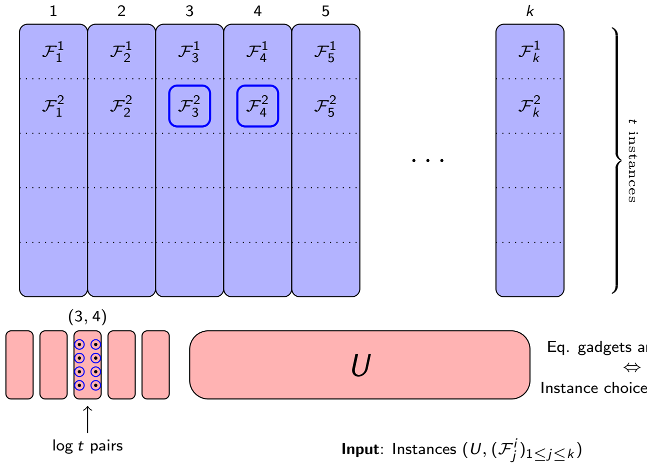
$\forall_i$ add $\mathrm{bin}(i)$ to sets from $\mathcal{F}_3^i$

$\forall_i$ add $\overline{\mathrm{bin}(i)}$ to sets from $\mathcal{F}_4^i$

Input: Instances $(U, (\mathcal{F}_j^i)_{1 \le j \le k})$

Output: Instance $(U^*, (\mathcal{F}_j^*)_{1 \le j \le k})$

Cross-composing into COLOURFUL SET COVER



Eq. gadgets are covered
⇔
Instance choices are equal

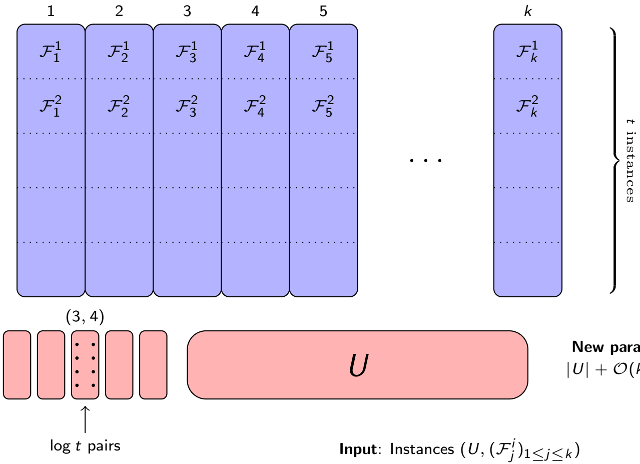
$\forall_i$ add $\mathrm{bin}(i)$ to sets from $\mathcal{F}_3^i$

$\forall_i$ add $\overline{\mathrm{bin}(i)}$ to sets from $\mathcal{F}_4^i$

Input: Instances $(U, (\mathcal{F}_j^i)_{1 \le j \le k})$

Output: Instance $(U^*, (\mathcal{F}_j^*)_{1 \le j \le k})$

Cross-composing into COLOURFUL SET COVER



New parameter: $|U| + \mathcal{O}(k^2 \log t)$

$\forall_i$ add $\mathrm{bin}(i)$ to sets from $\mathcal{F}_3^i$

$\forall_i$ add $\overline{\mathrm{bin}(i)}$ to sets from $\mathcal{F}_4^i$

Input: Instances $(U, (\mathcal{F}_j^i)_{1 \le j \le k})$

Output: Instance $(U^*, (\mathcal{F}_j^*)_{1 \le j \le k})$

Wrapping up

- SC and CSC are equivalent wrt. PPTs.

Wrapping up

- SC and CSC are equivalent wrt. PPTs.
- CSC does not admit a polynomial compression, unless **NP** $\subseteq$ **coNP**$/\mathrm{poly}$.

Wrapping up

- SC and CSC are equivalent wrt. PPTs.
- CSC does not admit a polynomial compression, unless $\mathbf{NP} \subseteq \mathbf{coNP}/\mathrm{poly}$.
- So neither does SC.

Wrapping up

- SC and CSC are equivalent wrt. PPTs.
- CSC does not admit a polynomial compression, unless $\mathbf{NP} \subseteq \mathbf{coNP}/\mathrm{poly}$.
- So neither does SC.
- **Note**: parameterization of Set Cover by $|\mathcal{F}|$ also does not admit a polynomial compression.

Wrapping up

- SC and CSC are equivalent wrt. PPTs.
- CSC does not admit a polynomial compression, unless $\mathbf{NP} \subseteq \mathbf{coNP}/\mathrm{poly}$.
- So neither does SC.
- **Note**: parameterization of Set Cover by $|\mathcal{F}|$ also does not admit a polynomial compression.
  - The composition is quite different.

Structural parameters

- **Idea**: Parameterize the problem by a quantitative measure of structure of the graph, rather than intended solution size.

Structural parameters

- **Idea**: Parameterize the problem by a quantitative measure of structure of the graph, rather than intended solution size.
  - **Example**: treewidth parameterizations

Structural parameters

- **Idea**: Parameterize the problem by a quantitative measure of structure of the graph, rather than intended solution size.
  - **Example**: treewidth parameterizations
- From kernelization point of view: work of Bodlaender, Jansen, and Kratsch.

Structural parameters

- **Idea**: Parameterize the problem by a quantitative measure of structure of the graph, rather than intended solution size.
  - **Example**: treewidth parameterizations
- From kernelization point of view: work of Bodlaender, Jansen, and Kratsch.
- Original motivation of cross-composition.

Application 4: CLIQUE par. by vertex cover

CLIQUE/VC

Input: Graph G, a vertex cover X of G, integer k
Parameter: $|X|$
Question: Is there a clique of size k in G?

Application 4: Clique par. by vertex cover

Clique/VC

Input: Graph G, a vertex cover X of G, integer k
Parameter: $|X|$
Question: Is there a clique of size k in G?

