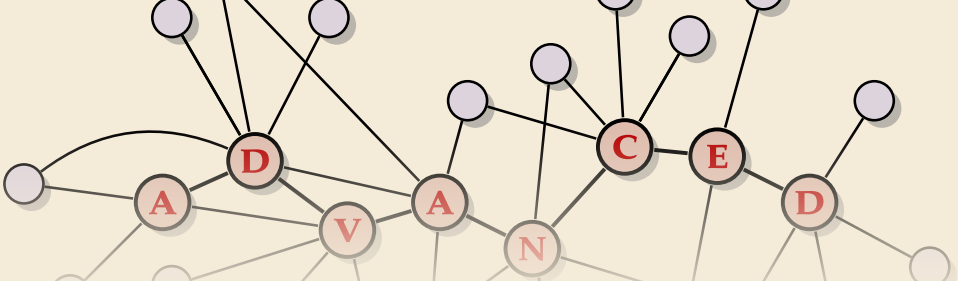


5 Parameterized Hardness

27 May 2025

Prof. Dr. Sebastian Wild

CS627 (Summer 2025)
Philipps-Universität Marburg
version 2025-05-27 16:50

Outline

5 Parameterized Hardness

5.1 Parameterized Reductions

5.2 Nondeterministic FPT: Para-NP

5.3 Bounded Nondeterminism: W[P]

5.4 Tail-nondeterministic NRAM

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 P = NP

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

$\rightsquigarrow$ 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*) $\kappa_2\big(A(x)\big) \leq g\big(\kappa_1(x)\big)$ for a computable function $g : \mathbb{N} \to \mathbb{N}$.

(does not depend |x|

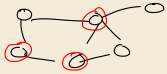
We then write $(L_1, \kappa_1) \leq_{fpt} (L_2, \kappa_2)$. ◀

$L_1 \leq_p L_2$

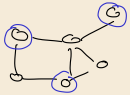
Not all reductions are fpt

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

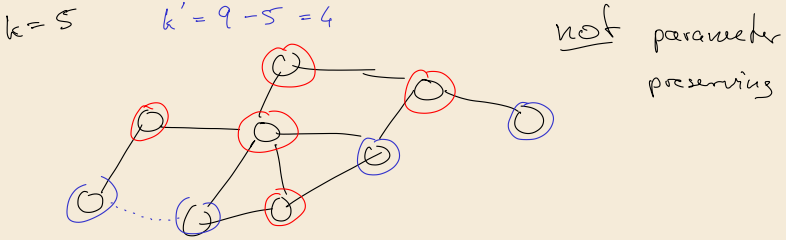
Many reductions from classical complexity theory are **not** parameter preserving.

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!)



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!)
- $\not\Rightarrow$ p-IndependentSet $\leq_{fpt}$ p-VertexCover
  - Indeed, we know VertexCover $\in$ FPT, but IndependentSet probably isn't.

Not all reductions are fpt

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

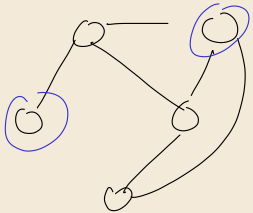
Recall:

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

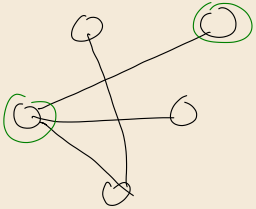
IndependentSet
Given: graph $G = (V, E)$ and $k \in \mathbb{N}$
Question: $\exists V' \subset V \ : \ |V'| \geq k \ \wedge \ \forall u, v \in V' : \{u, v\} \notin E$

- We know: 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!)
- $\not\Rightarrow$ p-IndependentSet $\leq_{fpt}$ p-VertexCover
  - Indeed, we know VertexCover $\in$ FPT, but IndependentSet probably isn't.
- But: p-IndependentSet $\leq_{fpt}$ p-Clique (and p-Clique $\leq_{fpt}$ p-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. $A \in \text{para-NP}$, $B \leq_{fpt} A \Rightarrow B \in \text{para-NP}$
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.

∃ 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. ∃ algo. A $\quad x \in L' \Leftrightarrow A(x) \in L$

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 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.

$\rightsquigarrow$ $L_{\leq d}$ 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
⇝ We must weaken the class. Unfortunately, TM inconvenient here.

We will restrict our study of parameterized complexity to the key tools to prove a problem is unlikely to allow an FPT algorithm.

But we do see hardness against classes. W[1]-hardness

We could say "p-Clique seems hard"
+ p-Clique $\leq_{fpt}$ A

Bye bye, TM

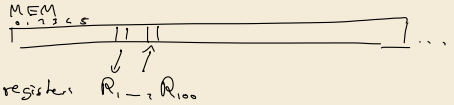
para-NP is too large a class to have interesting complete problems
⇝ We must weaken the class. Unfortunately, TM inconvenient here.

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~~ memory cells, and
4. those never contain numbers larger than $f(k) \cdot p(n)$.

for computable functions f and g, and a polynomial p



$w = O(\log n \cdot f(k))$

compared to para-NP: 2. is new
non deterministic step is more powerful

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.

