

Algorithms and Networks

Kernelization

Hans Bodlaender and Stefan Kratsch

April 7, 2011

Introduction
Kernelization
Kernels for
Vertex Cover
Sunflowers
Kernels for
Graph Packing
Meta results and
lower bounds
Summary



1

This lecture

- ▶ in the previous lectures we were interested in exact algorithms
- ▶ in particular in algorithms for parameterized problems, running in time

$$\mathcal{O}(f(k) \cdot n^c)$$

so-called fixed-parameter tractable algorithms

- ▶ today we will look at how parameterized complexity can also be helpful for understanding preprocessing/data reduction for hard problems

there are two motivations for doing this...

Introduction
Kernelization
Kernels for
Vertex Cover
Sunflowers
Kernels for
Graph Packing
Meta results and
lower bounds
Summary



2

Motivation I: Preprocessing

- ▶ we have seen exponential-time algorithms for various problems (and there are many more)
- ▶ often simple reduction rules can be used to ease the argumentation, e.g., to make branching rules
- ▶ can these reduction rules provably shrink the input instance (in polynomial time), before we apply the costly exponential time algorithms?

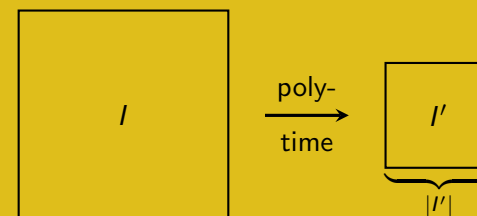
Introduction
Kernelization
Kernels for
Vertex Cover
Sunflowers
Kernels for
Graph Packing
Meta results and
lower bounds
Summary



3

Preprocessing

Preprocessing for an NP-hard language \mathcal{L}



such that $I \in \mathcal{L} \Leftrightarrow I' \in \mathcal{L}$ and $|I'| < |I|$

Fact: No NP-complete problem admits such a preprocessing unless $P = NP$.

Introduction
Kernelization
Kernels for
Vertex Cover
Sunflowers
Kernels for
Graph Packing
Meta results and
lower bounds
Summary



4

Why would it imply $P = NP$?

Fact: No NP-complete problem admits such a preprocessing unless $P = NP$.

- ▶ assume a polynomial-time preprocessing for $\mathcal{L} \in NP$, say $R : \Sigma^* \rightarrow \Sigma^*$ s.t. for all $I \in \Sigma^*$:
 - $I \in \mathcal{L}$ if and only if $R(I) \in \mathcal{L}$
 - $|R(I)| < |I|$
- ▶ this gives a polynomial-time algorithm for deciding $I \in \mathcal{L}$:

L-Algorithm(I)

1. while $|I| > 42$ replace I by $R(I)$
2. decide $I \in \mathcal{L}$ in constant time (as $|I| \leq 42$)

note: argument works of course for any decidable NP-hard problem



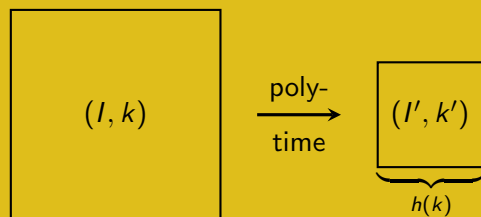
A lesson learned?

- ▶ unless $P = NP$ we cannot hope to shrink all instances of any NP-hard problem
- ▶ for each polynomial-time preprocessing algorithm, there must be hard instances that it cannot shrink
- ▶ we know already that parameterized complexity allows us to talk more easily about size and hardness (the parameter) of inputs
- ▶ can we define a notion of preprocessing by using a parameterized perspective?



Kernelization

Kernelization: a polynomial-time mapping:



such that (I, k) and (I', k') are equivalent.

polynomial kernelization: size $h(k) = poly(k)$



Motivation II: We already have kernels!

