Advanced Algorithms (2IL40)

Bas Kloet & Christian Luijten

May 27, 2006

Survey paper on

Fixed-Parameter Tractability

Abstract

Fixed-Parameter Tractability (FPT) is a method to find solutions in polynomial time for problems which normally only have exponential time solutions. This document is intended to be read as a basic introduction to the field of FPT. It describes the main approach, types of problems which can or cannot be solved through the use of FPT and gives a number of examples.

Bas Kloet & Christian Luijten (Group 10)
Advanced Algorithms (2IL40)
Department of Mathematics & Computer Science
Technische Universiteit Eindhoven

1 Introduction

There is a large group of natural computational problems that have a time complexity that is exponential or worse. For many of these problems however there are techniques that can be used to dramatically lessen the time needed to find an (acceptable) solution, either by approximating the solution or restricting the problem space. This survey describes a technique that is based on the second approach, namely Fixed-Parameter Tractability.

> **Definition.** A *fixed-parameter tractable* problem is a parameterized problem where for each fixed parameter value y the problem is solvable in time $O(n^c)$ where c is a constant independent of the parameter y. [DF92]

The main body of this survey consists of three parts. Section 2 gives an explanation of the workings of FPT, its history and main research areas. Section 3 gives a number of real-world applications of FPT and section 4 gives a conclusion on the current state of FPT and some research frontiers.

The goal of this survey is to provide the reader with an introduction to FPT, give an idea of its strengths and weaknesses and provide links to more in-depth information. It is assumed that the reader has basic knowledge of complexity theory and algorithms.

2 Fixed-parameter tractability

The main reason for the research of Downey & Fellows on FPT was to find a way to solve NP-complete problems in certain cases. The idea is that since NP-complete problems are often only truly intractable for a small range of parameter values, the fixing of those parameter values could lead to a tractable problem.

The definition Downey and Fellows use for parameterized problems:

> **Definition.** A *parameterized problem* is a set $L \subseteq \Sigma^* \times \Sigma^*$ where Σ is a fixed alphabet.
>
> For a parameterized problem L and $y \in \Sigma^*$ we write L_y to denote the associated fixed-parameter problem (y is the parameter) $L_y = \{x | (x, y) \in L\}$. [DF92]

In other words, L is a problem with at least one free parameter, L_y is the problem with that parameter fixed. A problem L is called *fixed-parameter tractable* if L is in NP, but L_y is in P.

Some problems in the class of NP-complete problems are FPT, while others are not. This leads to the conjecture that there exists a hierarchy within NP, which is described in more detail in Section 2.3.

2.1 History

There is very little to be found on the history of FPT. but in [Slo01] Christian Sloper pieced together its probable development, of which the following is an extract.

The history of FPT most likely stems from the Robertson-Seymour theorems on *graph minors*.

> **Definition.** A graph H is a minor of graph G, $H \leq_m G$, if a graph isomorphic to H can be obtained by repeatedly contracting edges of a subgraph of G. Contracting edge (u, v) means identifying u and v as a single vertex maintaining all neighbours. [Slo01]

Robertson and Seymour proved Wagner's Conjecture, which states that for every infinite set of graphs $G1, G2, \ldots$ there exists $i < j$ such that $G_i \leq_m G_j$. The proof will not be described here, but in it they used an algorithm for finding tree-decompositions of width $\leq k$ for fixed values of k. The original time complexity of this algorithm was $O(n^{k+2})$, but by using a divide-and-conquer approach they created an algorithm of time complexity $O(n^2)$.

The step from $O(n^{O(k)})$ to $O(n^{O(1)})$ drew the interest of Fellows and Langston, who observed that a similar algorithm could be created for Vertex Cover. Following this, Fellows and Downey started exploring the possibilities of Parameterized Complexity and that is where the field of FPT truly began.

2.2 Fixed-Parameter Tractability

As with NP-complete problems, the distinction between FPT problems and non-FPT problems is made using reductions; if a parameterized problem P is reducible to another parameterized problem P', and P is fixed-parameter tractable, then P' is also fixed-parameter tractable. [DF92]

Various algorithms have been shown fixed-parameter tractable, such as determining of a fixed graph that another graph is a minor isomorphic of this graph, determining the existence of an isomorphic subtree for a tree of fixed width. [DF92]

Other algorithms don't have the fixed-parameter tractability property and can only be solved in exponential time by exhaustively trying all possible solutions. An example is the Dominating Set Problem: determining whether there exists a dominating set of size k or less in a graph G. A dominating set of size k of a graph $G = (V, E)$ is a subset D of V of k vertices such that each vertex *not* in D is joined to at least one member of D by an edge in E. The k-Dominating Set Problem is only solvable in $O(n^{k+1})$ time. [DF92]

2.3 W-Hierarchy

As said in the previous paragraph, there is a group of problems that are not Fixed Parameter tractable. There is no concrete proof for this (such a proof would imply that $P \neq NP$, which still isn't officially proven), but the existence of this group is made extremely plausible by using a completeness program.