- W.l.o.g. $k \leq |X| + 1$.

Application 4: Clique par. by vertex cover

Clique/VC

Input:	Graph G, a vertex cover X of G, integer k
Parameter:	$\|X\|$
Question:	Is there a clique of size k in G?

- W.l.o.g. $k \leq |X| + 1$.
- Trivially FPT.

Application 4: Clique par. by vertex cover

Clique/VC

Input: Graph G, a vertex cover X of G, integer k
Parameter: $|X|$
Question: Is there a clique of size k in G?

- W.l.o.g. $k \leq |X| + 1$.
- Trivially FPT.
- We make a cross-composition from the standard Clique problem.

Application 4: CLIQUE par. by vertex cover

CLIQUE/VC

Input:	Graph G, a vertex cover X of G, integer k
Parameter:	$\|X\|$
Question:	Is there a clique of size k in G?

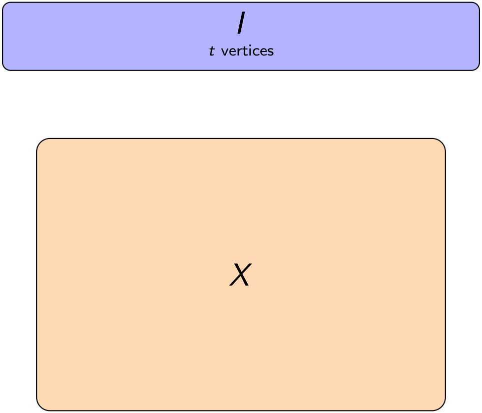
- W.l.o.g. $k \leq |X| + 1$.
- Trivially FPT.
- We make a cross-composition from the standard CLIQUE problem.
- Assume the same number of vertices n and the same target size of the clique k.

Cross-composing into CLIQUE/VC

Input: Instances (G_i, k)

Output: Instance (G, X, k^*)

Cross-composing into CLIQUE/VC

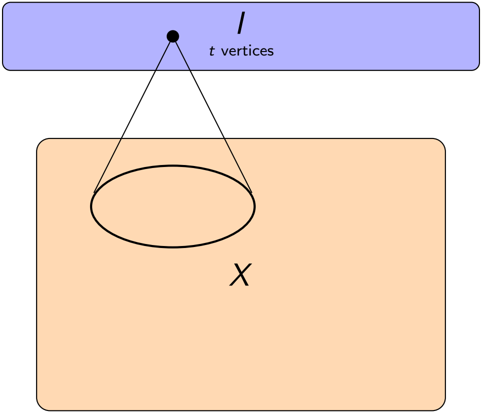


Input: Instances (G_i, k)

Output: Instance (G, X, k^*)

Size constraints force us
to take one vertex from I.

Cross-composing into CLIQUE/VC



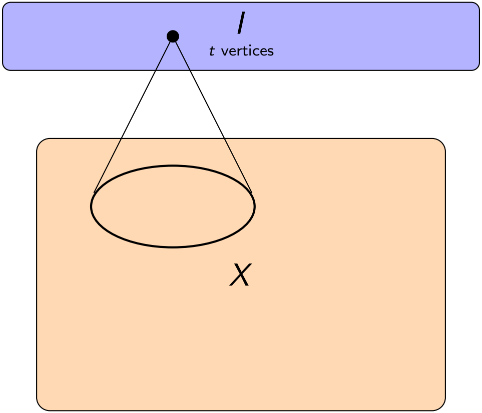
Input: Instances (G_i, k)

Output: Instance (G, X, k^*)

Size constraints force us to take one vertex from I.

Neighbourhood of i-th vertex from I acts as instance (G_i, k).

Cross-composing into CLIQUE/VC



Input: Instances (G_i, k)

Output: Instance (G, X, k^*)

Size constraints force us to take one vertex from I.

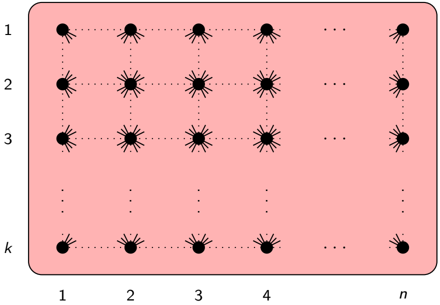
Neighbourhood of i-th vertex from I acts as instance (G_i, k).

Problem:
Design a 'universal' modulator X.

Cross-composing into CLIQUE/VC

Input: Instances (G_i, k)

Output: Instance (G, X, k^*)

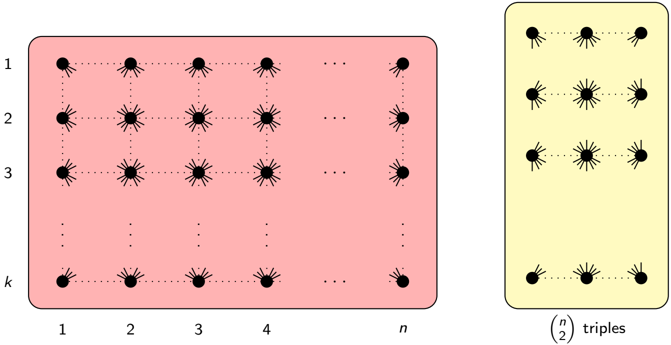


All connections are present except ones in the same row/column.

