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 the parameters are not obviously part of the problem definition, we may consider others:

- 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 Motivating Question

We may have an application in which the instances are large, but a certain parameter \( k \) is small. May this be 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 depends on the parameter.
Formalization of Fixed-Parameter Tractability

**Formalization**

**Definition (parameterized problem)**

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

- a decision problem \(A\) and
- a polynomial-time computatable 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 \(\kappa(G, k) = k\).

**Side remark:** The field is quite young – it started at the very end of the previous century, by the work of Downey and Fellows.
Formalization

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 choice of the parameter is essential. If \(\kappa(x) = \|x\|\) then \((A, \kappa) \in \text{FPT}\) for every decidable problem \(A\). [WHY? How to choose \(f\)?]
Example: Parameterized SAT

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

Thus, we can cope with \(\text{SAT}\) if we know that the number of variables is small (even if \(\varphi\) itself is large).

Trying the same with \(\text{CLIQUE}\), with the clique size \(k\) as the parameter: 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 \(\text{CLIQUE}\) is \(W[1]\)-complete.
Let’s start with a possible application scenario:

**Goal:** choose an 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 is **HITTING SET**, another important NP-complete problem.
A Non-Trivial Example

HITTING SET

Input: Sets $X_1, \ldots, X_m$ and a number $k \in \mathbb{N}$.
Question: Is there a set $X$ of size $k$ such that $X_i \cap X \neq \emptyset$ for all $i$?

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?
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:

- We must hit each \( X_i \) anyway, so 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)
\quad \text{if } k = 0 \text{ or } m = 0 \text{ then return } m = 0
\quad \text{for } x \in X_1 \text{ do}
\quad \quad X' \leftarrow \text{empty list}
\quad \quad \text{for } j = 2, \ldots, m \text{ do}
\quad \quad \quad \text{if } x \notin X_j \text{ then append } X_j \text{ to } X'
\quad \quad \text{if } \text{hit}(X', k - 1) \text{ then return true}
\quad \text{return false}
```

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

- $T(n, 0, s) \leq c$
- $T(n, k, s) \leq s \cdot T(n, k - 1, s) + csn$ \quad (for $k > 0$ and a suitable constant $c$).
**k-s-HITTING SET is in FPT**

\[
\begin{align*}
T(n,0,s) & \leq c \\
T(n,k,s) & \leq s \cdot T(n,k-1,s) + csn
\end{align*}
\]

Proving by induction on \(k\) that

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

for all \(n \geq 1\) and \(s \geq 2\):

- **Induction basis**: by the choice of \(c\).
- **Inductive step**: 
  \[
  \begin{align*}
  T(n,k,s) & \leq s \cdot T(n,k-1,s) + csn \\
  & \leq s \cdot (2s^{k-1} - 1) \cdot csn + csn \\
  & = (2s^k - s) \cdot csn + csn \\
  & = (2s^k - s + 1) \cdot csn \\
  & = (2s^k - 1) \cdot csn \quad \text{(because } s \geq 2)\n  \end{align*}
  \]
The Method of Bounded Search Trees

\[ \text{hit}(\cdots) \]
\[ \cdots \]
\[ \text{for } x \in X_1 \text{ do } \begin{array}{l}
\text{branching factor at most } s \\
\text{\cdots} \\
\text{if } \text{hit}(X', k-1) \text{ then } \cdots \text{ descending at most } k \text{ times} \\
\text{\cdots} 
\end{array} \]

- The search tree is of size \( O(s^k) \).
- The computation at each individual node takes polynomial time.
- 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 \( n \).
You can find a bit more about FPT in Section 11.4 of *Algorithms* (the official course textbook) and much more in books by Downey and Fellows, Flum and Grohe, and Niedermeier.