Efficient Algorithms and Problem Complexity
– Dealing with NP-Completeness: Fixed-Parameter Tractability –

Frank Drewes
Department of Computing Science
Umeå University
Today's Menu

1. The General Idea of Fixed-Parameter Tractability

2. Formalization of Fixed-Parameter Tractability

3. A Non-Trivial Example
Parameters

Instances of NP-complete problems often contain a parameter $k \in \mathbb{N}$.

- **CLIQUE** = Does the graph $G$ contain a clique of size $k$?
- **TSP_D** = Does the distance matrix $M$ allow for a tour of length $\leq k$?
- **BIN PACKING** = Can items of size $s_1, \ldots, s_n$ be stored in $k$ bins?
- **LCS** = Do strings $u_1, \ldots, u_n$ contain a common subsequence of length $k$?

Even if they are not explicit, we may define parameters:

- For **HAM**, we may consider the maximum degree of nodes.
- For **SAT**, we may count the number of distinct variables.
- For **INTEGER PROGRAMMING**, we may use the largest coefficient of the inequalities.
Why are We Interested in Parameters?

The Question

We may have an application in which the instances are large, but the parameters are small. In which cases is this an advantage?

Our hope: An algorithm for a problem $A \in \text{NP}$, instead of simply running in time $2^{p(n)}$ for some polynomial $p$, may run in time

$$f(k) \cdot p(n)$$

for a (computable) function $f$.

Then $A$ is essentially solvable in polynomial time, except that the constant factor increases with the parameter.
Formalization of Fixed-Parameter Tractability

Definition (parameterized problem)

A parameterized problem is a pair $(A, \kappa)$ consisting of

- a decision problem $A$ and
- a polynomial-time computable function $\kappa$ that maps every instance $x$ of $A$ to a number $\kappa(x) \in \mathbb{N}$, the parameter.

Example: If $A = \text{CLIQUE}$, let $\kappa(G, k) = k$. 
Formalization of Fixed-Parameter Tractability

**Definition (fixed-parameter tractable)**

A parameterized problem \((A, \kappa)\) is **fixed-parameter tractable** if there are a computable function \(f\) and a RAM \(M\) such that \(M\) decides \(A\) in time \(f(\kappa(x)) \cdot p(|x|)\).

\(\text{FPT}\) denotes the class of all fixed-parameter tractable problems.

**Notes:**

- \(P \subseteq \text{FPT}\) (if we identify a decision problem \(A\) with \((A, \kappa)\) where \(\kappa(x) = 0\)).
- Every slice \((A, \kappa)_l = \{x \in A | \kappa(x) = l\}\) of a problem in \(\text{FPT}\) is in \(P\).
- The parameter \(\kappa(x) = |x|\) puts every decidable problem into \(\text{FPT}\).
Example: Parameterized SAT

Parameterized SAT ($\kappa(\phi) = \text{number of distinct variables in } \phi$) is in FPT:
For a formula $\phi$ with $k$ variables, test all $2^k$ truth assignments
$\Rightarrow$ running time $O(2^k \cdot |\phi|)$.

Thus, we can cope with SAT if we know that the number of variables is small (even if $\phi$ itself is large).

Trying to do the same with CLIQUE:
For an input graph $G = (V, E)$ with $|V| = n$, checking all subsets of $V$ of size $k$ means to check $n^k$ possibilities. 😞

Can we do better? Perhaps not, because CLIQUE is W[1]-complete.
A Non-Trivial Example

Application Scenario

Let’s start with a possible application scenario:

**Goal:** choose a editorial board of \( k \) editors for a new scientific journal.

**Available data:** bibliography database of the most important books in the area covered by the journal.

**Assumption:** Choosing at least one author of each book guarantees good coverage of the area.

**Question:** Can we select \( k \) authors so that we “hit” the author list of each book at least once?

This problem is called **HITTING SET**. It is one of the most important NP-complete problems.
Let us denote the parameterized version, with the obvious parameter $k$, by $k$-HITTING SET. Is $k$-HITTING SET fixed-parameter tractable?

Unfortunately, $k$-HITTING SET is W[2]-complete
⇒ maybe it is time to give up?
A Non-Trivial Example

Don’t Give up so Quickly!

In our editorial board example, \( k \) is the (small) number of editors wanted.

Is there another (small) parameter?

YES! – Books to tend to have very few authors, so let’s look at . . .

**k-s-HITTING SET**

*Input:* Nonempty sets \( X_1, \ldots, X_m \) and a number \( k \in \mathbb{N} \).

*Parameter:* \( k + s \), where \( s = \max_{1 \leq i \leq m} |X_i| \).

*Question:* Is there a set \( X \) of size \( k \) such that \( X_i \cap X \neq \emptyset \) for all \( i \)?

Basic observations underlying the algorithm on the next slide:

- Since we must hit each \( X_i \) anyway, we can process them in any order.
- For every \( X_i \), we have to try at most \( s \) choices.
A Non-Trivial Example

k-s-HITTING SET is in FPT

\[
\text{hit}(X_1 \cdots X_m, k)
\]

if \( k = 0 \) or \( m = 0 \) then return \( m = 0 \)

for \( x \in X_1 \) do

\( X' \leftarrow \) empty list

for \( j = 2, \ldots, m \) do

if \( x \notin X_j \) then append \( X_j \) to \( X' \)

if \( \text{hit}(X', k - 1) \) then return true

return false

Recurrence relation \( T(n, k, s) \) for running time:

\[
T(n, 0, s) \leq c
\]

\[
T(n, k, s) \leq s \cdot T(n, k - 1, s) + cn \quad (\text{for } k > 0 \text{ and a suitable constant } c).
\]
**k-s-HITTING SET is in FPT**

\[
T(n, 0, s) \leq c \\
T(n, k, s) \leq s \cdot T(n, k - 1, s) + cn
\]

Proving by induction on \(k\) that

\[
T(n, k, s) \leq (2s^k - 1) \cdot cn
\]

for all \(s \geq 2\):

- **Induction basis:** obvious, because \(T(n, 0, s) \leq c\).
- **Inductive step:**
  \[
  T(n, k, s) \leq s \cdot T(n, k - 1, s) + cn \\
  \leq s \cdot (2s^{k-1} - 1) \cdot cn + cn \\
  = (2s^k - s) \cdot cn + cn \\
  = (2s^k - s + 1) \cdot cn \\
  = (2s^k - 1) \cdot cn \quad \text{(because } s \geq 2)\
  \]
A Non-Trivial Example

The Method of Bounded Search Trees

\[
\text{hit}(\cdots)
\]

\[
\text{...}
\]

\[
\text{for } x \in X_1 \text{ do branching at most } s \text{ times}
\]

\[
\text{...}
\]

\[
\text{if } \text{hit}(X', k - 1) \text{ then } \cdots \text{ descending at most } k \text{ times}
\]

\[
\text{...}
\]

- The tree of instances that the recursion searches is of size \(O(s^k)\).
- The computation at each individual node takes polynomial time (linear in this case).
- In total, this gives the bound from the preceding slide.

Note that the size of the search tree is not important for fixed-parameter tractability – only that it does not depend on \(n\).
Please read Section 11.4 in the textbook.