This lecture

- in the previous lectures we were interested in exact algorithms
- in particular in algorithms for parameterized problems, running in time $O(f(k) \cdot n^c)$ 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...

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?

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

Preprocessing

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

\[ I \longrightarrow \text{poly-time} \longrightarrow I' \]

such that $I \in \mathcal{L} \Leftrightarrow I' \in \mathcal{L}$ and $|I'| < |I|$
Why would it imply \( P = NP \)?

**Fact:** No NP-complete problem admits such a preprocessing unless \( P = NP \).

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

L-Algorithm(I)

1. while \( |I| > 42 \) replace \( I \) by \( R(I) \)
2. decide \( I \in L \) 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:

\[
\begin{array}{ccc}
(l, k) & \xrightarrow{\text{poly-time}} & (l', k') \\
& \downarrow h(k) & \\
\end{array}
\]

such that \((l, k)\) and \((l', k')\) 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 \( O(f(k) \cdot n^c) \) time algorithm

**decidable**: (roughly) there is an algorithm that solves the problem in “some” time depending only on the input size
Kernel ⇒ FPT

Proof: (fix some parameterized problem $Q$)
- assume polynomial-time kernelization for $Q$
  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 $(x, k) \in Q$ if and only if $(x', k') \in Q$
- 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 $O(f(h(k)) + n^c)$

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 $O(1.27^k \cdot n^c)$,
  but that only gives a kernel of size $O(1.27^k)$ by the
  lemma we just showed
- can we do better?

FPT ⇒ Kernel

Proof:
- assume FPT-algorithm for $Q$ 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, if 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

Outline

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

\[
\begin{align*}
(x, k) & \in Q \text{ if and only if } (x', k') \in Q \\
|x'| & \leq h(k) \text{ and } k' \leq h(k) \text{ for some computable function } h
\end{align*}
\]

\( h(k) \) is called the \textit{size} of the kernelization \( K \).

\( K \) is a \textit{polynomial kernelization} if \( h(k) \) is polynomial in \( k \).

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 \textit{safe}, i.e., it does not change whether the instance is YES or NO
- most often the parameter value is not increased

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

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)
Outline

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

Vertex Cover

**Vertex Cover**

| **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 $O(1.47^k \cdot n^c)$ time algorithm  
  (best known: $O(1.27^k n^c)$)
- key: branching on including a vertex or all its neighbors

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 $G$

**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
**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)

**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
  $$H(A \cup B, \{\{u, v\} \mid \{u, v\} \in E(G) \land u \in A \land v \in B\})$$
- 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 $B_0 = B \setminus (X \setminus \emptyset)$ (as $|B| > k$)
  - $A_x = X \cap A$ and $A_0 = A \setminus X$
- $X \subseteq B$: 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 $A_x = A \cap X \neq \emptyset$

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
  $$\begin{align*}
  &\min \sum_{v \in V} x_v \\
  &x_u + x_v \geq 1, \text{ for each edge } \{u, v\} \\
  &0 \leq x_v \leq 1, \text{ for each } v \in V
  \end{align*}$$
- 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
**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)

**Outline**

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

**d-Hitting Set (k)**

**Input:** A set $U$ and a family $H$ of subsets of $U$ each of size at most $d$, and an integer $k$.

**Parameter:** $k$.

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

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

A **sunflower** of cardinality \( t \) consists of \( t \) sets \( F_1, \ldots, F_t \) such that
\[
F_i \cap F_j = C \quad \text{for all} \quad i \neq j.
\]
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 \( \mathcal{H} \subseteq \binom{U}{d} \) of size greater than \( kd \cdot d! \) 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 \( \mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2 \cup \ldots \cup \mathcal{H}_d \) according to size
- if any \( \mathcal{H}_i \) is larger than \( kd \cdot d! \) we get a sunflower (actually > \( k! \cdot i! \) suffices)
- else \( \mathcal{H} \) must be at most size \( d \cdot kd \cdot d! \)

---

**d-Hitting Set**

**d-Hitting Set (k)**

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

**Parameter:** \( k \).

**Output:** Is there \( S \subseteq U \), 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:

\[
H_1 \implies C
\]

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

**note:** empty core \( \implies \) **there is no solution of size \( k \)**
d-Hitting Set – Sunflower-based kernel

given $\langle U, H, k \rangle$

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

Outline

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

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 $O(k^3)$ vertices
- Moser: “A Problem Kernelization for Graph Packing” (2009) gives a kernel with $O(k^{h-1})$ vertices for packing $k$ disjoint copies of any given graph $H$ on $h$ vertices
- let’s have a look at some reduction rules
Triangle Packing(k) – Reduction Rules

**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

Triangle Packing(k) – Kernelization

- 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
  \[
  |Q| \leq (3k - 3) \cdot |V(W)| = O(k^2)
  \]

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)

Triangle Packing(k) – Kernelization

- 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)| \leq 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
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

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 \(O(k^3)\) vertices of \(I\) remain, as at most \(|E_i|\) many can be matched
- in total we have
  \[
  |V(G)| = |V(W)| + |V(Q)| + |V(I')| = O(k^3)
  \]
  vertices left (this completes the kernelization)

Outline

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

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
Lower bounds – Excluding polynomial kernels

- under their “OR-distillation conjecture” certain problems do not admit polynomial kernels, e.g., 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 $O(k^2)$ is optimal for 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

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 ODD CYCLE TRANSVERSAL(k), FEEDBACK VERTEX SET(k), and MULTICUT(k)

Questions?