Cross-composing into CLIQUE/VC

Input: Instances (G_i, k)

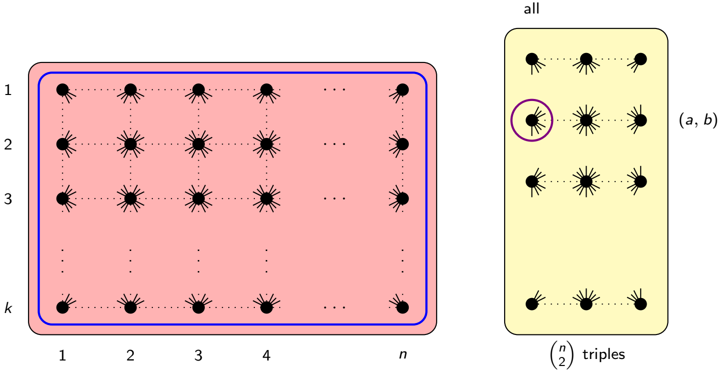
Output: Instance (G, X, k^*)



Cross-composing into CLIQUE/VC

Input: Instances (G_i, k)

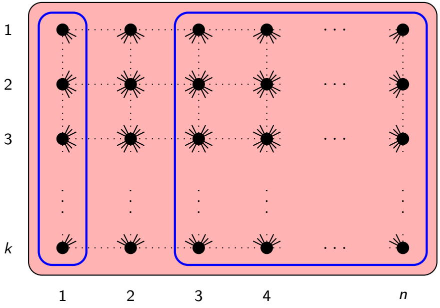
Output: Instance (G, X, k^*)

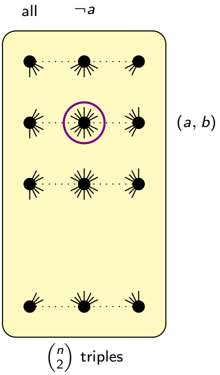


Cross-composing into CLIQUE/VC

Input: Instances (G_i, k)

Output: Instance (G, X, k^*)

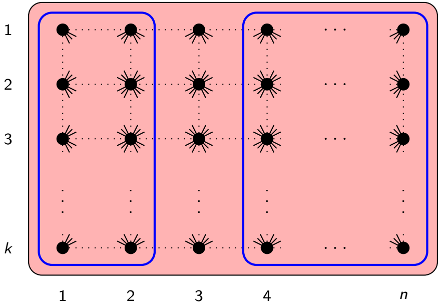


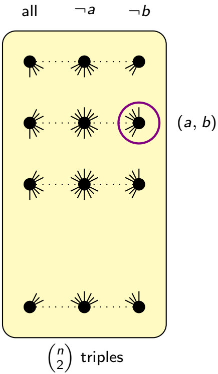


Cross-composing into Clique/VC

Input: Instances (G_i, k)

Output: Instance (G, X, k^*)

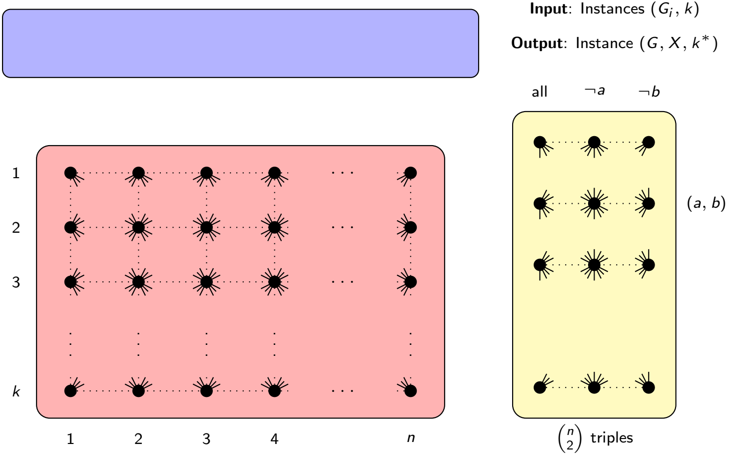




Cross-composing into Clique/VC

Input: Instances (G_i, k)

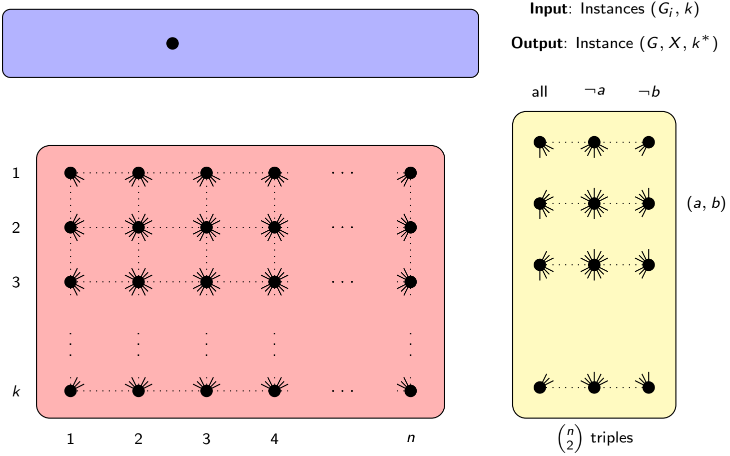
Output: Instance (G, X, k^*)



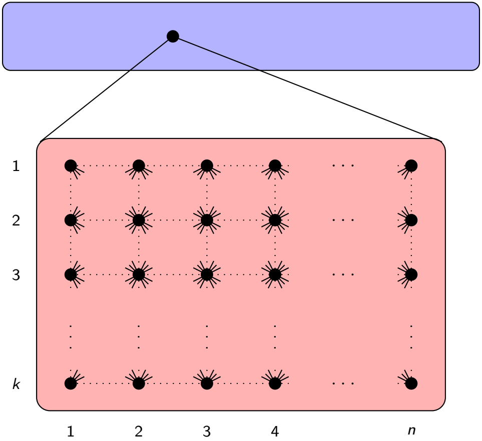
Cross-composing into CLIQUE/VC

Input: Instances (G_i, k)

Output: Instance (G, X, k^*)

