26. Hardness Theory in Parameterized Complexity

Goal: Develop a theory of hardness, in analogue of NP-hardness, that is suitable to prove FPT-results unlikely.

Problem: Need a suitable notion of reduction and a class of hard problems.
In this class, we need a master problem.

26.1 Parameterized Reductions

Goal: Come up with a notion of reduction from L to L' that ensures the following:

If L' is computable in time $f(k) n^c$,
then L is computable in time $g(k) n^d$.

Observation: Each slice (fix k) can only reduce to a finite number of slices of L'.
Otherwise, the slice (the parameter) for L' will depend on (the size of) L's input.

Definition:
Consider languages L and L' from $\Sigma^* \times \mathbb{N}$.
We say L reduces to L' by a parameterized many-one reduction, if there are functions $k \mapsto k'$ and $k \mapsto k''$ (from $\mathbb{N}$ to $\mathbb{N}$) and a function $(x,k) \mapsto x'$ so that
$(x,k) \in L$ iff $(x',k') \in L'$.
Moreover, $(x,k) \mapsto x'$ has to be computable in time $k'' \cdot |x|^c$, for some $c \in \mathbb{N}$.

Lemma: If L' is FPT, so is L.

To give an example of a parameterized reduction, consider the following problems:

WEIGHTED CNF SAT:

Given: A Boolean formula F in CNF.

Parameter: $k \in \mathbb{N}$.

Question: Does F have a satisfying assignment with exactly k variables set to true?

WEIGHTED BINARY INTEGER PROGRAMMING:

Given: A binary matrix A and a binary vector b ($+ = \vee$, $\cdot = \wedge$).

Parameter: $k \in \mathbb{N}$.

Question: Does $Ax \geq b$ have a solution of weight (sum of 1s) k?

We call a formula monotone, if it does not contain negation symbols.
We call a formula antimonotone, if it does not contain negation symbols except for literals, which are all negated.

Theorem:

WEIGHTED MONOTONE CNF SAT reduces to WEIGHTED BINARY IP via a polynomial-time many-one reduction that happens to be a parameterized many-one reduction.

Proof:

Let (F, k) be an instance of WMCNFSAT.
Let $C_1, \ldots, C_p$ be the clauses of F and $x_1, \ldots, x_m$ be the variables.
We define the matrix

$$A := (a_{ij} \mid 1 \leq i \leq p,\ 1 \leq j \leq n)$$ // 1 row per clause, 1 column per variable.

by $a_{ij} := 1$ if $x_j \in C_i$
$a_{ij} := 0$ otherwise.

Let $b = 1$.

Claim: F has a satisfying assignment of weight k
iff $Ax \geq b$ has a k-weighted solution.

Complexity: The system $Ax \geq b$ is computable in polynomial time.
We can thus have $k'' = 1$.
The new parameter is $k' = k$. □

Note: · Classical many-one reductions are typically not parameterized many-one reductions.
· These reductions do not have enough structure
in the sense that we lose control of the parameter.

Example:
CNF-SAT has a polynomial-time many-one reduction to 3SAT, due to Karp '72.

· Consider formula F and one of its clauses $C = \{l_1, \dots, l_m\}$ with $m > 3$ literals.
We replace C by the following clauses to obtain a 3SAT instance:
$\{l_1, l_2, z_1\}\ \{\neg z_1, l_3, z_2\}\ \{\neg z_2, l_4, z_3\} \dots \{\neg z_{m-3}, l_{m-1}, l_m\}$.
Here, the $\{z_1, \dots, z_{m-3}\}$ are new variables.
Correctness of the construction is immediate by resolution.
Let $f(F)$ be the formula that we obtain by applying the trick to all clauses.

Then the size of $f(F)$ is bounded by $3 \cdot n \cdot q$
where n = size of the largest clause
q = number of clauses.

· The reduction is not a parameterized reduction
from WEIGHTED CNF SAT to WEIGHTED 3SAT.