Lemma: A decidable parameterized problem is fixed-parameter tractable if and only if it admits a kernelization. [folklore]

recall: fixed-parameter tractable = has $\mathcal{O}(f(k) \cdot n^c)$ time algorithm

decidable: (roughly) there is an algorithm that solves the problem in "some" time depending only on the input size



Kernel \Rightarrow FPT

Proof: (fix some parameterized problem \mathcal{Q})

- ▶ assume polynomial-time kernelization for \mathcal{Q} with size bound h
- ▶ given input (x, k)
- ▶ apply kernelization to get (x', k') of size at most $h(k)$ (for some function h)
- ▶ have $(x, k) \in \mathcal{Q}$ if and only if $(x', k') \in \mathcal{Q}$
- ▶ solve (x', k') in time depending on its size (at most $h(k)$), say in time $f(h(k))$, as it is decidable
- ▶ together this gives a runtime of $\mathcal{O}(f(h(k)) + n^c)$



FPT \Rightarrow Kernel

Proof:

- ▶ assume FPT-algorithm for \mathcal{Q} of runtime $f(k) \cdot n^c$
- ▶ given input (x, k)
- ▶ run the algorithm for n^{c+1} steps
- ▶ if it finishes then we have solved the instance in polynomial time
- ▶ otherwise, it follows that

$$f(k) \cdot n^c > n^{c+1} \Rightarrow f(k) > n$$

- ▶ thus we either solve the instance, or we know that its size is bounded by $f(k)$
- ▶ hence we have a kernelization



Summing up the motivation

- ▶ kernelization is a way of formalizing preprocessing
- ▶ preprocessing is important to save runtime (and it is essentially for free: polynomial vs. exponential time)
- ▶ our FPT-algorithms already give kernelizations, however they are of exponential size
- ▶ e.g. VERTEX COVER(k) can be solved in $\mathcal{O}(1.27^k \cdot n^c)$, but that only gives a kernel of size $\mathcal{O}(1.27^k)$ by the lemma we just showed
- ▶ can we do better?



Outline

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary



Outline

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary



Universiteit Utrecht

13

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary

Kernelization

Let $Q \subseteq \Sigma^* \times \mathbb{N}$.

A **kernelization** for Q is an algorithm K which on input (x, k) computes in time $\mathcal{O}((|x| + k)^c)$ an instance (x', k') such that:

- ▶ $(x, k) \in Q$ if and only if $(x', k') \in Q$
- ▶ $|x'| \leq h(k)$ and $k' \leq h(k)$ for some computable function h

$h(k)$ is called the **size** of the kernelization K

K is a **polynomial kernelization** if $h(k)$ is polynomial in k



Universiteit Utrecht

14

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary

Reduction rules

- ▶ typically kernelizations are achieved by a set of reduction rules
- ▶ each rule can be performed in polynomial time
- ▶ if a rule applies to the current instance then it will modify the instance (slightly), getting an equivalent instance
- ▶ we must show that each rule is **safe**, i.e., it does not change whether the instance is YES or NO
- ▶ most often the parameter value is not increased

need to show: If none of the rules applies then the instance size is bounded by a function in the parameter.



Universiteit Utrecht

15

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary

Literature

- ▶ any of the three books on parameterized complexity (see the previous lecture)
- ▶ Guo & Niedermeier: "Invitation to data reduction and problem kernelization" 2007 (survey article)
- ▶ Bodlaender: "Kernelization: New Upper and Lower Bound Techniques", 2009 (survey of recent developments)



Universiteit Utrecht

16

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary

Outline

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary



Universiteit Utrecht

17

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary

Vertex Cover

Vertex Cover(k)

Input: A graph G and an integer k .

Parameter: k .

Output: Is there a set S of at most k vertices such that each edge has an endpoint in S ?

equivalently: such that $G - S$ is an independent set

- ▶ we had a $\mathcal{O}(1.47^k \cdot n^c)$ time algorithm (best known: $\mathcal{O}(1.27^k n^c)$)
- ▶ key: branching on including a vertex or all its neighbors



Universiteit Utrecht

18

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary

Vertex Cover: Kernelization by degree

input: (G, k)

Reduction Rule 1: if G has a vertex v of degree $> k$ then select it, i.e., delete v and decrease k by one

proof: there is no vertex cover of size $\leq k$ that does not include v , since it would include the $\geq k + 1$ neighbors

Reduction Rule 2: delete any isolated vertices from G



Universiteit Utrecht

19

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary

Vertex Cover: Kernelization by degree

Reduction Rule 3: if Rule 1 does not apply and G has more than k^2 edges then answer NO

proof:

- ▶ each vertex has degree at most k (by Rule 1)
- ▶ thus any set of at most k vertices can cover at most k^2 edges



Universiteit Utrecht

20

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary

Vertex Cover: Kernelization by degree

Lemma: If none of the rules applies to (G, k) then G has at most $2k^2$ vertices and at most k^2 edges.