The completeness program works just like the proof for the existence of the group of NP problems. There is a base problem for which the existence of a FPT solution is extremely unlikely and other problems can be reduced to this problem. The difference between NP and Fixed Parameter Intractability is that there is a clear hierarchy between different intractable parameterized problems, called the W-Hierarchy. This hierarchy was first introduced by Fellows and Downey in [FD98].

To explain the W-Hierarchy we will need a number of definitions.[DF94] We decided to copy these integrally for easy reading:

> **Definition.** A Boolean circuit is of *mixed type* if it consists of circuits having *small gates* and *large gates*.
>
> **Definition.** *Small gates*: *not*, *and* and *or* gates with bounded fan-in.
>
> **Definition.** *Large gates*: *not*, *and* and *or* gates with unrestricted fan-in.
>
> **Definition.** The *depth* of a circuit C is defined as the maximum number of gates (small or large), not counting *not* gates, on an input-output path in C.
>
> **Definition.** The *weft* of a circuit C is defined as the maximum number of large gates, on an input-output path in C.
>
> **Definition.** A family of circuits F has *bounded depth* if there is a constant h such that every circuit in the family F has depth at most h.
>
> **Definition.** We say that F has *bounded weft* if there is a constant t such that every circuit in the family F has weft at most t.
>
> **Definition.** Let F be a family of decision circuits. We allow that F may have many different circuits with a given number of inputs. To F we associate the parameterized circuit problem $L_F = \{(C, K) : C \in F \text{ and } C \text{ accepts an input vector of weight } k\}$

Definition. A parameterized problem L belongs to $W[t]$ if L uniformly reduces to the parameterized circuit problem $L_{F(t,h)}$ for the family $F(t,h)$ of mixed type decision circuits of weft at most t and depth at most h, for some constant h.

Of all these definitions the one of most interest to us is the last one, since it is this definition that allows us to describe a hierarchy on the complexity of parameterized problems.

We will first describe a problem of type $W[1]$, specifically the problem of SHORT TURING MACHINE COMPUTATION. This problem is especially useful since it provides us with a *very* strong indication that problems in $W[1]$ are fixed parameter intractable.

Definition. SHORT TURING MACHINE COMPUTATION: Given a Nondeterministic Turing Machine M, a string x and a parameter k, does M have a length k computation path accepting x.

The proof that SHORT TURING MACHINE COMPUTATION can be uniformly reduced to a mixed type decision circuit of weft at most 1 and therefor is in class $W[1]$ is described in [FD98]. We will not repeat that proof here, but will discuss a very important implication. Since the SHORT TURING MACHINE COMPUTATION problem has little to no structure that can be used to find an efficient solution, it is widely regarded to be fixed parameter intractable. Since the complexity of problems that are in $W[t]$, with $t > 1$, is at least that of SHORT TURING MACHINE COMPUTATION, this makes a strong case for the conjecture that *all* problems in $W[t]$ are fixed parameter intractable.

This leads us to the final definitions of the W-Hierarchy:

Definition. $W[P]$: the class of circuits obtained by having no restriction on depth, i.e. P-size circuits.

Definition. $W[SAT]$: boolean circuits of P-size.

Definition. The *W-Hierarchy* is the union of the $W[t]$ classes together with the classes $W[SAT] \subseteq W[P]$.

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq W[SAT] \subseteq W[P]$$

3 Applications

3.1 Introduction

Now that the basic workings of FPT have been explained, it is time to give a number of real-world applications. This section will focus on four areas where FPT is or can be used, specifically graph theory, satisfiability, cryptography and gene research.

It will contain algorithms nor proofs and the topics are only lightly touched upon. For detailed information you are advised to read the referenced papers.

3.2 Graph theory

Graph theory studies the properties of graphs. Graphs are used in many fields: in cartography, graphs are the generalization of a country map. In telecommunications, they are the generalization of the used network. Logistics use graphs to abstract real-world distances between destinations.

Many computational problems are solved using graph theory. However, certain types of problems cannot be solved and are part of the NP-hard problems. The Traveling Salesman Problem (TSP) is discussed here. Other famous NP-hard problems are Minimum Weight Triangulation Problem (MWT), Hamiltonian Path and Vertex Cover. MWT is not discussed in detail here, but there is a fixed-parameter algorithm[HO04] for it which runs in $O(6^k n^5 \log n)$ time, where n is the total number of points and k is the number of inner points.

