Advanced Algorithmics

Strategies for Tackling Hard Problems

Sebastian Wild
Markus Nebel

Lecture 3

2017-04-27

3 Fixed-parameter Tractability and Efficient Exponential Algorithms

Idea: Refine complexity analysis in two dimensions
(size of input 'n' ; parameter 'k')
and investigate the influence of various parameters.

Hope: We find parameter for which the problem can be
solved efficiently for small k.

The ideal case is that k is small in practice.

Drosophila: CNF-SAT

- size of clause $= k$ literals

  $k=2$ $2SAT \in P$

  $k=3$ $3SAT \in NP$-complete

- number of variables $\rightsquigarrow O(2^k \cdot n)$ brute-force method

  3SAT 1.49^k

- number of clauses m $\rightsquigarrow 1.24^m$
- length of formula #literals ℓ $\rightsquigarrow 1.08^\ell$
- 'weight' of formula : #1's in satisfying assignment
- structural properties of formula

MAX-CNF-SAT ; threshold k

Definition 3.1 (Parametrization)

Let Σ a (finite) alphabet. A *parametrization* (of $\Sigma^\star$) is a mapping $\kappa : \Sigma^\star \to \mathbb{N}$ that is poly-time computable. ◀

Definition 3.2 (Parametrized problem)

A *parameterized (decision) problem* is a pair (L, κ) of a language $L \subset \Sigma^\star$ and a parametrization κ of $\Sigma^\star$. ◀

Definition 3.3 (Canonical Parametrizations)

We can often specify a parametrized problem conveniently as a language of *pairs* $L \subset \Sigma^\star \times \mathbb{N}$ with

$$(x, k) \in L \;\wedge\; (x, k') \in L \quad \to \quad k = k'$$

using the *canonical parametrization* $\kappa(x, k) = k$. ◀

Examples

As before: Typically leave encoding implicit.
Naming convention: Add prefix p-SAT.

Definition 3.4 (p-SAT)

Given: formula boolean ϕ (same as before)
Parameter: number of variables
Question: Is there a satisfying assignment $v : [n] \to \{0, 1\}$? ◀

Definition 3.5 (p-Clique)

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

Definition 3.6 (Canonically Parametrized Optimization Problems)

Let $U = (\Sigma_I, \Sigma_O, L, L_I, M, cost, goal)$ be an optimization problem.
Then p-U denotes the *(canonically) parameterized (decision) problem* given by the threshold problem $Lang_U$.

Recall: $Lang_U$ is the set of pairs (x, k) of all instances $x \in L_I$ that have solutions that are ~~are~~ weakly "better" than k.

Examples:

- p-Clique
- p-Vertex-Cover
- p-Graph-Coloring
- ...

Naming convention for other parameters:
p-*clause*-CNF-SAT: CNF-SAT with parameter "number of *clauses*"

Examples of Running Times

only consider brute-force methods

- p-SAT : k variables , n length for formula

  2^k candidates each can be checked in linear time

  $\rightsquigarrow$ $O(2^k n)$ $\quad k = O(\log n)$ $\quad k = \log^c n$ $\rightsquigarrow 2^{\log^c n} = n^{\log(n)^{c-1}}$

- p-Cliques k threshold n vertices m edges

  $\binom{n}{k}$ candidates check $O(k^2)$ $\qquad \binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k!} \leq n^k$

  $\rightsquigarrow$ $O(n^k k^2)$ $\qquad k = O(1)$

- p-Vertex Cover $\binom{n}{k}$ candidates check $O(m)$

  $\rightarrow$ $O(n^k \cdot m)$

- p-Graph-Coloring k colors , n , m $\qquad k = 3 \rightsquigarrow$ NP-complete

  k^n candidates check $O(m)$ $\rightarrow$ $O(k^n \cdot m)$

3.1 Fixed-Parameter Tractability

Definition 3.7 (fpt-algorithm)
Let κ be a parametrization for $\Sigma^\star$.
A (deterministic) algorithm A (with input alphabet Σ) is a *fixed-parameter tractable algorithm (fpt-algorithm)* w.r.t. κ if its running time on $x \in \Sigma^\star$ with $\kappa(x) = k$ is at most

$$f(k) \cdot p(|x|) \quad = \quad \mathcal{O}\big(f(k) \cdot |x|^c\big)$$

where p is a polynomial of degree c and f is an arbitrary computable function. ◀

Definition 3.8 (FPT)
A parametrized problem (L, κ) is *fixed-parameter tractable* if there is an fpt-algorithm that decides it.
The complexity class of all such problems is denoted by $\mathcal{FPT}$. ◀

Intuitively, $\mathcal{FPT}$ plays the role of $\mathcal{P}$.

