5 Parameterized Hardness

27 May 2025

Prof. Dr. Sebastian Wild

CS627 (Summer 2025)
Philipps-Universität Marburg
version `2025-06-01 15:49 H`

Outline

5 Parameterized Hardness

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

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 $g : \mathbb{N} \to \mathbb{N}$.

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

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: $\exists V' \subset V \ : \ |V'| \le k \ \wedge \ \forall \{u, v\} \in E : \big(u \in V' \vee v \in V'\big)$

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

- We know: IndependentSet $\le_p$ 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-IndependentSet $\le_{fpt}$ p-VertexCover
  - Indeed, we know VertexCover $\in$ FPT, but IndependentSet probably isn't.
- But: p-IndependentSet $\le_{fpt}$ p-Clique (and p-Clique $\le_{fpt}$ p-IndependentSet)
  - Set $G' = (V, \binom{V}{2} \setminus E)$ and $k' = k$ (Independent set iff clique in complement graph)

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?

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.

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

Proof:

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.

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

- but we rather expect them $\notin$ FPT

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

5.3 Bounded Nondeterminism: W[P]

Bye bye, TM

para-NP is too large a class to have interesting complete problems
$\rightsquigarrow$ We must weaken the class. Unfortunately, TM inconvenient here.

Definition 5.5 (Nondeterministic RAM (NRAM), κ-restricted)

An NRAM M is a word-RAM with $w = O(\log n)$ with the additional operation to nondeterministically guess a number between 0 and a current register content.
An NRAM M that decides a parameterized problem (L, κ) is *κ-restricted* if on input $x \in \Sigma^\star$ with $n = |x|$ and $k = \kappa(x)$

1. it performs at most $f(k) \cdot p(n)$ steps,
2. at most $g(k)$ of them nondeterministic,
3. uses at most $f(k) \cdot p(n)$ registers, and
4. those never contain numbers larger than $f(k) \cdot p(n)$.

for computable functions f and g, and a polynomial p ◀

W[P]

Definition 5.6 (W[P])

The class W[P] is the set of all parameterized problems (L, κ) decidable by a κ-restricted NRAM. ◀

A first $\mathsf{W}[P]$-complete problem?

Define hardness and completeness for $\mathsf{W}[P]$ using $\leq_{fpt}$.

What could be the mother of all $\mathsf{W}[P]$-complete problems?

Some parameterized version of SAT? Parameter #variables does not work. (Why?)

- What can be guessed using k numbers in $[n]$?

$\rightsquigarrow$ A subset of variables of *size* k!

Weighted SAT

Definition 5.7 (Weighted Satisfiability)

Given: Boolean formula φ and integer $k \in \mathbb{N}$

Parameter: k

Recall: weight = #*true* variables

Question: $\exists$ satisfying assignment with weight $= k$?

Theorem 5.8 (p-WSAT(CIRC) is $\mathsf{W}[P]$-complete)

The weighted satisfiability problem for boolean **circuits** parameterized by the weight is $\mathsf{W}[P]$-complete.

Proof (Rough Idea):

5.4 Tail-nondeterministic NRAM

Tail-nondeterminism

Circuit satisfiability still too strong to show hardness of many interesting problems.
$\rightsquigarrow$ We must weaken the class *further*.

Definition 5.9 (tail-nondeterministic NRAM)

A κ-restricted NRAM M for a problem (L, κ) is called *tail-nondeterministic* if all nondeterministic steps occur only among the last $h(\kappa(x))$ steps. ◀

Definition 5.10 (W[1])

The class $\mathsf{W}[1]$ consists of all parameterized decision problems (L, κ) that are decided by a tail-nondeterministic κ-restricted NRAM. ◀

As before, define hardness and completeness for $\mathsf{W}[1]$ w.r.t. $\leq_{fpt}$.

Stop

Definition 5.11 (k-step Halting Problem)

Given: A nondeterministic (single-tape) Turing machine M, an input x and $k \in \mathbb{N}$ be given.
Parameter: k
Question: Does M accepts x after at most k computation steps? ◀

- M is part of input, so state space and tape alphabet are not fixed!

$\rightsquigarrow$ up to n different non-deterministic choices in *each* step. (n is size of encoding of M)

$\rightsquigarrow$ Trivial algorithm has to simulate up to n^{k+1} steps of M.

- Equivalent here to halting problem for $x = \varepsilon$, since we can hard-wire the given input into the states of a TM M' constructed from M.

W[1]-completeness

Theorem 5.12 (k-step halting problem W[1]-complete)

The k-step Halting Problem (for single-tape TM) parameterized by k is W[1]-complete. ◀

More natural problems?

Definition 5.13 (p-WSAT(2CNF))

Given: Boolean formula φ in 2-CNF and integer $k \in \mathbb{N}$
Parameter: k
Question: $\exists$ satisfying assignment with weight $= k$?

Theorem 5.14

p-WSAT(2CNF) is W[1]-complete.

Proof is a lengthy logic detour; omitted here. (See Flum, Grohe.)

Theorem 5.15

p-WSAT(2CNF$^-$) is W[1]-complete.

p-WSAT(2CNF$^-$) means: *all* literals *negated*.

p-Independent-Set is W[1]-complete

Theorem 5.16

p-INDEPENDENTSET is W[1]-complete.

Proof:

Partial Vertex Cover

Definition 5.17 (Partial Vertex Cover)

Given: graph $G = (V, E)$, target size $t \in \mathbb{N}$, threshold $k \in \mathbb{N}$
Parameter: k
Questions: $\exists C \subseteq V : |C| = k \wedge C$ covers at least t edges?

Theorem 5.18

p-PartialVertexCover is W[1]-hard.

Proof:

We show p-IndependentSet $\leq_{fpt}$ p-PartialVertexCover

Partial Vertex Cover [2]

Proof (continued):

Conclusion

- some care is needed to lift complexity theory to parameterized problems
- but: theory of W[1]-hardness and fpt-reductions is an effective framework to show that a parameterized problem is unlikely to admit an fpt algorithm
  - need new "gadgets" for fpt reductions
- further refinements possible (W[t] hierarchy)
  - p-DominatingSet is W[1]-hard, but likely $\notin$ W[1].
    (can be shown to be W[2]-complete and likely W[2] $\supsetneq$ W[1])
- W[1]-hardness suffices for negative results