◀

Is p-CLIQUE in W[P] ?

brute force: guess (nondeterministically) k-set S of vertices
(k steps, numbers between $[0..n)$)
check if S is a clique

n = #vertices

$O(n^2 + k + k^2)$ fpt time

n^2: to count vertices; k: guessing; k^2: κ-restricted

Yes.

A first W[P]-complete problem?

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

What could be the mother of all W[P]-complete problems?

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

A first W[P]-complete problem?

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

What could be the mother of all W[P]-complete problems?

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

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

A first W[P]-complete problem?

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

What could be the mother of all 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

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

Recall: weight = #*true* variables

◀

clearly in W[P] : same algorithm as for p-CLIQUE ✓

Weighted SAT

Definition 5.7 (Weighted Satisfiability)

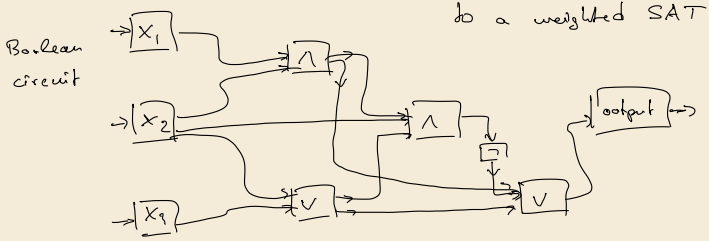
Given: Boolean formula φ and integer $k \in \mathbb{N}$
Parameter: k
Question: $\exists$ satisfying assignment with weight $= k$?

Recall: weight = #*true* variables

Theorem 5.8 (p-WSAT(CIRC) is W[P]-complete)

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

Proof (Rough Idea): Goal: "translate" any π-restricted NRAM to a weighted SAT instance



formulas are tree-circuits

Idea of reduction: $(L, \kappa) \in W[P]$ Claim $(L, \kappa) \leq_{fpt}$ p-WSAT(Circ)

$\Leftarrow$

$\exists$ NRAM M that decides L and is κ-restricted

Simulate M with a multi-tape TM M'

Simulate M' by a Boolean circuit C

"trick" on circuits gives circuit D s.t.

C satisfiable $\iff$ D is satisfiable with weight k

$W[P]$ contains p-CLIQUE etc. but it does not seem possible to show p-WSAT(Circuits) $\leq_{fpt}$ p-CLIQUE

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*.

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. ◀

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 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 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? ◀

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. ◀

Proof: ∘ "∈ W[1]"

Given TM M, input x, steps k

We construct an NRAM A

simulate A for k steps (and copy output)

x-restricted: k nondet. steps

tail-nondet: — " — can come last, copy of output $O(1)$

∘ "W[1]-hard" $(L, x) \in W[1]$

$\Rightarrow \exists$ NRAM A for L x-restricted, tail-nondeterministic

Given y as input to A, we try to simulate $A(y)$ via k-SHP.

reduction $(L, x) \leq_{fpt} k\text{-SHP}$

given y, run A for its deterministic part (fpt-running time)

encode "snapshot" of A at this point
(memory, register contents + program counter)

Construct nondet. TM M from A and snapshot
that simulates nondet. tail of A ($h(\kappa(y))$ steps)

output of reduction: $(M, h(\kappa(y)))$

fpt reduction: fpt-time
parameter-preserving

correctness by construction: A accepts y
$\Leftrightarrow$ M terminates
after $h(\kappa(y))$ steps

□

More natural problems?

3SAT NP-complete
2SAT $\in$ P

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 omitted.

Theorem 5.15

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

$(\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_3} \vee \overline{x_2}) \wedge \ldots$

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

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

Theorem 5.16

p-IndependentSet is W[1]-complete.

Proof:

- "$\in$ W[1]" — use k non-det guessing steps of NRAM (⇒ k-restricted) to guess subset $I \subseteq V$; check I is indep. set. Time $= O(k^2)$ ⇒ tail-nondet.
- "W[1]-hard" — prove $p\text{-WSAT}(2\text{CNF}^-) \leq_{fpt} p\text{-Indep.Set}$

  Given $\varphi = \bigwedge_{(i,j)\in I} (\overline{x_i} \vee \overline{x_j})$, k forced weight

  construct $G = \big([n], \{\{i,j\} : (i,j) \in I\big)$, $k' = k$

  '⇒' G has indep. set of size $k' = k$, then use set these variables to true and no clause contains 2 true variables ⇒ φ satisfied

  '⇐' If φ has a satisfying assignment of weight k

then the true-variables are an indep. set in G

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? ◀

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

Given graph $G = (V, E)$, k

$G' = (V', E')$

$t = k + |V|$

$$V' = V \cup \bigcup_{v \in V} \{ w_{v,i} : i = 1, \ldots, |V| - \deg(v) \}$$

$$E' = E \cup \bigcup_{v \in V} \{ \{v, w_{v,i}\} - " - \}$$

$\Rightarrow G'$ is $|V|$-regular (same degree)

Partial Vertex Cover [2]

Proof (continued):