3.2.1 Traveling Salesman Problem is Fixed-Parameter Tractable

One of the most famous NP-hard problems in graph theory is TSP. An exhaustive search finding the shortest path through a set of points costs $O(n!)$ time. A dynamic programming solution will get the running time within the exponential range: $O(n^2 2^n)$.

Four researchers, Deĭneko, Hoffmann, Okamoto and Woeginger discovered that TSP is FPT if most of the points are arranged in convex position. In their article 'The Traveling Salesman Problem (TSP) with Few Inner Points' they claim $O(k!kn)$ and $O(2^k k^2 n)$ running time for the two algorithms they give (where n is the number of points and k is the number of inner points). [DHOW04]

First some definitions Deĭneko et al. use: Inner points of a point set P are those points that lie inside the convex hull of P. A tour on P is a linear order of the points on P. [DHOW04]

First algorithm The first algorithm, with running time $O(k!kn)$ has space complexity $O(k)$. Essentially, it will first find a linear order on the inner points (π) and a cyclic order on the outer points (γ). Then it will try to find a shortest tour on P respecting the orders π and γ using dynamic programming.

The dynamic programming phase will construct an array $F_1[i, j, m]$, where i is the points p_i in the outer points set, j is the point q_j in the inner points and the index m is the position (Inn or Out). The value of $F_1[i, j, m]$ represents the length of a shortest path on $\{p_1, \ldots, p_i\} \cup \{q_1, \ldots, q_j\}$. The exact conditions which are satisfied can be found in [DHOW04].

The length of a shortest tour can now be computed as:

$$\min\{F_1[n-k, k, Out] + d[p_{n-k}, p_1], F_1[n-k, k, Inn] + d(q_k, p_1)\}$$

A recursive algorithm can be used to calculate F_1 for every i, j and m.

Second algorithm The second algorithm has a better running time ($O(2^k k^2 n)$) at the expense of a larger space complexity ($O(2^k kn)$). Like with the first algorithm, it will first find a linear order on the inner points (π), but will then immediately start the dynamic programming.

The array which stores information now gets $F_2[i, S, r]$, i is the same as with the first algorithm, S is a subset of the inner points of P and r is a point out of the inner points.

$F_2[i, S, r]$ is to be interpreted as the length of the shortest path on $\{p_1, \ldots, p_i\} \cup S$ that satisfies the conditions in [DHOW04].

The length of a shortest tour is then:

$$\min\{F_2[n-k, Inn(P), r] + d(r, p_1) | r \in Inn(P) \cup \{p_{n-k}\}\}$$

Conclusion The intuition that “Fewer inner points make the problem easier to solve”[DHOW04] is correct.

3.3 Proving (un)satisfiability

Proving (un)satisfiability of propositional formulas has always been a point of much interest to algorithm researchers. It is therefor no surprise that there are a number of different approaches to this problem that use FPT.

One group of fixed parameter solutions to the satisfiability problem is based on structural decompositions, which means that they create a graph from the propositional formula. Possible parameters that can be fixed are the tree-width, branch-width or clique-width of the resulting graph. These approaches are all useful for a subset of propositional formulas and are relatively well researched.

One relatively new approach is based on the *maximum deficiency* of a *minimal unsatisfiable* CNF formula F, as described in [Sze03].

> **Definition.** A formula is *minimal unsatisfiable* if it is unsatisfiable, but omitting any of its clauses makes it satisfiable.

> **Definition.** The deficiency of F is: $\delta(F) := m - n$, where m is the number of clauses and n is the number of variables.

The class of minimal unsatisfiable formulas with deficiency k is denoted by $\mathrm{MU}(k)$.

Original algorithms to decide whether a formula is in $\mathrm{MU}(k)$ had a time complexity of $n^{O(k)}$, which means that even for small k the algorithms are impractical for large inputs.

In [Sze03] Stefan Szeider shows that $\mathrm{MU}(k)$ is fixed-parameter tractable and gives an algorithm with time complexity $O(2^k n^4)$, bringing the problem into the realm of FPT.

A very useful byproduct of the algorithm is a general algorithm which runs in time $O(2^k n^3)$ on instances F, if the *maximum deficiency* over all subsets $F' \subseteq F$ is at most k. In other words, if for each subset of the CNF formula F the number of clauses m is at most k bigger than the number of literals n, then a satisfying assignment can be found in polynomial time.

The main reason that this specific algorithm is so interesting is that it is incomparable with the approaches based on structural decompositions. There are problems with bounded maximum deficiency but arbitrarily large tree, branch or clique-width and vice versa.

Another point of interest is that maximum deficiency can be calculated in polynomial time by matching algorithms, whereas computation of tree/branch-width is NP-hard and it is unknown whether the recognition of graphs with fixed clique-width can be done in polynomial time.