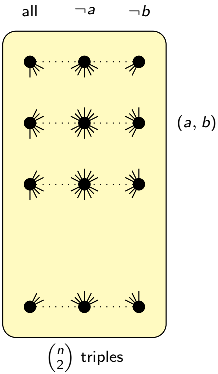


Cross-composing into CLIQUE/VC

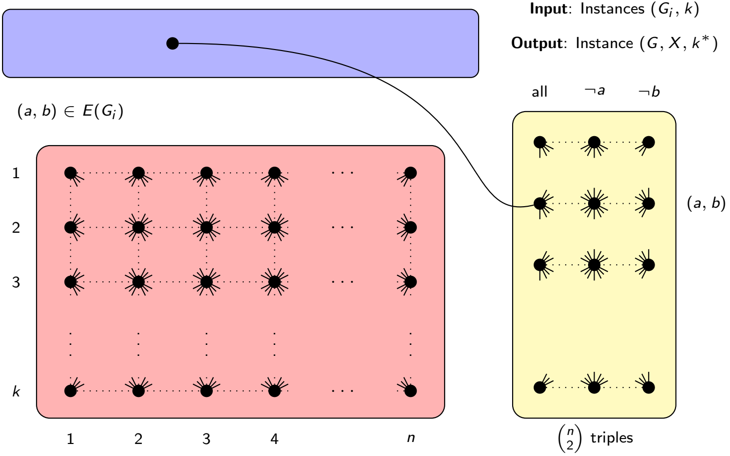


Input: Instances (G_i, k)

Output: Instance (G, X, k^*)



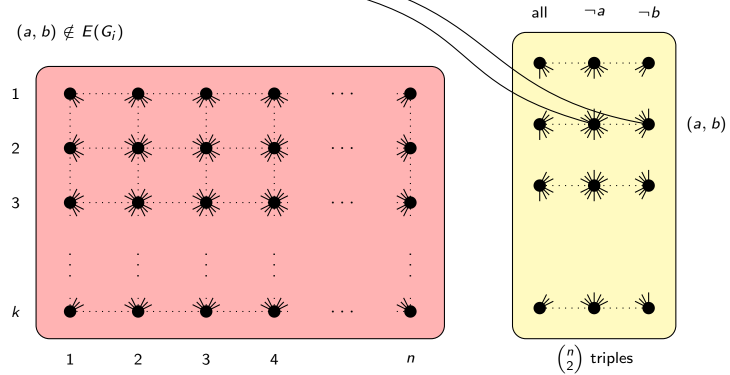
Cross-composing into Clique/VC



Cross-composing into CLIQUE/VC

Input: Instances (G_i, k)

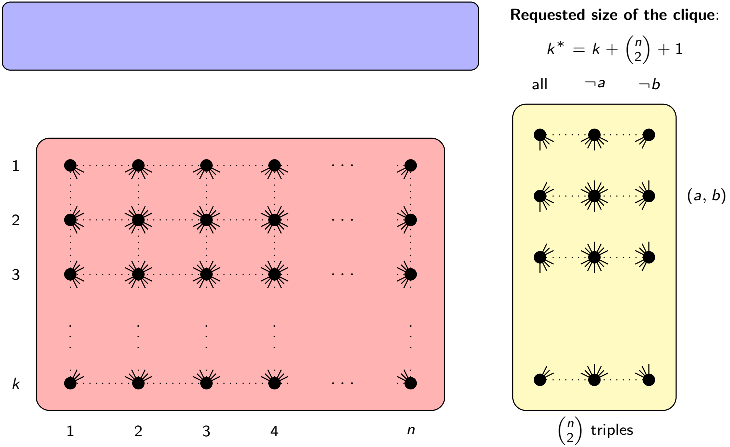
Output: Instance (G, X, k^*)



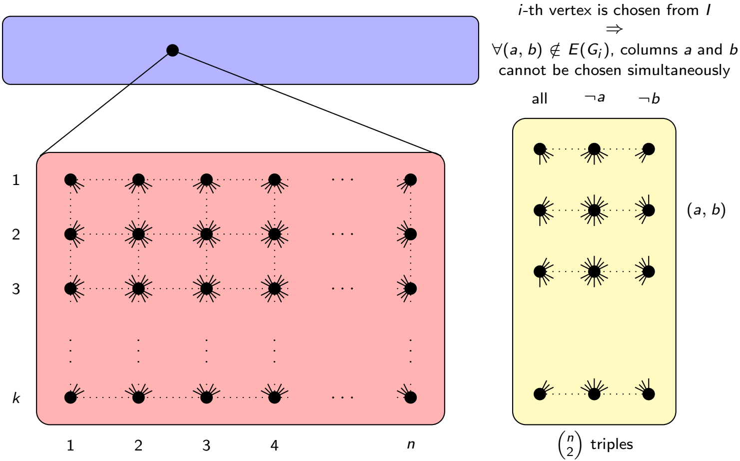
Cross-composing into CLIQUE/VC

Requested size of the clique:

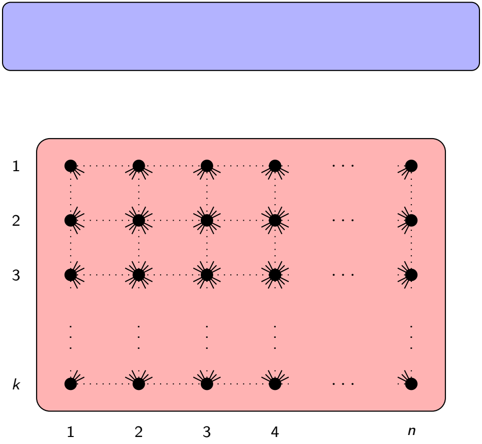
$$k^* = k + \binom{n}{2} + 1$$



Cross-composing into CLIQUE/VC

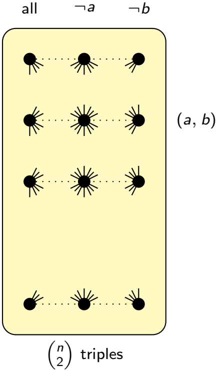


Cross-composing into Clique/VC



New parameter:

$|X| = kn + 3\binom{n}{2}$



Conclusion of the case study

- Making compositions is highly non-trivial.

