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 better branching rules
- can such reduction rules provably shrink the input instance (in polynomial time), before we apply the costly exponential time algorithms?

Preprocessing

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

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

Fact: No NP-complete problem admits such a preprocessing unless $P = NP$. 
Why would it imply $P = NP$?

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

- assume a polynomial-time preprocessing for $L \in \text{NP}$, say $R : \Sigma^* \to \Sigma^*$ 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 $I \in L$:

  \[
  \text{L-Algorithm}(I) \\
  \begin{align*}
  &1. \text{ while } |I| > 42 \text{ replace } I \text{ by } R(I) \\
  &2. \text{ decide } I \in L \text{ in constant time (as } |I| \leq 42) 
  \end{align*}
  \]

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:

\[
(I, k) \xrightarrow{\text{poly-time}} (I', k') \xrightarrow{h(k)}
\]

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

**polynomial kernelization:** size $h(k) = \text{poly}(k)$

Motivation II: We already have kernels!

**Fact:** A parameterized problem is fixed-parameter tractable if and only if it is decidable and 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 $\Rightarrow$ 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 $O(f(h(k)) + n^c)$

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

Outline

Introduction

Kernelization

Kernels for Vertex Cover

Sunflowers

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 $O((|x| + k)^c)$ an instance $(x', k')$ such that:

1. $(x, k) \in Q$ if and only if $(x', k') \in Q$
2. $|x'| \leq h(k)$ and $k' \leq h(k)$ for a computable function $h$

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

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

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.
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

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 $O(1.47^k \cdot n^c)$ 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
**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

**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 $O(k)$ 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 $G$ then 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 \( k \).

**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: \(|A_0 \cup B_X| \leq 2k + k\)

---

**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 \( B_0 = B \setminus X \neq \emptyset \) (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 \( A_X = A \cap X \neq \emptyset \)

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
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.

Theorem: Let \(x^\ast\) be an optimal solution to the vertex cover LP. There is an optimal (integer) vertex cover that includes each \(v\) with \(x_v^\ast = 1\) and excludes each \(v\) with \(x_v^\ast = 0\).

[Nemhauser & Trotter 1975]

Note: the original theorem addresses Independent Set.

Proof:

- Let \(V_1\) contain all \(v\) with \(x_v^\ast = 1\); 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)

Here is the resulting kernelization:

- Given an input \((G, k)\)
- Compute an optimal half-integral solution \(x^\ast\) for the LP.
- If \(\sum_{v \in V} x_v^\ast > k\) 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 \(x^\ast\) can have value \(\frac{1}{2}\).

Outline

- Introduction
- Kernelization
- Kernels for Vertex Cover
- Sunflowers
- Meta results and lower bounds
- Summary
d-Hitting Set (k)

**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 \( O(k^d n^e) \) 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 \( H \subseteq \binom{U}{d} \) 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 \( H = H_1 \cup H_2 \cup \ldots \cup H_d \) according to size
- if any \( H_i \) is larger than \( k^d \cdot d! \) we get a sunflower (actually \( > k^d \cdot i! \) suffices)
- else \( H \) must be at most size \( d \cdot k^d \cdot d! \)

d-Hitting Set

**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 \( H \in H \)?

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:

\[
H_1, H_2, H_3, ..., H_{k+1} \Rightarrow C
\]

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

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

---

**d-Hitting Set – Sunflower-based kernel**

given \((U, H, k)\)

- 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

---

**d-Hitting Set – Wrap-up**

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

---

**Outline**

- Introduction
- Kernelization
- Kernels for Vertex Cover
- Sunflowers
- 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

Outline

- Introduction
- Kernelization
- Kernels for Vertex Cover
- Sunflowers
- 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 Directed Feedback Vertex Set($k$) and Multiway Cut($k$)
Questions?