Theorem 3.9 (p-variables-SAT is FPT)

p-variables-SAT $\in \mathcal{FPT}$.

$$\underset{\text{p-SAT}}{\shortparallel}$$

brute-force method does the trick

$$\underbrace{2^k}_{f(k)} \, p(n)$$

$\square$

. . . but #variables not usually small

Theorem 3.10 (k never decreases → FPT)

Let $g : \mathbb{N} \to \mathbb{N}$ weakly increasing, unbounded and computable, and κ a parametrization with

$$\forall x \in \Sigma^\star : \kappa(x) \geq g(|x|).$$

Then $(L, \kappa) \in \mathcal{FPT}$ for *any* decidable L.

g weakly increasing: $\quad n \leq m \;\to\; g(n) \leq g(m)$
g unbounded: $\quad \forall t\ \exists n \;:\; g(n) \geq t$

Proof: L decidable $\leadsto$ $\exists T$: $x \overset{?}{\in} L$ decidable in $\leq T(|x|)$ steps

w.l.o.g. T weakly increasing

$T(|x|) \geqslant |x|$

Idea: hide $T(|x|)$ in $f(k)$ part of fpt-running time bound

$$h(n) = \begin{cases} \max\{m \in \mathbb{N} : g(m) \leq n\} & n \geq g(1) \\ 1 & \text{otherwise} \end{cases}$$

(1) s weakly increasing and unbounded

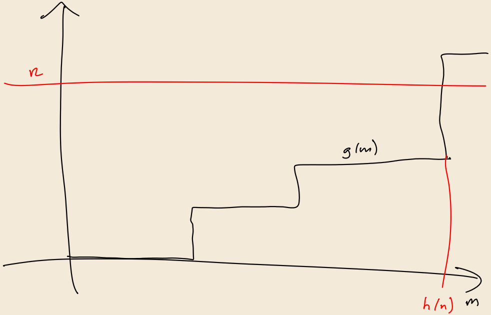
$\Rightarrow h$ well-defined

(2) h weakly increasing

(3) s computable

$\Rightarrow$ h computable

(4) $h(g(n)) \geq n$



time to decide $x \overset{?}{\in} L$

$$T(|x|) \underset{\substack{T \text{ incr.} \\ (4)}}{\leq} T(h(\underbrace{g(|x|)})) \underset{T,h \text{ incres.}}{\leq} T(h(k)) =: f(k) \qquad \square$$

$k = s(x)$

$\mathcal{P} \mathrel{\hat{\approx}} \mathcal{FPT}$

Some problems seem $\notin \mathcal{FPT}$; but how to prove that.

$\Rightarrow$ Reductions, hardness

3.2 Parametrized Reductions and Hardness

$L_1 \leq_p L_2 \qquad x \in L_1 \leftrightarrow A(x) \in L_2 \qquad$ '$A \in P$'

Definition 3.11 (Parametrized Reduction)

Let (L_1, κ_1) and (L_2, κ_2) be two parametrized 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) $\kappa_2(A(x)) \leq g(\kappa_1(x))$ for a computable function $g : \mathbb{N} \to \mathbb{N}$.

does not depend $|x|$

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

Not all reductions are fpt.

Many reductions from classical complexity theory are not parameter preserving.

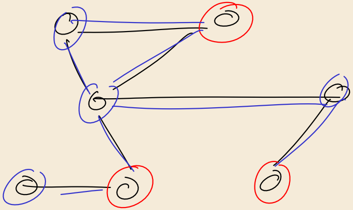
Recall:

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

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

Independent-Set $\leq_p$ Vertex-Cover

$G' = G \qquad k' = |V| - k$



$\not\Rightarrow$ p-Independent-Set $\overset{?}{\leq}_{fpt}$ p-Vertex-Cover

What's the equivalent of NP?

Parametrized NP: Non-deterministic NP

$\mathcal{P}$ corresponds to $\mathcal{FPT}$ ... but what is the analogue for $\mathcal{NP}$?

Definition 3.12 (para-NP)

The class *para-$\mathcal{NP}$* consists of all parametrized decision problems that are solved by a *non-deterministic* fpt-algorithm.

$\leq f(k) \cdot p(|x|)$

Some nice properties:

1. para-$\mathcal{NP}$ is closed under fpt-reductions.
2. $\mathcal{FPT} = \text{para-}\mathcal{NP} \iff \mathcal{P} = \mathcal{NP}$
3. an analogue for *kernalization* in $\mathcal{FPT}$ holds for para-$\mathcal{NP}$ (discussed later)

$\rightsquigarrow$ Can define para-$\mathcal{NP}$-hard and para-$\mathcal{NP}$-complete similarly as for $\mathcal{NP}$:

Definition 3.13 (para-NP-hard)

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

. . . is too strict

Theorem 3.14 (para-NP-complete → NP-complete for finite parameter)
Let (L, κ) be a nontrivial ($\emptyset \neq L \neq \Sigma^\star$) parametrized problem that is para-$\mathcal{NP}$-complete.
$\exists d$ Then $L_{\leq d} = \{x \in L : \kappa(x) \leq d\}$ is $\mathcal{NP}$-hard.

The converse is essentially also true.

Proof: para-$\mathcal{NP}$-complete (L, κ)

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

non. det. fpt-algo for L'

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

$\Rightarrow (L', \kappa_{one}) \leq_{fpt} (L, \kappa)$

i.e. $A(x) \in L \iff x \in L'$

A fpt-algo rt. $\leq f(k) \cdot n^c = $ poly-time

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

$\kappa(A(x)) \leq g(\kappa_{one}(x)) = g(1) =: d$

$\Rightarrow$ found a poly-time reduction (A)

from L' to $L_{\leq d} = \{ x \in L : \varkappa(x) \leq d \}$

$\Rightarrow$ $L_{\leq d}$ NP-hard. $\square$

This means that only very few problems are para-NP-complete (for example p-Graph-Coloring)

But p-Clique, p-Independent-Set, p-Dominating-Set cannot be para-NP-complete, since fixing $k \leq d$ makes the problems poly-time solvable (brute-force method), and we assume they are $\notin$ FPT.

$\Rightarrow$ para-NP-theory does not help settle the complexity of these problems.