Parameterized intractability: the W-hierarchy

Serge Gaspers

UNSW

Outline

Outline

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[·]: 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.

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.

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?

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.

> 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: INDEPENDENT SET $\leq_P$ VERTEX COVER
- However: VERTEX COVER $\in$ FPT but INDEPENDENT SET is not known to be in 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.

- 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 Π_1 to a parameterized decision problem Π_2 is an algorithm, which, for any instance I of Π_1 with parameter k produces an instance I' of Π_2 with parameter k' such that

- I is a YES-instance for Π_1 $\Leftrightarrow$ I' is a YES-instance for Π_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 Π_1 to Π_2, we write $\Pi_1 \leq_{\mathsf{FPT}} \Pi_2$.

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

New FPT algorithms via reductions

Lemma 2

If Π_1, Π_2 are parameterized decision problems such that $\Pi_1 \leq_{\mathsf{FPT}} \Pi_2$, then $\Pi_2 \in \mathsf{FPT}$ implies $\Pi_1 \in \mathsf{FPT}$.

Proof sketch.

To obtain an FPT algorithm for Π_1, perform the reduction and then use an FPT algorithm for Π_2 on the resulting instance. $\square$

Outline

Boolean Circuits

Definition 3

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 ≥ 2 is either an AND-node ($\wedge$) or an OR-node ($\vee$).

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 ≥ 3 on a directed path from an input node to the output node.
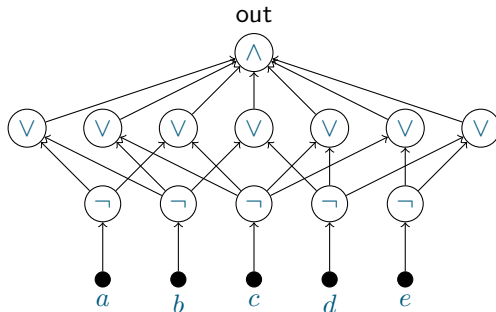
A depth-3, weft-1 Boolean circuit with inputs a, b, c, d, e.

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 4

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 5 (W-hierarchy)

Let $t \in \{1, 2, \dots\}$. A parameterized problem Π is in the parameterized complexity class W[t] if there exists a parameterized reduction from Π to WCS[$\mathcal{C}_{t,d}$] for some constant $d \geq 1$.

Independent Set and Dominating Set

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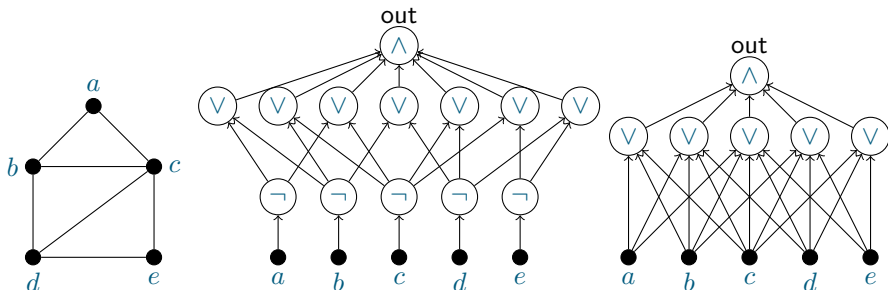> Input: A graph $G = (V, E)$ and an integer k
> Parameter: k
> Question: Does G have a dominating set of size at most k?

"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 8

Let $t \in \{1, 2, \dots\}$.

A parameterized decision problem Π is W[t]-hard if for every parameterized decision problem Π' in W[t], there is a parameterized reduction from Π' to Π.

Π is W[t]-complete if $\Pi \in$ W[t] and Π is W[t]-hard.

W-hardness

Definition 8

Let $t \in \{1, 2, \dots\}$.

A parameterized decision problem Π is W[t]-hard if for every parameterized decision problem Π' in W[t], there is a parameterized reduction from Π' to Π. Π is W[t]-complete if $\Pi \in$ W[t] and Π is W[t]-hard.

Theorem 9 (Downey and Fellows, 1995b)

INDEPENDENT SET *is* W[1]-*complete.*

Theorem 10 (Downey and Fellows, 1995a)

DOMINATING SET *is* W[2]-*complete.*

Proving W-hardness

To show that a parameterized decision problem Π is W[t]-hard:

- Select a W[t]-hard problem Π'
- Show that $\Pi' \leq_{\mathsf{FPT}} \Pi$ by designing a parameterized reduction from Π' to Π
  - Design an algorithm, that, for any instance I' of Π' with parameter k', produces an equivalent instance I of Π 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 $f(k') \cdot |I'|^{O(1)}$
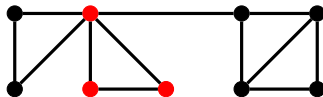
Outline

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.

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 W[1]-hard by a parameterized reduction from INDEPENDENT SET.

Clique is W[1]-hard

Lemma 11

Independent Set $\leq_{\mathsf{FPT}}$ 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.

Clique is W[1]-hard

Lemma 11

INDEPENDENT SET $\leq_{\textsf{FPT}}$ 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 $G' \leftarrow \overline{G} = (V, \{uv : u, v \in V, u \neq v, uv \notin E\})$.

Clique is W[1]-hard

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.

Clique is W[1]-hard

Lemma 11

INDEPENDENT SET $\leq_{\mathsf{FPT}}$ 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 $\overline{G}$.

Clique is W[1]-hard

Lemma 11

INDEPENDENT SET $\leq_{\mathsf{FPT}}$ 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.

Clique is W[1]-hard

Lemma 11

INDEPENDENT SET $\leq_{\mathsf{FPT}}$ 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

INDEPENDENT SET $\leq_{\mathsf{FPT}}$ 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. $\square$

Clique is W[1]-hard II

Corollary 12

CLIQUE *is* W[1]*-hard*

Outline

Further Reading

- Chapter 13, *Fixed-parameter Intractability* in (Cygan et al., 2015)
- Chapter 13, *Parameterized Complexity Theory* in (Niedermeier, 2006)
- Elements of Chapters 20–23 in (Downey and Fellows, 2013)

References I

Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh (2015). *Parameterized Algorithms*. Springer. DOI: 10.1007/978-3-319-21275-3.

Rodney G. Downey and Michael R. Fellows (1995a). "Fixed-Parameter Tractability and Completeness I: Basic Results". In: *SIAM Journal on Computing* 24.4, pp. 873–921.

– (1995b). "Fixed-Parameter Tractability and Completeness II: On Completeness for W[1]". In: *Theoretical Computer Science* 141.1&2, pp. 109–131.

– (2013). *Fundamentals of Parameterized Complexity*. Springer. DOI: 10.1007/978-1-4471-5559-1.

Rolf Niedermeier (2006). *Invitation to Fixed Parameter Algorithms*. Oxford University Press. DOI: 10.1093/ACPROF:OSO/9780198566076.001.0001.