4a. Parameterized intractability: the W-hierarchy

COMP6741: Parameterized and Exact Computation

Serge Gaspers

Semester 2, 2018

Contents

1 Reminder: Polynomial Time Reductions and NP-completeness 1

2 Parameterized Complexity Theory 2
  2.1 Parameterized reductions 3
  2.2 Parameterized complexity classes 3

3 Case study 5

4 Further Reading 6

1 Reminder: Polynomial Time Reductions and NP-completeness

Polynomial-time reduction

Definition 1. A polynomial-time reduction from a decision problem $\Pi_1$ to a decision problem $\Pi_2$ is a polynomial-time algorithm, which, for any instance of $\Pi_1$ produces an equivalent instance of $\Pi_2$. If there exists a polynomial-time reduction from $\Pi_1$ to $\Pi_2$, we say that $\Pi_1$ is polynomial-time reducible to $\Pi_2$ and write $\Pi_1 \leq_P \Pi_2$.

Important: $\leq_P$ is transitive.

New polynomial-time algorithms via reductions

Lemma 2. If $\Pi_1, \Pi_2$ are decision problems such that $\Pi_1 \leq_P \Pi_2$, then $\Pi_2 \in P$ implies $\Pi_1 \in P$.

A brief history

P vs. NP problem

P $\neq$ NP?

One of the seven famous Millennium Prize Problems stated by the Clay Mathematics Institute in 2000.

The hardest problem in NP

Satisfiability is one of the “hardest” problems in NP.


21 more “hardest” problems

3-SAT, 3-COLORING, INDEPENDENT SET, VERTEX COVER, …

NP-completeness

**Definition 3** (NP-hard). A decision problem \( \Pi \) is **NP-hard** if \( \Pi' \leq_p \Pi \) for every \( \Pi' \in \text{NP} \).

**Definition 4** (NP-complete). A decision problem \( \Pi \) is **NP-complete** if

1. \( \Pi \in \text{NP} \), and
2. \( \Pi \) is NP-hard.

**Proving NP-completeness**

**Lemma 5.** If \( \Pi \) is a decision problem such that \( \Pi' \leq_p \Pi \) for some NP-hard decision problem \( \Pi' \), then \( \Pi \) is NP-hard. 

If, in addition, \( \Pi \in \text{NP} \), then \( \Pi \) is NP-complete.

Method to prove that a decision problem \( \Pi \) is NP-complete:

1. Prove \( \Pi \in \text{NP} \)
2. Prove \( \Pi \) is NP-hard.

- Select a known NP-hard decision problem \( \Pi' \).
- Describe an algorithm that transforms every instance \( I \) of \( \Pi' \) to an instance \( r(I) \) of \( \Pi \).
- Prove that for each instance \( I \) of \( \Pi' \), we have that \( I \) is a Yes-instance of \( \Pi' \) \( \iff \) \( r(I) \) is a Yes-instance of \( \Pi \).
- Show that the algorithm runs in polynomial time.

2 Parameterized Complexity Theory

**Main Parameterized Complexity Classes**

- \( n \): instance size
- \( k \): parameter

- \( \text{P} \): class of problems that can be solved in \( n^{O(1)} \) time
- \( \text{FPT} \): class of parameterized problems that can be solved in \( f(k) \cdot n^{O(1)} \) time
- \( \text{W}[\cdot] \): parameterized intractability classes
- \( \text{XP} \): class of parameterized problems that can be solved in \( f(k) \cdot n^{g(k)} \) time ("polynomial when \( k \) is a constant")

\[ \text{P} \subseteq \text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \cdots \subseteq \text{W}[P] \subseteq \text{XP} \]

**Note:** We assume that \( f \) is computable and non-decreasing.

**Polynomial-time reductions for parameterized problems?**

A **vertex cover** in a graph \( G = (V, E) \) is a subset of vertices \( S \subseteq V \) such that every edge of \( G \) has an endpoint in \( S \).

<table>
<thead>
<tr>
<th><strong>Vertex Cover</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Graph ( G ), integer ( k )</td>
</tr>
<tr>
<td><strong>Parameter:</strong> ( k )</td>
</tr>
<tr>
<td><strong>Question:</strong> Does ( G ) have a vertex cover of size ( k )?</td>
</tr>
</tbody>
</table>

An **independent set** in a graph \( G = (V, E) \) is a subset of vertices \( S \subseteq V \) such that there is no edge \( uv \in E \) with \( u, v \in S \).

