4a. Parameterized intractability: the W-hierarchy

COMP6741: Parameterized and Exact Computation

Serge Gaspers

School of Computer Science and Engineering, UNSW Sydney, Australia

19T3
Outline

1. Parameterized Complexity Theory
   - Parameterized reductions
   - Parameterized complexity classes

2. Case study

3. Further Reading
Outline

1. Parameterized Complexity Theory
   - Parameterized reductions
   - Parameterized complexity classes

2. Case study

3. Further Reading
Main Parameterized Complexity Classes

$n$: instance size  
$k$: parameter

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

\[
P \subseteq FPT \subseteq W[1] \subseteq W[2] \cdots \subseteq W[P] \subseteq XP
\]

**Note:** We assume that $f$ is computable and non-decreasing.
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 $S$.

### Vertex Cover

**Input:** Graph $G$, integer $k$  
**Parameter:** $k$  
**Question:** Does $G$ have a vertex cover of size $k$?

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 $u, v \in S$.

### Independent Set

**Input:** Graph $G$, integer $k$  
**Parameter:** $k$  
**Question:** Does $G$ have an independent set of size $k$?
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 $S$.

**Vertex Cover**

- **Input:** Graph $G$, integer $k$
- **Parameter:** $k$
- **Question:** Does $G$ have a vertex cover of size $k$?

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 $u, v \in S$.

**Independent Set**

- **Input:** Graph $G$, integer $k$
- **Parameter:** $k$
- **Question:** Does $G$ have an independent set of size $k$?

- **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 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.
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.
Outline

1. Parameterized Complexity Theory
   - Parameterized reductions
   - Parameterized complexity classes

2. Case study

3. Further Reading
Parameterized reduction

Definition 1

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 \textsc{Yes}-instance for $\Pi_1$ $\iff$ $I'$ is a \textsc{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 $\Pi_1 \leq_{\text{FPT}} \Pi_2$.

Note: We can assume that $f$ and $g$ are non-decreasing.
Lemma 2

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

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.
Outline

1 Parameterized Complexity Theory
   - Parameterized reductions
   - Parameterized complexity classes

2 Case study

3 Further Reading
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.
A depth-3, weft-1 Boolean circuit with inputs $a, b, c, d, e$. 
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 $\in \mathbf{XP}$. 
WCS for special circuits

Definition 4

The class of circuits $C_{t,d}$ contains the circuits with weft $\leq t$ and depth $\leq d$.

For any class of circuits $C$, we can define the following problem.

WCS[$C$]

Input: A Boolean circuit $C \in C$, an integer $k$
Parameter: $k$
Question: Is there an assignment with weight $k$ that satisfies $C$?
Definition 5 (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 $WCS[C_{t,d}]$ for some constant $d \geq 1$. 
Theorem 6

**Independent Set** \( \in W[1] \).

Theorem 7

**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 \( k \)?
Parameterized reductions from **INDEPENDENT SET** to \( \text{WCS}[c_{1,3}] \) and from **DOMINATING SET** to \( \text{WCS}[c_{2,2}] \).

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

Definition 8

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.
Definition 8

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 9 ([DF95b])

**Independent Set** is $W[1]$-complete.

Theorem 10 ([DF95a])

**Dominating Set** is $W[2]$-complete.
To show that a parameterized decision problem $\Pi$ is $W[t]$-hard:

- Select a $W[t]$-hard problem $\Pi'$
- Show that $\Pi' \leq_{FPT} \Pi$ 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 $f(k') \cdot |I'|^{O(1)}$
Outline

1. Parameterized Complexity Theory
   - Parameterized reductions
   - Parameterized complexity classes

2. Case study

3. Further Reading
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 $G$.

**Clique**

- **Input:** Graph $G = (V, E)$, integer $k$
- **Parameter:** $k$
- **Question:** Does $G$ have a clique of size $k$?

We will show that Clique is \textbf{W[1]}-hard by a parameterized reduction from \textbf{Independent Set}.
Lemma 11

\textsc{Independent Set} \leq_{\text{FPT}} \textsc{Clique}.

Proof.

Given any instance \((G = (V, E), k)\) for \textsc{Independent Set}, we need to describe an \textsc{FPT} algorithm that constructs an equivalent instance \((G', k')\) for \textsc{Clique} such that \(k' \leq g(k)\) for some computable function \(g\).
Clique is W[1]-hard

Lemma 11

\textbf{Independent Set} \leq_{\text{FPT}} \textbf{Clique}.

Proof.

Given any instance \((G = (V, E), k)\) for \textbf{Independent Set}, we need to describe an \textbf{FPT} algorithm that constructs an equivalent instance \((G', k')\) for \textbf{Clique} such that \(k' \leq g(k)\) for some computable function \(g\).

\textbf{Construction}. Set \(k' \leftarrow k\) and \(G' \leftarrow \overline{G} = (V, \{uv : u, v \in V, u \neq v, uv \notin E\})\).
Lemma 11
INDEPENDENT SET \leq_{\text{FPT}} \text{ 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.
Lemma 11

**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 \(\overline{G}\).
Lemma 11

**Independent Set** \( \leq_{\text{FPT}} \text{Clique} \).

**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 \(G\).
Lemma 11

**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.** \(k' \leq k\).
Clique is W[1]-hard

Lemma 11

\textbf{Independent Set} \leq_{FPT} \textbf{Clique}.

Proof.

Given any instance \((G = (V, E), k)\) for \textbf{Independent Set}, we need to describe an \textbf{FPT} algorithm that constructs an equivalent instance \((G', k')\) for \textbf{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 \textbf{Yes}-instance for \textbf{Independent Set} if and only if \((G', k')\) is a \textbf{Yes}-instance for \textbf{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 \textbf{FPT} time, and even in polynomial time.
Corollary 12

\textsc{Clique} is W[1]-hard
Outline

1 Parameterized Complexity Theory
   - Parameterized reductions
   - Parameterized complexity classes

2 Case study

3 Further Reading
Further Reading

- Chapter 13, *Fixed-parameter Intractability* in [Cyg+15]
- Chapter 13, *Parameterized Complexity Theory* in [Nie06]
- Elements of Chapters 20–23 in [DF13]
<table>
<thead>
<tr>
<th>Reference</th>
<th>Authors and Title</th>
</tr>
</thead>
</table>