Conclusion of the case study

- Making compositions is highly non-trivial.
- Requires good understanding of the problem.

Conclusion of the case study

- Making compositions is highly non-trivial.
- Requires good understanding of the problem.
- Intuitively, what we exploit is **choice**. We need to identify it, and build a technical construction on top of it.

Conclusion of the case study

- Making compositions is highly non-trivial.
- Requires good understanding of the problem.
- Intuitively, what we exploit is **choice**. We need to identify it, and build a technical construction on top of it.
- Often it requires a lot of gadgeteering...

Conclusion of the case study

- Making compositions is highly non-trivial.
- Requires good understanding of the problem.
- Intuitively, what we exploit is **choice**. We need to identify it, and build a technical construction on top of it.
- Often it requires a lot of gadgeteering...
- ... experience ...

Conclusion of the case study

- Making compositions is highly non-trivial.
- Requires good understanding of the problem.
- Intuitively, what we exploit is **choice**. We need to identify it, and build a technical construction on top of it.
- Often it requires a lot of gadgeteering...
- ... experience ...
- ... tricks that I did not mention ...

Conclusion of the case study

- Making compositions is highly non-trivial.
- Requires good understanding of the problem.
- Intuitively, what we exploit is **choice**. We need to identify it, and build a technical construction on top of it.
- Often it requires a lot of gadgeteering...
- ... experience ...
- ... tricks that I did not mention ...
- or clever ideas.

Weak compositions

- We have seen an $\mathcal{O}(k^2)$ kernel for Feedback Vertex Set (bitsize $\mathcal{O}(k^2 \log k)$).

Weak compositions

- We have seen an $\mathcal{O}(k^2)$ kernel for Feedback Vertex Set (bitsize $\mathcal{O}(k^2 \log k)$).
- Can we prove that a subquadratic kernel is unlikely?

Weak compositions

- We have seen an $\mathcal{O}(k^2)$ kernel for FEEDBACK VERTEX SET (bitsize $\mathcal{O}(k^2 \log k)$).
- Can we prove that a subquadratic kernel is unlikely?
- **YES**

Weak compositions

- We have seen an $\mathcal{O}(k^2)$ kernel for Feedback Vertex Set (bitsize $\mathcal{O}(k^2 \log k)$).
- Can we prove that a subquadratic kernel is unlikely?
- **YES**
- *Weak compositions*: proving lower bounds on kernelization complexity for problems that do have polynomial kernels.

Weak compositions

- We have seen an $\mathcal{O}(k^2)$ kernel for Feedback Vertex Set (bitsize $\mathcal{O}(k^2 \log k)$).
- Can we prove that a subquadratic kernel is unlikely?
- **YES**
- *Weak compositions*: proving lower bounds on kernelization complexity for problems that do have polynomial kernels.
- First results by Dell and van Melkebeek (STOC 2010), the framework here by (Dell, Marx; Hermelin, Wu; SODA 2012).

Back to Fortnow and Santhanam

- OR-distillation of $t = k^{1000}$ instances of size k into one instance with bitsize k^7 is unlikely.

Back to Fortnow and Santhanam

- OR-distillation of $t = k^{1000}$ instances of size k into one instance with bitsize k^7 is unlikely.
- Well, it should be unlikely even if we required any bitsize sublinear in t.

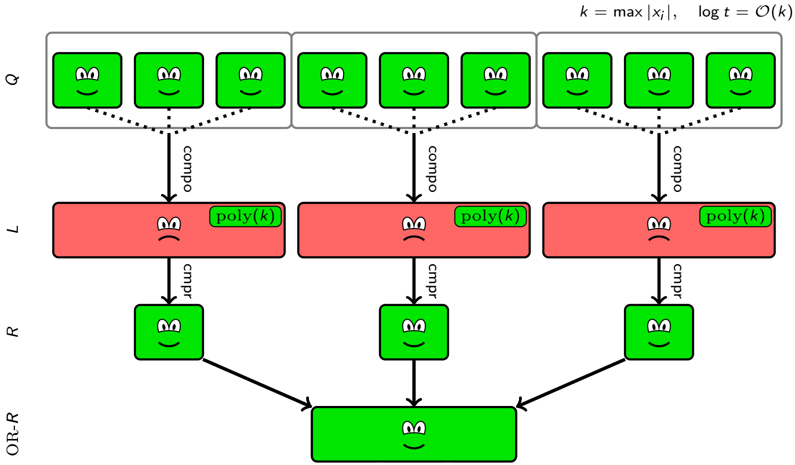
Back to Fortnow and Santhanam

- OR-distillation of $t = k^{1000}$ instances of size k into one instance with bitsize k^7 is unlikely.
- Well, it should be unlikely even if we required any bitsize sublinear in t.
- If one examines the proof, then one can exclude OR-distillation into bitsize $\mathcal{O}(t^{1-\epsilon} \cdot k^c)$, for any constants $\epsilon > 0$ and c.

