5 Parameterized Hardness

27 May 2025

Prof. Dr. Sebastian Wild

CS627 (Summer 2025)
Philipps-Universität Marburg

version `2025-05-27 01:37`

Outline

5 Parameterized Hardness

How to prove $\notin$ FPT?

- For some problems, no algorithm seems to achieve fpt running time

How to prove $\notin$ FPT?

- For some problems, no algorithm seems to achieve fpt running time
- example: p-Clique

$\rightsquigarrow$ maybe no fpt algorithm can exist for p-Clique!

How to prove $\notin$ FPT?

- For some problems, no algorithm seems to achieve fpt running time
- example: p-CLIQUE

$\rightsquigarrow$ maybe no fpt algorithm can exist for p-CLIQUE!

- **Problem:** Certainly exists in case $\mathsf{P} = \mathsf{NP}$

$\rightsquigarrow$ strongest lower bound we can hope for will have to be conditional on $\mathsf{P} \neq \mathsf{NP}$

How to prove $\notin$ FPT?

- For some problems, no algorithm seems to achieve fpt running time
- example: p-CLIQUE

$\rightsquigarrow$ maybe no fpt algorithm can exist for p-CLIQUE!

- **Problem:** Certainly exists in case $\mathsf{P} = \mathsf{NP}$

$\rightsquigarrow$ strongest lower bound we can hope for will have to be conditional on $\mathsf{P} \neq \mathsf{NP}$

- Typical complexity-theory results:
  No algorithm has property X unless (more of less widely believed) complexity hypothesis Y fails.

5.1 Parameterized Reductions

FPT Reductions

Goal: Compare relative hardness of *parameterized* problems

$\rightsquigarrow$ Need a notion of reductions on parameterized problems

- to preserve (non)existence of fpt algorithms, need to keep small k

FPT Reductions

Goal: Compare relative hardness of *parameterized* problems

⇝ Need a notion of reductions on parameterized problems

- to preserve (non)existence of fpt algorithms, need to keep small k

Definition 5.1 (Parameterized Reduction)

Let (L_1, κ_1) and (L_2, κ_2) be two parameterized problems (over alphabets Σ_1 resp. Σ_2).
An *fpt-reduction* (*fpt many-one reduction*) from (L_1, κ_1) to (L_2, κ_2) is a mapping $A : \Sigma_1^\star \to \Sigma_2^\star$ so that for all $x \in \Sigma_1^\star$

1. (*equivalence*) $x \in L_1 \iff A(x) \in L_2$,
2. (*fpt*) A is computable by an fpt-algorithm (w.r.t. to κ_1), and
3. (*parameter-preserving*) $\kappa_2\big(A(x)\big) \le g\big(\kappa_1(x)\big)$ for a computable function $g : \mathbb{N} \to \mathbb{N}$.

does not depend $|x|$

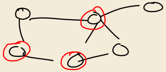
We then write $(L_1, \kappa_1) \le_{fpt} (L_2, \kappa_2)$.

$L_1 \le_p L_2$

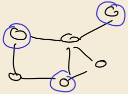
Not all reductions are fpt

Many reductions from classical complexity theory are **not** parameter preserving.

Recall:



VERTEXCOVER
Given: graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: $\exists V' \subset V \ : \ |V'| \leq k \ \wedge \ \forall \{u, v\} \in E : \big(u \in V' \vee v \in V'\big)$



INDEPENDENTSET
Given: graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: $\exists V' \subset V \ : \ |V'| \geq k \ \wedge \ \forall u, v \in V' : \{u, v\} \notin E$

Not all reductions are fpt

Many reductions from classical complexity theory are **not** parameter preserving.

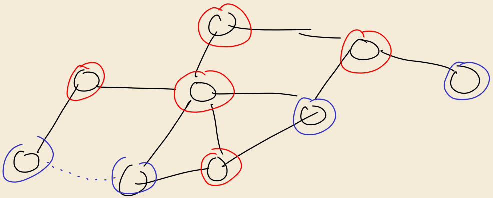
Recall:

VERTEXCOVER
Given: graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: $\exists V' \subset V \;:\; |V'| \leq k \;\wedge\; \forall \{u, v\} \in E : (u \in V' \vee v \in V')$

INDEPENDENTSET
Given: graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: $\exists V' \subset V \;:\; |V'| \geq k \;\wedge\; \forall u, v \in V' : \{u, v\} \notin E$

- We know: INDEPENDENTSET $\leq_p$ VERTEXCOVER:
  - Set $G' = G$ and $k' = |V(G)| - k$ (Complement of an indep. set must be a vertex cover, and vice versa!)

$k = 5$ $k' = 9 - 5 = 4$



<u>not</u> parameter preserving

Not all reductions are fpt

Many reductions from classical complexity theory are **not** parameter preserving.

Recall:

VertexCover
Given: graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: $\exists V' \subset V \ : \ |V'| \leq k \ \wedge \ \forall \{u, v\} \in E : \big(u \in V' \vee v \in V'\big)$

IndependentSet
Given: graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: $\exists V' \subset V \ : \ |V'| \geq k \ \wedge \ \forall u, v \in V' : \{u, v\} \notin E$

- We know: $\textsc{IndependentSet} \leq_p \textsc{VertexCover}$:
  - Set $G' = G$ and $k' = |V(G)| - k$ (Complement of an indep. set must be a vertex cover, and vice versa!)
- $\not\Rightarrow$ p-$\textsc{IndependentSet} \leq_{fpt} p$-$\textsc{VertexCover}$
  - Indeed, we know $\textsc{VertexCover} \in \mathsf{FPT}$, but IndependentSet probably isn't.

Not all reductions are fpt

Many reductions from classical complexity theory are **not** parameter preserving.