- ▶ at most k^2 edges (Rules 1 & 3)
- ▶ at most $2k^2$ vertices can be endpoints of those edges
- ▶ no further isolated vertices (Rule 2)



Vertex Cover – Kernelization through crowns

- ▶ let us try to get a kernel with $\mathcal{O}(k)$ vertices for Vertex Cover
- ▶ the key role will be played by crown decompositions

A crown decomposition of G is tuple (H, C) (head H and crown C) such with $H, C \subseteq V(G)$, $H \cap C = \emptyset$ and such that:

1. C is an independent set
2. all neighbors of C are in H
3. there is a matching of H into C (i.e., each vertex of H has a private neighbor in C)



Why crowns are useful

Lemma: If (H, C) is a crown decomposition of G that there is a minimum vertex cover of G that includes H and excludes C .

proof:

- ▶ let S be a minimum vertex cover of G
- ▶ for each vertex $v \in H$ which is not in S , its private neighbor in C must be in S (to cover their edge)
- ▶ thus we can add the missing vertices of H and remove all vertices of C : getting $S' = (S \setminus C) \cup H$
- ▶ now $H \subseteq S'$ so all edges between H and C are covered
- ▶ hence no edges incident with C are uncovered by switching from S to S'
- ▶ clearly $|S'| \leq |S|$



How to find crowns

- ▶ we will only do so when G has more than $3k$ vertices (assuming an instance (G, k))
- ▶ first compute an independent set (C will be a subset of it)
- ▶ to do so compute a maximal matching $M \subseteq E(G)$
- ▶ if $|M| > k$ answer NO
- ▶ else let A be the endpoints of all edges in M , $|A| \leq 2k$
- ▶ let B denote the remaining vertices, $|B| > k$
- ▶ B is an independent set since M was maximal

Fact: If $|M| > k$ then G has no vertex cover of size k .

we show: While $|V(G)| > 3k$ we can find a crown, or we prove that G has no vertex cover of size at most k .



How to find crowns

Theorem: In a bipartite graph a maximum matching is as large as a minimum vertex cover. [König 1931]

- ▶ have an independent set B of size $> k$ and $A = V(G) \setminus B$
- ▶ compute a vertex cover X and a maximum matching M for the edges between A and B i.e. for the bipartite graph

$$H(A \cup B, \{\{u, v\} \mid \{u, v\} \in E(G) \wedge u \in A \wedge v \in B\})$$

- ▶ if $|M| = |X| > k$ then answer NO as H is a subgraph of G
- ▶ else ($|M| = |X| \leq k$) let
 - $B_X = X \cap B$ and $B_0 = B \setminus X \neq \emptyset$ (as $|B| > k$)
 - $A_X = X \cap A$ and $A_0 = A \setminus X$
- ▶ $X \not\subseteq B$: as $|B| > |X|$ there would be a $v \in B \setminus X$ whose neighbors are also not in X (we have no isolated vertices)
- ▶ thus $A_X = A \cap X \neq \emptyset$



How to find crowns

Claim: (A_X, B_0) is a crown.

- ▶ $|M| = |X|$ implies that each matching edge has exactly one endpoint in X (as none of them can have zero)
- ▶ thus M gives a matching of A_X into B_0
- ▶ by M being maximum it follows that no vertex of B_0 has a neighbor in A_0 (or we could extend M)
- ▶ B_0 is an independent set
- ▶ hence (A_X, B_0) is a crown

note: at most $3k$ vertices remain: $|A_0 \cup B_X| \leq 2k + k$



Vertex Cover – Kernelization through crowns

given input (G, k)

- ▶ if G has at most $3k$ vertices, then we are done
- ▶ else we can find a crown (A_X, B_0) which we may delete (by selecting A_X for the vertex cover and updating k)
- ▶ we get the instance $(G - (A_X \cup B_0), k - |A_X|)$
- ▶ $G - (A_X \cup B_0)$ has at most $3k$ vertices

this completes our kernelization



Vertex Cover – Kernelization by using LPs

- ▶ based on a (by now) well-known result of Nemhauser and Trotter (1975), far predating the notion of a kernel
- ▶ main idea: consider the relaxation of vertex cover (as an LP) where we may select vertices fractionally

$$\min \sum_{v \in V} x_v$$

$$x_u + x_v \geq 1, \text{ for each edge } \{u, v\}$$

$$0 \leq x_v \leq 1, \text{ for each } v \in V$$

- ▶ it can be showed that all extremal points (vertices/corners) of the polytope are half-integral