Back to Fortnow and Santhanam

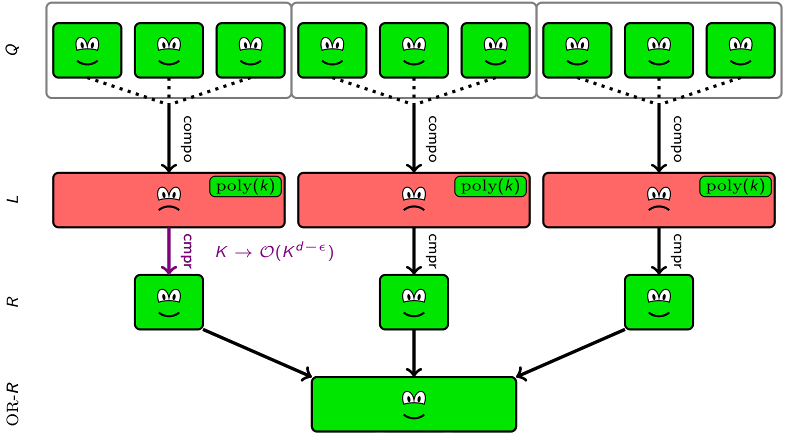
- OR-distillation of $t = k^{1000}$ instances of size k into one instance with bitsize k^7 is unlikely.
- Well, it should be unlikely even if we required any bitsize sublinear in t.
- If one examines the proof, then one can exclude OR-distillation into bitsize $\mathcal{O}(t^{1-\epsilon} \cdot k^c)$, for any constants $\epsilon > 0$ and c.
- Let's look again at the proof of the cross-composition Theorem.

Cross-composition proof, recap

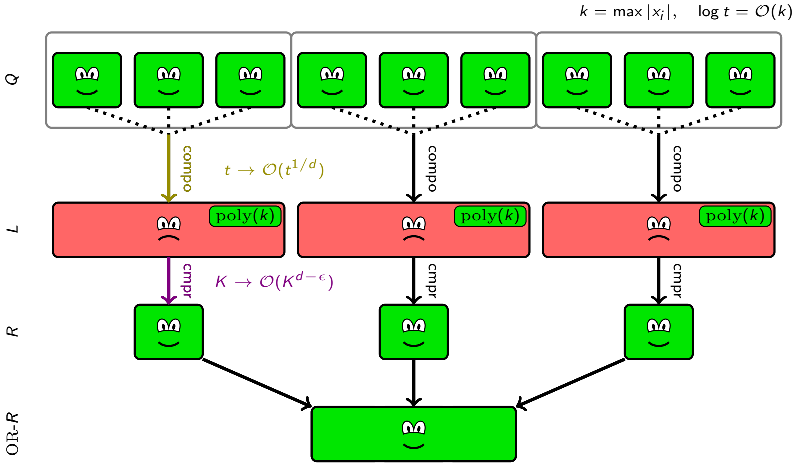


Cross-composition proof, recap

$k = \max |x_i|, \quad \log t = \mathcal{O}(k)$



Cross-composition proof, recap



Weak compositions, formally

- It seems that a composition with output bitsize dependence on t being $\mathcal{O}(t^{1/d})$ should exclude compression into bitsize $\mathcal{O}(k^{d-\epsilon})$.

Weak compositions, formally

- It seems that a composition with output bitsize dependence on t being $\mathcal{O}(t^{1/d})$ should exclude compression into bitsize $\mathcal{O}(k^{d-\epsilon})$.

Weak cross-composition

An unparameterized problem Q *weakly cross-composes* into a parameterized problem L, if there exists a polynomial equivalence relation $\mathcal{R}$, a real constant $d \geq 1$, and an algorithm that, given $\mathcal{R}$-equivalent strings $x_1, x_2, \ldots, x_t$, in time $\mathrm{poly}\left(t + \sum_{i=1}^{t} |x_i|\right)$ produces one instance $(y, k^\star)$ such that

- $(y, k^\star) \in L$ iff $x_i \in Q$ for at least one $i = 1, 2, \ldots, t$,
- $k^\star = t^{1/d+o(1)} \cdot \mathrm{poly}\left(\max_{i=1}^{t} |x_i|\right)$.

Weak cross-composition theorem

- Constant d will be called the *dimension* of the weak cross-composition.

Weak cross-composition theorem

- Constant d will be called the *dimension* of the weak cross-composition.

Weak cross-composition theorem

Suppose some **NP**-hard problem Q admits a weak cross-composition into L with dimension d. Suppose further that L admits a polynomial compression with bitsize $\mathcal{O}(k^{d-\epsilon})$, for some $\epsilon > 0$. Then $\mathbf{NP} \subseteq \mathbf{coNP}/\mathrm{poly}$.

Weak cross-composition theorem

- Constant d will be called the *dimension* of the weak cross-composition.

Weak cross-composition theorem

Suppose some **NP**-hard problem Q admits a weak cross-composition into L with dimension d. Suppose further that L admits a polynomial compression with bitsize $\mathcal{O}(k^{d-\epsilon})$, for some $\epsilon > 0$. Then $\mathbf{NP} \subseteq \mathbf{coNP}/\mathrm{poly}$.

- **Note**: Also called *cross-composition of bounded cost* by Bodlaender et al. (SIDMA, 2014).

Lower bound for Vertex Cover

- Using weak cross-composition we now prove that Vertex Cover does not admit a kernel with bitsize $\mathcal{O}(k^{2-\epsilon})$, for any $\epsilon > 0$. (unless...)

Lower bound for Vertex Cover

- Using weak cross-composition we now prove that Vertex Cover does not admit a kernel with bitsize $\mathcal{O}(k^{2-\epsilon})$, for any $\epsilon > 0$. (unless...)
- But it had a linear kernel!?

Lower bound for Vertex Cover

- Using weak cross-composition we now prove that Vertex Cover does not admit a kernel with bitsize $\mathcal{O}(k^{2-\epsilon})$, for any $\epsilon > 0$. (unless...)
- But it had a linear kernel!?
- Vertex Cover has a kernel with $2k$ vertices, which therefore requires $\mathcal{O}(k^2)$ bits to be encoded.

Lower bound for Vertex Cover

- Using weak cross-composition we now prove that Vertex Cover does not admit a kernel with bitsize $\mathcal{O}(k^{2-\epsilon})$, for any $\epsilon > 0$. (unless...)
- But it had a linear kernel!?
- Vertex Cover has a kernel with $2k$ vertices, which therefore requires $\mathcal{O}(k^2)$ bits to be encoded.
- **Crux**: choose an appropriate problem Q to start with.