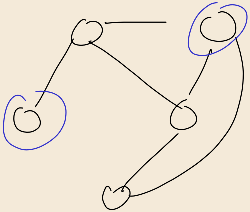
Recall:

VertexCover
Given: graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: $\exists V' \subset V \ : \ |V'| \leq k \ \wedge \ \forall \{u, v\} \in E : (u \in V' \vee v \in V')$

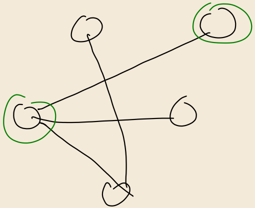
IndependentSet
Given: graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: $\exists V' \subset V \ : \ |V'| \geq k \ \wedge \ \forall u, v \in V' : \{u, v\} \notin E$

- We know: $\textsc{IndependentSet} \leq_p \textsc{VertexCover}$:
  - Set $G' = G$ and $k' = |V(G)| - k$ (Complement of an indep. set must be a vertex cover, and vice versa!)
- $\not\Rightarrow$ $p\text{-}\textsc{IndependentSet} \leq_{fpt} p\text{-}\textsc{VertexCover}$
  - Indeed, we know $\textsc{VertexCover} \in \mathsf{FPT}$, but IndependentSet probably isn't.
- But: $p\text{-}\textsc{IndependentSet} \leq_{fpt} p\text{-}\textsc{Clique}$ (and $p\text{-}\textsc{Clique} \leq_{fpt} p\text{-}\textsc{IndependentSet}$)
  - Set $G' = (V, \binom{V}{2} \setminus E)$ and $k' = k$ (Independent set iff clique in complement graph)

$k = 2$



$k' = 2$



5.2 Nondeterministic FPT: Para-NP

Parameterized NP: Non-deterministic NP

Good, so we have reductions.

If P corresponds to FPT . . . but what is the analogue for NP?

Parameterized NP: Non-deterministic NP

Good, so we have reductions.

If P corresponds to FPT . . . but what is the analogue for NP?

Definition 5.2 (para-NP)

The class para-NP consists of all parameterized decision problems that are solved by a *non-deterministic* fpt-algorithm.

$f(k) \cdot n^{O(1)}$

Parameterized NP: Non-deterministic NP

Good, so we have reductions.

If P corresponds to FPT ... but what is the analogue for NP?

Definition 5.2 (para-NP)

The class para-NP consists of all parameterized decision problems that are solved by a *non-deterministic* fpt-algorithm.

Some nice properties:

1. para-NP is closed under fpt-reductions. $A \in$ para-NP , $B \leq_{fpt} A \implies B \in$ para-NP
2. FPT = para-NP $\iff$ P = NP
3. an analogue for *kernalization* in FPT holds for para-NP

Parameterized NP: Non-deterministic NP

Good, so we have reductions.

If P corresponds to FPT . . . but what is the analogue for NP?

Definition 5.2 (para-NP)

The class para-NP consists of all parameterized decision problems that are solved by a *non-deterministic* fpt-algorithm. ◀

Some nice properties:

1. para-NP is closed under fpt-reductions.
2. FPT = para-NP $\iff$ P = NP
3. an analogue for *kernalization* in FPT holds for para-NP

Can define para-NP-hard and para-NP-complete similarly as for NP:

Definition 5.3 (para-NP-hard)

(L, κ) is para-NP-hard if $(L', \kappa') \leq_{fpt} (L, \kappa)$ for all $(L', \kappa') \in$ para-NP. ◀

Hello hardness, my old friend

Theorem 5.4 (para-NP-complete → NP-complete for finite parameter)
Let (L, κ) be a nontrivial ($\emptyset \neq L \neq \Sigma^\star$) parameterized problem that is para-NP-complete.
Then $L_{\leq d} = \{x \in L : \kappa(x) \leq d\}$ is NP-hard.
$\exists d$

The converse is essentially also true (using a generalization of kernelizations).

Proof: Let (L, κ) para-NP-complete

Let L' NP-complete $\Rightarrow$ $(L', \kappa_{one}) \in$ para-NP

$\kappa_{one}(x) = 1$

non-det. algorithm for L' is also non-det. fpt algorithm for (L', κ_{one})

para-NP-complete $\Rightarrow$ $(L', \kappa_{one}) \leq_{fpt} (L, \kappa)$ i.e. $\exists$ algo. A $x \in L' \iff A(x) \in L$

running time of A $f(\underbrace{\kappa_{one}(x)}_{1}) \cdot |x|^c$

$d = f(1) = O(1)$ ∎

running time of A is polynomial $|x|$

A maps L' to $L_{\leq d} = \{x \in L : \kappa(x) \leq d\}$

in poly time

$\Rightarrow L' \leq_p L_{\leq d} \Rightarrow L_{\leq d}$ NP-hard

para-NP-complete is too strict

Above Theorem means that many problems cannot be para-NP-complete!

For each of the following

- p-CLIQUE,
- p-INDEPENDENTSET
- p-DOMINATINGSET

bounding k by a **constant** d makes *polytime* algorithm possible.

FPT $f(k)\, n^c$

"XP" $n^{O(k)}$ for $\underline{\text{constant}}$ k is polytime

para-NP-complete is too strict

Above Theorem means that many problems cannot be para-NP-complete!

For each of the following

- p-CLIQUE,
- p-INDEPENDENTSET
- p-DOMINATINGSET

bounding k by a **constant** d makes *polytime* algorithm possible.

$\rightsquigarrow$ $L_{\le d}$ cannot be NP-complete for each of these

- but we rather expect them $\notin$ FPT

$\rightsquigarrow$ para-NP theory does not settle complexity status