<table>
<thead>
<tr>
<th><strong>Independent Set</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Graph ( G ), integer ( k )</td>
</tr>
<tr>
<td><strong>Parameter:</strong> ( k )</td>
</tr>
<tr>
<td><strong>Question:</strong> Does ( G ) have an independent set of size ( k )?</td>
</tr>
</tbody>
</table>

- We know: \( \text{INDEPENDENT SET} \leq_p \text{VERTEX COVER} \)
- However: \( \text{VERTEX COVER} \in \text{FPT} \) but \( \text{INDEPENDENT SET} \) is not known to be in \( \text{FPT} \)
We will need another type of reductions

- Issue with polynomial-time reductions: parameter can change arbitrarily.
- We will want the reduction to produce an instance where the parameter is bounded by a function of the parameter of the original instance.
- Also: we can allow the reduction to take FPT time instead of only polynomial time.

2.1 Parameterized reductions

Parameterized reduction

Definition 6. A parameterized reduction from a parameterized decision problem $\Pi_1$ to a parameterized decision problem $\Pi_2$ is an algorithm, which, for any instance $I$ of $\Pi_1$ with parameter $k$ produces an instance $I'$ of $\Pi_2$ with parameter $k'$ such that

- $I$ is a Yes-instance for $\Pi_1$ $\iff$ $I'$ is a Yes-instance for $\Pi_2$,
- there exists a computable function $g$ such that $k' \leq g(k)$, and
- there exists a computable function $f$ such that the running time of the algorithm is $f(k) \cdot |I|^{O(1)}$.

If there exists a parameterized reduction from $\Pi_1$ to $\Pi_2$, we write $\Pi_1 \leq_{FPT} \Pi_2$.

Note: We can assume that $f$ and $g$ are non-decreasing.

New FPT algorithms via reductions

Lemma 7. If $\Pi_1, \Pi_2$ are parameterized decision problems such that $\Pi_1 \leq_{FPT} \Pi_2$, then $\Pi_2 \in FPT$ implies $\Pi_1 \in FPT$.

Proof sketch. To obtain an FPT algorithm for $\Pi_1$, perform the reduction and then use an FPT algorithm for $\Pi_2$ on the resulting instance.

2.2 Parameterized complexity classes

Boolean Circuits

Definition 8. A Boolean circuit is a directed acyclic graph with the nodes labeled as follows:

- every node of in-degree 0 is an input node,
- every node with in-degree 1 is a negation node ($\neg$), and
- every node with in-degree $\geq 2$ is either an AND-node ($\land$) or an OR-node ($\lor$).

Moreover, exactly one node with out-degree 0 is also labeled the output node.

The depth of the circuit is the maximum length of a directed path from an input node to the output node.

The weft of the circuit is the maximum number of nodes with in-degree $\geq 3$ on a directed path from an input node to the output node.

Example

A depth-3, weft-1 Boolean circuit with inputs $a, b, c, d, e$. 
**Weighted Circuit Satisfiability**
Given an assignment of Boolean values to the input gates, the circuit determines Boolean values at each node in the obvious way. If the value of the output node is 1 for an input assignment, we say that this assignment satisfies the circuit. The weight of an assignment is its number of 1s.

---

### Weighted Circuit Satisfiability (WCS)

| Input: A Boolean circuit $C$, an integer $k$ |
| Parameter: $k$ |
| Question: Is there an assignment with weight $k$ that satisfies $C$? |

**Exercise:** Show that Weighted Circuit Satisfiability $\in$ XP.

---

**WCS for special circuits**

**Definition 9.** The class of circuits $\mathcal{C}_{t,d}$ contains the circuits with weft $\leq t$ and depth $\leq d$.

For any class of circuits $\mathcal{C}$, we can define the following problem.

---

### WCS[$\mathcal{C}$]

| Input: A Boolean circuit $C \in \mathcal{C}$, an integer $k$ |
| Parameter: $k$ |
| Question: Is there an assignment with weight $k$ that satisfies $C$? |

---

**W classes**

**Definition 10 (W-hierarchy).** Let $t \in \{1, 2, \ldots \}$. A parameterized problem $\Pi$ is in the parameterized complexity class $W[t]$ if there exists a parameterized reduction from $\Pi$ to $\text{WCS}[\mathcal{C}_{t,d}]$ for some constant $d \geq 1$.