Multicolored Biclique

Multicolored Biclique

Input: Bipartite graph H with bipartition $A \uplus B$, where $A = A_1 \uplus \ldots \uplus A_k$, $B = B_1 \uplus \ldots \uplus B_k$.

Question: Is there a biclique of size $K_{k,k}$ in H that contains one vertex from each A_i and each B_i?

Multicolored Biclique

Multicolored Biclique

Input: Bipartite graph H with bipartition $A \uplus B$, where $A = A_1 \uplus \ldots \uplus A_k$, $B = B_1 \uplus \ldots \uplus B_k$.

Question: Is there a biclique of size $K_{k,k}$ in H that contains one vertex from each A_i and each B_i?

- Multicolored Biclique is **NP**-hard even if each A_i and B_i has the same size.

Multicolored Biclique

Multicolored Biclique

Input: Bipartite graph H with bipartition $A \uplus B$, where $A = A_1 \uplus \ldots \uplus A_k$, $B = B_1 \uplus \ldots \uplus B_k$.

Question: Is there a biclique of size $K_{k,k}$ in H that contains one vertex from each A_i and each B_i?

- Multicolored Biclique is **NP**-hard even if each A_i and B_i has the same size.
- We provide a weak cross composition of dimension 2 from this version into Vertex Cover.

Multicolored Biclique

Multicolored Biclique

Input: Bipartite graph H with bipartition $A \uplus B$, where $A = A_1 \uplus \ldots \uplus A_k$, $B = B_1 \uplus \ldots \uplus B_k$.

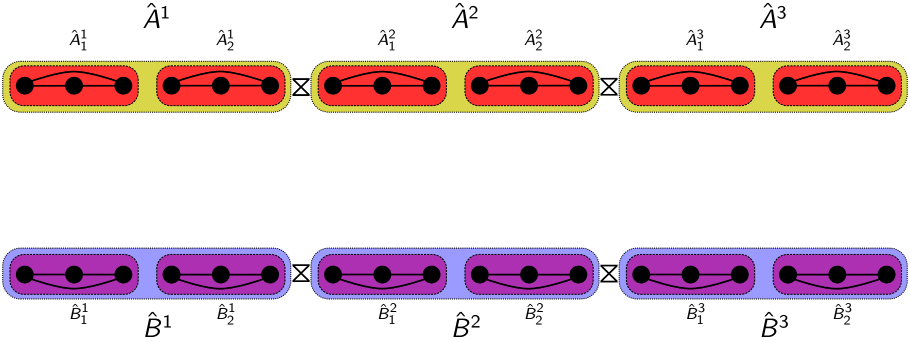
Question: Is there a biclique of size $K_{k,k}$ in H that contains one vertex from each A_i and each B_i?

- Multicolored Biclique is **NP**-hard even if each A_i and B_i has the same size.
- We provide a weak cross composition of dimension 2 from this version into Vertex Cover.
- **Assumptions**: all the input instances have the same k, each color class has size n in every input instance, $t = s^2$.

Composition

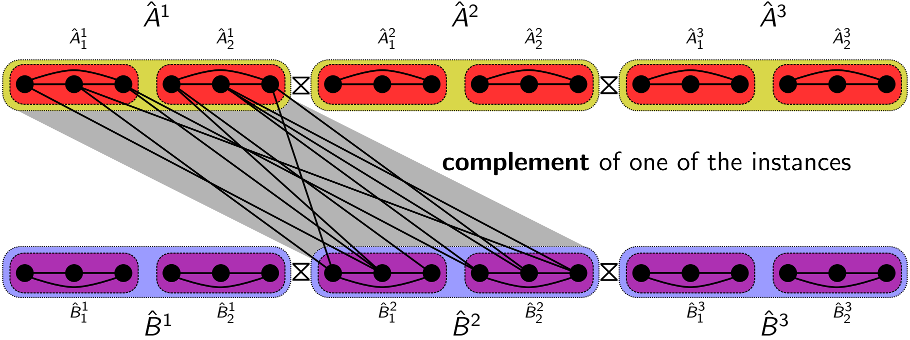
Create s copies of the left side and s copies of the right side.

$N = s \cdot 2kn$



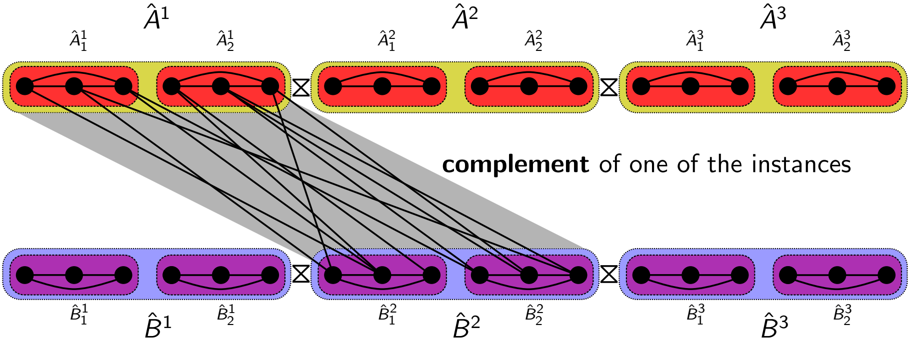
Composition

Embed s^2 instances into s^2 pairs of the sides.