half-integral: (in this case) all variables are 0, $\frac{1}{2}$, or 1



Vertex Cover – Kernelization by using LPs

Theorem: Let x^* be an optimal solution to the vertex cover LP. There is an optimal (integer) vertex cover that includes each v with $x_v^* = 1$ and excludes each v with $x_v^* = 0$. [Nemhauser & Trotter 1975]

note: the original theorem addresses Independent Set **proof:**

- ▶ let V_1 contain all v with $x_v^* = 1$; let V_0 contain...
- ▶ all neighbors of V_0 are in V_1 (check the LP!)
- ▶ V_0 is an independent set
- ▶ if $|V_1| > |V_0|$ then setting all those variables to $\frac{1}{2}$ would be cheaper (check feasibility of this solution):

$$1 \cdot |V_1| + 0 \cdot |V_0| > \frac{1}{2} \cdot (|V_1| + |V_0|)$$

- ▶ (V_1, V_0) is a crown! (by Hall's Theorem)

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary



Universiteit Utrecht

29

Vertex Cover – Kernelization by using LPs

here is the resulting kernelization

- ▶ given an input (G, k)
- ▶ compute an optimal half-integral solution x^* for the LP
- ▶ if $\sum_{v \in V} x_v^* > k$ then there can also be no integer solution of cost at most k ; we return NO
- ▶ else, make the new instance $(G', k - |V_1|)$ with $G' = G - (V_1 \cup V_0)$
- ▶ G' has at most $2k$ vertices, as at most $2k$ variables in x^* can have value $\frac{1}{2}$

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary



Universiteit Utrecht

30

Outline

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary



Universiteit Utrecht

31

d-Hitting Set (k)

d-Hitting Set (k)

Input: A set \mathcal{U} and a family \mathcal{H} of subsets of \mathcal{U} each of size at most d , and an integer k .

Parameter: k .

Output: Is there $S \subseteq \mathcal{U}$, with $|S| \leq k$, such that $S \cap H \neq \emptyset$ for all $H \in \mathcal{H}$?

- ▶ generalizes Vertex Cover(k); that's the case when $d = 2$
- ▶ simple $\mathcal{O}(d^k n^c)$ time bounded search tree algorithm
- ▶ high-degree rule as for vertex cover does not hold
- ▶ we will use sunflowers to a similar effect

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

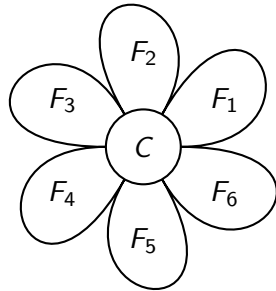
Summary



Universiteit Utrecht

32

Sunflowers



A **sunflower** of cardinality t consists of t sets F_1, \dots, F_t such that

$$F_i \cap F_j = C \text{ for all } i \neq j.$$

The set C is called the **core** of the sunflower.

note: t pairwise disjoint sets also form a sunflower



Sunflower Lemma

Sunflower Lemma: In every family of sets $\mathcal{H} \subseteq \binom{\mathcal{U}}{d}$ of size greater than $k^d \cdot d!$ one can find in polynomial time a sunflower of cardinality $k + 1$. [Erdős and Rado 1960]

note: the lemma is for the case that all sets have size $= d$

to apply it to sets of size **at most** d

- ▶ partition $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2 \cup \dots \cup \mathcal{H}_d$ according to size
- ▶ if any \mathcal{H}_i is larger than $k^d \cdot d!$ we get a sunflower (actually $> k^i \cdot i!$ suffices)
- ▶ else \mathcal{H} must be at most size $d \cdot k^d \cdot d!$



d-Hitting Set

d-Hitting Set (k)

Input: A set \mathcal{U} and a family \mathcal{H} of subsets of \mathcal{U} each of size at most d , and an integer k .

Parameter: k .

Output: Is there $S \subseteq \mathcal{U}$, with $|S| \leq k$, such that $S \cap H \neq \emptyset$ for all $H \in \mathcal{H}$?