The problem is that we introduce (and need for satisfiability) so many new variables.

Assume the original formula F has a weight-k satisfying assignment that makes precisely one literal l_j in C true.
To make f(F) true, we need to set

$z_1, \ldots, z_{j-2}$ to true.

So the weight of the satisfying assignment for f(F) not only depends on k, but also on j, the clause-size of F. □

Note:
- No parameterized reduction from WEIGHTED CNF SAT to WEIGHTED 3SAT is known.
- There is some evidence that such a reduction does not exist.
- If some NP-complete problems are not FPT, then it must be the case that some reductions cannot be lifted to parameterized reductions.
  Why? Because some NP-complete problems (like VERTEXCOVER) are FPT.

26.2 W[1] and an Analogue of Cook & Levin

Goal: Define a class that is to FPT like NP to P.

Observation:
- For NP, the master problem is computation in polynomial time for a non-deterministic TM.
- For the parameterized analogue, we bound the length of the computation.

SHORT TM ACCEPTANCE

Given: An NTM A and a word w.

Parameter: $k \in \mathbb{N}$.

Question: Does A have a computation accepting w in $\leq k$ steps.

Idea:
- If NTM acceptance is not tractable, then short NTM acceptance should not be fixed-parameter tractable.
- Support the claim by showing a number of problems of the same fixed-parameter complexity, none of them being FPT.

The hard class for FPT is defined via circuits.

Definition:
Let C be a circuit.
- The weft of C is defined to be the maximum number of large gates from any input to the output.
  A gate is large if its fan-in exceeds some predefined bound. (The value of the bound does not matter.)
- Let $\mathcal{F} = \{C_1, C_2, \ldots\}$ be a family of circuits.
  Associated with $\mathcal{F}$ is the parameterized language

$$L_{\mathcal{F}} := \{(C_i, k) \mid C_i \in \mathcal{F}, k \in \mathbb{N}, C_i \text{ has a weight-}k \text{ satisfying assignment}\}.$$

We write $L_{\mathcal{F}(t,h)}$ for the parameterized language associated with the family of all weft t depth h circuits.

Note: · The formulation L_F is not the language accepted by a non-uniform family of circuits.

· WEIGHTED 3SAT is included in $L_{F(1,3)}$.

Definition:

· A language L is in the class $W[t]$, if there is a parametrized many-one reduction to $L_{F(t,h)}$, for some h.

· Language L is $W[t]$-hard, if every problem in $W[t]$ admits a parameterized reduction to L.

In particular, $L_{F(t,-)} = \bigcup_{h \in \mathbb{N}} L_{F(t,h)}$ is $W[t]$-hard.

Our goal is to show the following.

Theorem (Analogue of Cook & Levin):

The following problems are complete for $W[1]$:

(1) WEIGHTED n-SAT for all $n \geq 2$.

(2) SHORT TM ACCEPTANCE.

Needs several steps, the first is a normalization for weft-1 circuits.

Recall: $L \in W[1]$ reduces to $L_{F(1,h)}$ for some h.

Show: We may assume the circuits in the family to have a simple shape:

↳ depth 3

↳ single output AND-gate

↳ AND-gate receives its arguments from OR-gates

↳ Fan-in of OR-gates bounded by some h'.

$\Rightarrow$ Circuit is isomorphic to CNF where clauses bounded by h'.

Definition:

Define $W[1,s]$ to be the problems from $W[1]$ reducible to $L_{F(s)}$.

Here, $F(s)$:= depth 3, wcft 1 normalized circuits where OR-gates on level 2 have fan-in $\leq s$.

Theorem: $W[1] = \bigcup_{s \in \mathbb{N}} W[1,s]$.

The next goal is to show that $W[1] = W[1,2]$.