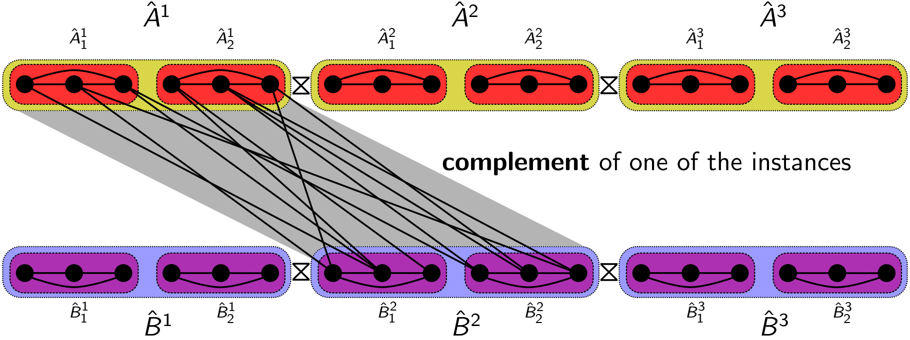
Composition

Ask for an independent set of size $2k$;
equivalently, a vertex cover of size $N - 2k$.



Composition

Edges on the sides ensure choosing one instance and respecting colors.
Edges originating in this instance ensure that the instance is solved.



Wrapping up

- Parameter in the output instance is $N - 2k = t^{1/2} \cdot 2kn - 2k$, which means that the weak composition has dimension 2.

Wrapping up

- Parameter in the output instance is $N - 2k = t^{1/2} \cdot 2kn - 2k$, which means that the weak composition has dimension 2.
- Hence, there is no kernel with $\mathcal{O}(k^{2-\epsilon})$ bits for VC.

Wrapping up

- Parameter in the output instance is $N - 2k = t^{1/2} \cdot 2kn - 2k$, which means that the weak composition has dimension 2.
- Hence, there is no kernel with $\mathcal{O}(k^{2-\epsilon})$ bits for VC.
- Reduction from VC to FVS: add a degree-2 vertex to every edge, thus creating a triangle.

Wrapping up

- Parameter in the output instance is $N - 2k = t^{1/2} \cdot 2kn - 2k$, which means that the weak composition has dimension 2.
- Hence, there is no kernel with $\mathcal{O}(k^{2-\epsilon})$ bits for VC.
- Reduction from VC to FVS: add a degree-2 vertex to every edge, thus creating a triangle.
- Hence also FVS does not admit a $\mathcal{O}(k^{2-\epsilon})$ kernel.

Wrapping up

- Parameter in the output instance is $N - 2k = t^{1/2} \cdot 2kn - 2k$, which means that the weak composition has dimension 2.
- Hence, there is no kernel with $\mathcal{O}(k^{2-\epsilon})$ bits for VC.
- Reduction from VC to FVS: add a degree-2 vertex to every edge, thus creating a triangle.
- Hence also FVS does not admit a $\mathcal{O}(k^{2-\epsilon})$ kernel.
- For more $\mathcal{O}(k^{d-\epsilon})$ lower bounds, see the book.

Further perspectives

- Compression vs. Kernelization

Further perspectives

- Compression vs. Kernelization
  - VC has kernel with $O(k)$ vertices and $O(k^2)$ edges. What about FVS?

Further perspectives

- Compression vs. Kernelization
  - VC has kernel with $O(k)$ vertices and $O(k^2)$ edges. What about FVS?
  - Is compositionality the only reason why polynomial kernelization is infeasible?

Further perspectives

- Compression vs. Kernelization
  - VC has kernel with $O(k)$ vertices and $O(k^2)$ edges. What about FVS?
  - Is compositionality the only reason why polynomial kernelization is infeasible?
  - Can we find a sensible problem where kernelization and compression are provable different?

Further perspectives

- Compression vs. Kernelization
  - VC has kernel with $O(k)$ vertices and $O(k^2)$ edges. What about FVS?
  - Is compositionality the only reason why polynomial kernelization is infeasible?
  - Can we find a sensible problem where kernelization and compression are provable different?
- Completeness theory for kernelization.

Further perspectives

- Compression vs. Kernelization
  - VC has kernel with $O(k)$ vertices and $O(k^2)$ edges. What about FVS?
  - Is compositionality the only reason why polynomial kernelization is infeasible?
  - Can we find a sensible problem where kernelization and compression are provable different?
- Completeness theory for kernelization.
  - Hermelin, Kratsch, Sołtys, Wahlström, Wu; IPEC 2013.

Further perspectives

- Compression vs. Kernelization
  - VC has kernel with $O(k)$ vertices and $O(k^2)$ edges. What about FVS?
  - Is compositionality the only reason why polynomial kernelization is infeasible?
  - Can we find a sensible problem where kernelization and compression are provable different?
- Completeness theory for kernelization.
  - Hermelin, Kratsch, Sołtys, Wahlström, Wu; IPEC 2013.
- Turing kernelization

Further perspectives

- Compression vs. Kernelization
  - VC has kernel with $O(k)$ vertices and $O(k^2)$ edges. What about FVS?
  - Is compositionality the only reason why polynomial kernelization is infeasible?
  - Can we find a sensible problem where kernelization and compression are provable different?
- Completeness theory for kernelization.
  - Hermelin, Kratsch, Sołtys, Wahlström, Wu; IPEC 2013.
- Turing kernelization
  - Turing kernel is a polynomial-time algorithm with an access to an oracle that resolves kernels.

Further perspectives

- Compression vs. Kernelization
  - VC has kernel with $O(k)$ vertices and $O(k^2)$ edges. What about FVS?
  - Is compositionality the only reason why polynomial kernelization is infeasible?
  - Can we find a sensible problem where kernelization and compression are provable different?
- Completeness theory for kernelization.
  - Hermelin, Kratsch, Sołtys, Wahlström, Wu; IPEC 2013.
- Turing kernelization
  - Turing kernel is a polynomial-time algorithm with an access to an oracle that resolves kernels.
  - How to show infeasibility of Turing kernelization?

Exercises

Exercise 15.4, all the remaining points.
Exercises 15.1 and 15.5.