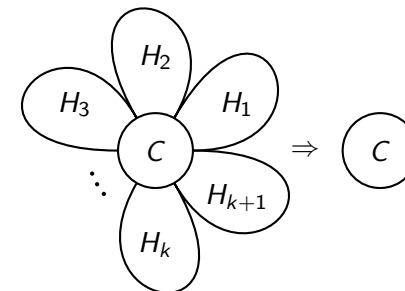
Idea for kernelization:

- ▶ while the instance is large, we can find a sunflower
- ▶ we *only* need to replace sunflowers in a good way



Sunflowers and Hitting Set

If we have a sunflower with $k + 1$ petals, we may as well keep only its core:



Picking only k elements there is no way to share an element with each set, without picking one from the core.

note: empty core \Rightarrow there is no solution of size k



d-Hitting Set – Sunflower-based kernel

given $(\mathcal{U}, \mathcal{H}, k)$

- ▶ if $|\mathcal{H}| \leq d \cdot k^d \cdot d!$ then stop and return current instance
- ▶ else use the Sunflower Lemma to find a sunflower of cardinality $k + 1$ in \mathcal{H}
- ▶ if the sunflower has an empty core then stop and return NO
- ▶ replace the sunflower by its core
- ▶ repeat

Introduction
Kernelization
Kernels for Vertex Cover
Sunflowers
Kernels for Graph Packing
Meta results and lower bounds
Summary



Universiteit Utrecht

37

d-Hitting Set – Wrap-up

- ▶ get a kernel with at most $|\mathcal{H}| \leq d \cdot k^d \cdot d! = \mathcal{O}(k^d)$
- ▶ we may discard elements of \mathcal{U} that are in no set, so $|\mathcal{U}| = \mathcal{O}(k^d)$
- ▶ the best known kernel has $|\mathcal{U}| = \mathcal{O}(k^{d-1})$ using a crown decomposition, but also has $|\mathcal{H}| = \mathcal{O}(k^d)$ (Abu-Khzam 2007)
- ▶ the bound on $|\mathcal{H}|$ is essentially optimal by a lower bound result

Introduction
Kernelization
Kernels for Vertex Cover
Sunflowers
Kernels for Graph Packing
Meta results and lower bounds
Summary



Universiteit Utrecht

38

Outline

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary

Introduction
Kernelization
Kernels for Vertex Cover
Sunflowers
Kernels for Graph Packing
Meta results and lower bounds
Summary



Universiteit Utrecht

39

Triangle Packing(k)

Triangle Packing(k)

Input: A graph G and an integer k .

Parameter: k .

Output: Does G contain at least k vertex-disjoint triangles?

triangle: a clique with three vertices

- ▶ Fellows et al.: “Finding k Disjoint Triangles in an Arbitrary Graph” (2004) gives a kernel with $\mathcal{O}(k^3)$ vertices
- ▶ Moser: “A Problem Kernelization for Graph Packing” (2009) gives a kernel with $\mathcal{O}(k^{h-1})$ vertices for packing k disjoint copies of any given graph H on h vertices
- ▶ let's have a look at some reduction rules

Introduction
Kernelization
Kernels for Vertex Cover
Sunflowers
Kernels for Graph Packing
Meta results and lower bounds
Summary



Universiteit Utrecht

40

Triangle Packing(k) – Reduction Rules

Introduction
Kernelization
Kernels for Vertex Cover
Sunflowers
Kernels for Graph Packing
Meta results and lower bounds
Summary

Reduction rule 1: Delete any edge or vertex that does not participate in a triangle.

proof:

- ▶ this will not increase the number of triangles
- ▶ any triangle packing for the original graph is still feasible after applying this rule



41

Triangle Packing(k) – Reduction Rules

Introduction
Kernelization
Kernels for Vertex Cover
Sunflowers
Kernels for Graph Packing
Meta results and lower bounds
Summary

Reduction rule 2: If the neighborhood of a vertex v contains at least $3k - 2$ vertex-disjoint edges then delete the vertex and decrease k by one:

$$(G, k) \mapsto (G - v, k - 1)$$

proof:

- ▶ deletion of one vertex removes at most one triangle from an optimal packing
- ▶ it is easy to see that a packing of $k - 1$ triangles into $G - v$ leaves at least one edge in the neighborhood of v unused (each triangle contains the endpoints of at most 3 edges)



42

Triangle Packing(k) – Kernelization

Introduction
Kernelization
Kernels for Vertex Cover
Sunflowers
Kernels for Graph Packing
Meta results and lower bounds
Summary