- Call a circuit monotone, if it does not contain NOT-gates (Boolean formula with only positive literals).
- Call a circuit antimonotone, if all input variables are negated, and there are no further NOT-gates.
- Use (ANTI) MONOTONE $W[1]$ and (ANTI) MONOTONE $W[1,s]$.
  Note that MONOTONE $W[1,s]$ circuits have depth $= 2$.

Theorem: $W[1,s] = $ ANTIMONOTONE $W[1,s]$ for all $s \geq 2$.

Theorem: $W[1] = W[1,2]$.

Proof:
With the previous two theorems, it is sufficient to show

ANTI MONOTONE $[1,s] = W[1,2]$ for all $s \geq 2$.

Let C be an antimonotone $2[1,s]$ circuit for which we try to determine a weight-k input vector.
We show how to produce C' in $2^{O(k)}$

that accepts an input vector of weight

$$h' = h \cdot 2^h + \sum_{i=2}^{s} \binom{h}{i}$$

(iff C accepts an input of weight h).

Note that $\binom{h}{s} = 0$ if $s > h$, so h' is dependent only on h.

Idea:

Let C use variables $x_1, \dots, x_n$.

- The idea is to create new variables representing all sets of at least 2 and at most s variables.
  Let $A_1, \dots, A_p$ be an enumeration of all such sets.
  Note that the inputs (all negated) to each OR-gate g in C are precisely the elements of some A_i.
- The new input variable corresponding to A_i represents the fact that all variables whose negations are input to g have value true.
  Thus, in the construction of C' the OR-gate is replaced by the negation of the new collective input variable:

$$\neg x \vee \neg y \iff \neg(x \wedge y) \iff \neg \text{new}.$$

Construction:

- Introduce new variables
  (1) $v_1, \dots, v_p$ for the sets $A_1, \dots, A_p$
  (2) For each x_j (original input variable), introduce 2^h copies

$$x_{j,0}, \dots, x_{j,2^h-1}.$$

- We change the OR-gate to $\neg v_i$, as described above.
- We add additional OR-gates that provide an enforcement mechanism for the change of variable.

We need the following implications:

(1) $x_{j,r} \Rightarrow x_{j,r+1 \bmod 2^k}$,

for $j = 1, \dots, n$, $r = 0, \dots, 2^k - 1$ ($n \cdot 2^k$ implications).

The implications state that all copies of the variable x_j are equivalent.

(2) If $A_i \subseteq A_j$, we add $v_j \Rightarrow v_i$.

(3) For each membership $x_j \in A_i$, we add

$$v_i \Rightarrow x_{j,0}.$$

One may expect to find an implication of the form

$$\bigwedge_{x_j \in A_i} x_j \Rightarrow v_i. \qquad (*)$$

It is given implicitly by the high choice of k.

Since $\sum_{i=2}^{k} \binom{k}{i} < 2^k$, the only way to arrive at $k \cdot 2^k$ is to set k of the original variables to true (which have 2^k copies each).

We still need $\sum_{i=2}^{s} \binom{k}{i}$ of the (collective) new variables v_i.

If we choose them inconsistently (violating $(*)$), implication (3) will force us into further 2^k variables, thus exceeding k'.

The construction increases the size of the circuit by a factor of n^s for the v_i.
Since s is fixed (not part of the input), this is polynomial.
Moreover, we added 2^k copies for each variable.
Altogether, this yields a parameterized reduction. □

- The reader may note that setting $k' = k$ and not copying the variables would not have worked.
We would set all collector input variables to false, thereby making the circuit C' true.
Moreover, we pick k (set to true) of the original variables to achieve k'.
This does not yield a satisfying assignment for the original circuit.
- In a similar way, $k' = k + \sum_{i=2}^{s} \binom{k}{i}$ without variable copies does not work.
The critical v_i occurring in C' can be set to false to make the circuit true.
The uncritical v_i make the value k', implying variables that do not lead to satisfaction of C.

Example: 

, $k = 2$.

If satisfying assignment of C' with weight k' does not necessarily translate to a satisfying assignment of C with weight k.