---

**Independent Set and Dominating Set**

**Theorem 11.** INDEPENDENT SET $\in$ W[1].

**Theorem 12.** DOMINATING SET $\in$ W[2].

**Recall:** A dominating set of a graph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that $N_G[S] = V$.

---

### Dominating Set

| Input: A graph $G = (V, E)$ and an integer $k$ |
| Parameter: $k$ |
| Question: Does $G$ have a dominating set of size at most $k$? |

---

**“Proof” by picture**

Parameterized reductions from INDEPENDENT SET to WCS[$\mathcal{C}_{1,3}$] and from DOMINATING SET to WCS[$\mathcal{C}_{2,2}$].

---

Setting an input node to 1 corresponds to adding the corresponding vertex to the independent set / dominating set.
W-hardness

**Definition 13.** Let \( t \in \{1, 2, \ldots \} \). A parameterized decision problem \( \Pi \) is \( W[t] \)-hard if for every parameterized decision problem \( \Pi' \) in \( W[t] \), there is a parameterized reduction from \( \Pi' \) to \( \Pi \). \( \Pi \) is \( W[t] \)-complete if \( \Pi \in W[t] \) and \( \Pi \) is \( W[t] \)-hard.

**Theorem 14.** **INDEPENDENT SET** is \( W[1] \)-complete.


**Theorem 15.** **DOMINATING SET** is \( W[2] \)-complete.


Proving W-hardness

To show that a parameterized decision problem \( \Pi \) is \( W[t] \)-hard:

- Select a \( W[t] \)-hard problem \( \Pi' \)
- Show that \( \Pi' \leq_{\text{FPT}} \Pi \) by designing a parameterized reduction from \( \Pi' \) to \( \Pi \)
  - Design an algorithm, that, for any instance \( I' \) of \( \Pi' \) with parameter \( k' \), produces an equivalent instance \( I \) of \( \Pi \) with parameter \( k \)
  - Show that \( k \) is upper bounded by a function of \( k' \)
  - Show that there exists a function \( f \) such that the running time of the algorithm is \( f(k') \cdot |I'|^{O(1)} \)

3 Case study

**Clique**

A clique in a graph \( G = (V, E) \) is a subset of its vertices \( S \subseteq V \) such that every two vertices from \( S \) are adjacent in \( G \).

<table>
<thead>
<tr>
<th>CLIQUE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
</tr>
<tr>
<td><strong>Parameter:</strong></td>
</tr>
<tr>
<td><strong>Question:</strong></td>
</tr>
</tbody>
</table>

- We will show that **CLIQUE** is \( W[1] \)-hard by a parameterized reduction from **INDEPENDENT SET**.

**Lemma 16.** **INDEPENDENT SET** \( \leq_{\text{FPT}} **CLIQUE** \).

**Proof.** Given any instance \( (G = (V, E), k) \) for **INDEPENDENT SET**, we need to describe an FPT algorithm that constructs an equivalent instance \( (G', k') \) for **CLIQUE** such that \( k' \leq g(k) \) for some computable function \( g \).

**Construction.** Set \( k' \leftarrow k \) and \( G' \leftarrow \overline{G} = (V, \{uv : u, v \in V, u \neq v, uv \notin E\}) \).

**Equivalence.** We need to show that \( (G, k) \) is a Yes-instance for **INDEPENDENT SET** if and only if \( (G', k') \) is a Yes-instance for **CLIQUE**.

(\( \Rightarrow \):) Let \( S \) be an independent set of size \( k \) in \( G \). For every two vertices \( u, v \in S \), we have that \( uv \notin E \). Therefore, \( uv \in E(\overline{G}) \) for every two vertices in \( S \). We conclude that \( S \) is a clique of size \( k \) in \( \overline{G} \).

(\( \Leftarrow \):) Let \( S \) be a clique of size \( k \) in \( \overline{G} \). By a similar argument, \( S \) is an independent set of size \( k \) in \( G \).

**Parameter.** \( k' \leq k \).

**Running time.** The construction can clearly be done in FPT time, and even in polynomial time.

**Corollary 17.** **CLIQUE** is \( W[1] \)-hard
4 Further Reading