- ▶ compute a maximal packing W of vertex-disjoint triangles (start with $W = \emptyset$ and add triangles while possible)
- ▶ if $|W| \geq k$ then return YES
- ▶ else $|V(W)| \leq 3k - 3$ and (clearly) $G - V(W)$ contains no triangles
- ▶ compute a maximal matching Q if $G - V(W)$
- ▶ each edge in Q forms a triangle with at least one vertex of $V(W)$ (Rule 1), but each vertex does so for at most $3k - 3$ edges (Rule 2), hence

$$|Q| \leq (3k - 3) \cdot |V(W)| = \mathcal{O}(k^2)$$



43

Triangle Packing(k) – Kernelization

Introduction
Kernelization
Kernels for Vertex Cover
Sunflowers
Kernels for Graph Packing
Meta results and lower bounds
Summary

- ▶ observe that $I = V(G) \setminus (V(W) \cup V(Q))$ is an independent set (as Q is maximal matching in $G - V(W)$)
- ▶ each vertex of I must be in some triangle (Rule 1) which contains at least one vertex of W (maximal triangle packing W)
- ▶ make a bipartite graph H with vertex sets I and E_I :
 - E_I contains all edges that form a triangle with some vertex of I
 - $|E_I| \leq |V(W) \cup V(Q)| \cdot |V(W)| \in \mathcal{O}(k^3)$
 - in H there is an edge from $v \in I$ to $\{u, w\} \in E_I$ if $\{u, v, w\}$ is a triangle



44

Triangle Packing(k) – Kernelization

- ▶ compute a maximum matching and a minimum vertex cover of H to find a crown (as for Vertex Cover)
- ▶ in the crown:
 - edges in E_I that have private neighbors in I
 - vertices in I do not have other neighbors in E_I
- ▶ hence it corresponds to a set of edges, each with a private choice of a vertex to complete it to a triangle
- ▶ the vertices of I in the crown do not have other edges to form a triangle with
- ▶ we can delete the vertices of I from the crown that are not private neighbors

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary



Universiteit Utrecht

45

Triangle Packing(k) – Kernelization

- ▶ idea: if a maximum packing of triangles uses some of the deleted vertices, then we can replace them by private neighbors of the edges
- ▶ thus at most $\mathcal{O}(k^3)$ vertices of I remain, as at most $|E_I|$ many can be matched
- ▶ in total we have

$$|V(G)| = |V(W)| + |V(Q)| + |V(I')| = \mathcal{O}(k^3)$$

vertices left (this completes the kernelization)

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary



Universiteit Utrecht

46

Outline

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary



Universiteit Utrecht

47

Meta results for kernelization

- ▶ meta-result = result for a whole class of problems
- ▶ e.g. Bodlaender et al.: “(Meta) Kernelization” (2009)
- ▶ amongst others they show that certain problems on planar graphs admit linear kernels when (roughly):
 1. they can be expressed in a certain type of logic
 2. for every solution S , the structure of $G - S$ is “simple” (i.e. everything not close to S has bounded treewidth)
 3. they fulfill some additional requirement relating to dynamic programming on bounded treewidth for the problem
- ▶ extended to more general graph classes / other requirements

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary



Universiteit Utrecht

48

Lower bounds – Excluding polynomial kernels

- ▶ Bodlaender, Downey, Fellows, and Hermelin: “On Problems without Polynomial Kernels” (2008)
- ▶ under their “OR-distillation conjecture” certain problems do not admit polynomial kernels, e.g., $\text{LONG PATH}(k)$
- ▶ various follow-up work (see e.g. the mentioned survey) on: concrete problems, techniques, dichotomies
- ▶ work by Dell and van Melkebeek (2010) implies concrete lower bounds: e.g. size $\mathcal{O}(k^2)$ is optimal for $\text{VERTEX COVER}(k)$ (size vs. number of vertices)

note: work of Fortnow & Santhanam (2008) and Yap (1983) showed the conjecture to be true under standard assumptions



Outline

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

Kernels for Graph Packing

Meta results and lower bounds

Summary



Kernelization

- ▶ a formalization of polynomial time preprocessing for NP-hard problems
- ▶ typically uses a set of reduction rules
- ▶ every FPT problem has a (possibly exponential sized) kernel
- ▶ polynomial and smaller kernels are desired
- ▶ recent work provides techniques to show lower bounds

Open problems: e.g. polynomial kernels for $\text{ODD CYCLE TRANSVERSAL}(k)$, $\text{FEEDBACK VERTEX SET}(k)$, and $\text{MULTICUT}(k)$



Questions?