Our proof of the modified Cook & Levin theorem relies on W[1]-completeness of two other problems.

INDEPENDENT SET

Given: Graph $G=(V,E)$.
Parameter: $k \in \mathbb{N}$
Question: Is there $V' \subseteq V$ with $|V'| = k$ so that $\forall u,v \in V': \{u,v\} \notin E$.

CLIQUE:

Given: Graph $G=(V,E)$.
Parameter: $k \in \mathbb{N}$.
Question: Is there $V' \subseteq V$ with $|V'| = k$ so that $G|_{V'}$ is a complete graph (clique).

Theorem:
(1) INDEPENDENT SET is W[1]-complete.
(2) CLIQUE is W[1]-complete.

Proof:
(1) Showing membership in W[1] is homework.
To show hardness, it is sufficient to show hardness for ANTIMONOTONE W[1,2] (combine the above two theorems).

Consider a Boolean formula F in CNF with clause size 2 and all literals negated.
We form a graph G_F with

- one vertex for each variable and
- having an edge between each pair of vertices corresponding to a clause.

Graph $G=$ has an independent set of size k
iff F has a satisfying assignment of weight k.

Indeed, if two variables from a set of variables were connected (and being in the set means being set to true), the corresponding clause would be false.

(2) Consider the complement of the given graph (edges where there were none before).
The complement has an independent set of size k
iff the original graph had a clique of size k. □

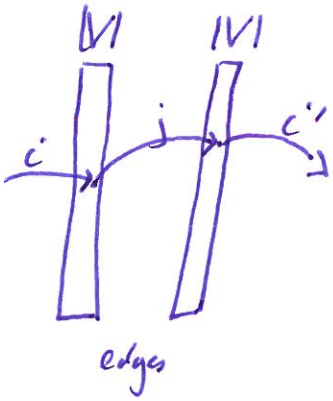
Theorem:
SHORT TM ACCEPTANCE is W[1]-complete.

Proof:
W[1]-hardness:
We give a reduction from CLIQUE.
- Let G be a graph for which we want to check whether it has a k-clique.
- We construct an NTM M so that
  M can reach an accepting state in $k' = f(k)$ steps
  iff G contains a k clique.

- M proceeds in two phases:

Phase 1: Write non-deterministically k symbols representing k-vertices of G into the first k type cells.
There are enough symbols for each vertex of G.

Phase 2: Make $\binom{k}{2}$ scans over the k cells, each checking for a pair i,j that $ij \in E$.
Needs $2|V|$ states in the TM for each scan:



Membership in $U[\tau]$:

Translate an instance of SHORT TM ACCEPTANCE to $L_{F(1,k)}$ for some fixed k.

Let $M = (\Sigma, Q, q_0, \delta, F)$ and let the bound be k.

We construct a circuit C of bounded depth (independent of M and k).

C will have:

↳ a single large output NAND-gate,

↳ the remaining gates small, and

↳ a weight k' input vector representing several things:

(1) i^{th} transition for M, for $1 \le i \le k$

(2) the head position at time i

(3) the state of M at time i

(4) the symbol in cell j at time i, $1 \le i, j \le k$.

Take $k' = k^2 + 3k$.
(4)

Construction of the circuit:

Input variables: $O(\underset{(1)}{k} \cdot \underset{(2)}{\delta} + \underset{(4)}{k}|Q| + \underset{(2)}{k^2}|\Sigma| + k^2)$.

↳ Set exactly one out of several choices to 1.

- $\neg x \vee \neg y$ for each pair of variables.

↳ No need to enforce that at least one is set to true.

Follows from $k' = k^2 + 3k$.

↳ Require consistency among the choices,

consistency requirements ensuring conjunctively

that we obtain a k-step computation of M.

Similar to Cook, Levin, and Ladner.

Each consistency check involves

only a bounded number of variables (computation is local).

↳ Fed "exactly one" and "consistency" small circuits

are fed into the large AND-gate. □