3.4 Cryptography

A particular field of interest in complexity theory and especially FPT is the field of cryptography. The reason for this is that, opposed to most other fields, cryptography relies on the fact that some problems are inherently hard to solve. While this is true in theory, real world implementations are forced to restrict certain parameters of the problem, which provides a foothold for solutions based on Fixed-Parameter Tractability (FPT).

This section is based on an article by Fellows and Koblitz [FK93]. For one, the article describes an algorithm that uses a combination of FPT and randomization to determine whether an n-digit number has a prime divisor less than or equal to n^k in expected time $f(k)n^3$. This can be used as an aid in finding the the prime numbers used in for example RSA encryption.

Fortunaltely for the field of cryptography, there also is a large number of problems relevant to cryptography for which even the parameterized versions have a exponential running time. Examples are k-Subset Sum, k-Perfect Code and k-Subset Product which are all hard for $W[1]$.

Finally we will describe two open questions that could have direct real-world consequenses if they should prove to be fixed parameter tractable. Both are described in greater depth in [FK93].

The first is the problem of the BOUNDED HAMMING WEIGHT DISCRETE LOGARITHM. Since there have been some suggestions to use exponents of faily small Hamming weight to speed up crypto systems, finding that this problem is fixed parameter tractable could be a big security breach for these systems. Some further investigation has shown that as of 2001 this problem was still open.

The second problem is the BOUNDED HAMMING WEIGHT ELLIPTIC CURVE DISCRETE LOGARITHM. Proposals for using elliptic curve cryptosystems in smart cards have included the idea of placing an upper bound on the Hamming weight of certain elements of the algorithm. As with the previous problem, finding a FPT algorithm could mean a possible security breach in those cards. As of 2006 we haven't been able to find any proof or refutation of the fixed parameter tractability of this problem.

3.5 Gene research

In genetics, many computational problems arise, mostly combinatorial. One of these is 'Perfect Path Phylogeny Haplotyping with Missing Data', which is FPT.[GNT04]

Since the reader is not expected to have a background in biology, some terms probably need explanation.

A *haplotype* is a variation in the DNA of chloroplasts or mitochondrions (these are certain elements in a cell). It is used in phylogenetics to decide on the origin of a certain individual.

Phylogeny is "the origin and evolution of a set of organisms, usually a set of species."[1]

Haplotyping via perfect phylogeny finally is "a method for haplotype inference where it is assumed that the (unknown) haplotypes underlying the (observed) genotype data can be arranged in a genetic tree in which each haplotype results from an ancestor haplotype via mutations."[GNT04]

For computer scientists, it is good to know that the problem can be stripped of its biological context: Translate haplotypes to binary strings, genotypes to

[1]from Wikipedia page Phylogenetics

trinary strings. The genotype resulting from two haplotypes has a 0-entry or 1-entry at those positions where both haplotypes agree (the value of the genotype-entry is then equal to that of the haplotypes' position) and a 2-entry if they don't.[GNT04]

What remains is a purely combinatorial problem, citing [GNT04]:

> A genotype matrix A *admits a perfect phylogeny* if there exists a rooted tree T, called *perfect phylogeny*, such that:
>
> 1. Each column of A labels exactly one edge of T.
> 2. Every edge of T is labeled by at least one column of A.
> 3. For each row r of A there are two nodes in T (possibly identical) labeled r' and r''. The labels r' and r'' are called *haplotype labels*.
> 4. For every row r of A the set of columns with value 2 in this row forms a path p in T between r' and r''. The set of columns with value 1 in this row forms a path from T's root to the top-most node on the path p.

The Perfect Phylogeny Haplotyping Problem (given a genotype matrix A, does A admit a perfect phylogeny?) is solvable efficiently, as is the Perfect Path Phylogeny Haplotyping Problem (given a genotype matrix A, does A admit a perfect phylogeny that is a path?). However, adding missing data complicates the problems into intractability.[GNT04]

Perfect Path Phylogeny Haplotyping with Missing Entries (PPPH) is NP-complete. To find FPT properties, Gramm, Niehoff and Tantau turned to the biological constraints on the problem and found three possible assumptions and gave motivations for them:

- Focus on path phylogenies (instead of the general case).
- Focus on directed phylogenies.
- Assume that for each single nucleotide polymorphism site only a small fraction of the information is missing.

The solution to PPPH is a two-stage algorithm. The first phase collapes multiple columns in the matrix with missing data to one 'consensus' column. If a perfect phylogeny can be found for this new matrix, there is also one for the original. This phase takes $O(4^k m^3 n)$ time, where $n \times m$ is the size of the genotype matrix A, and k is the number of missing entries per column.[GNT04]

The second phase of the algorithm involves dynamic programming. For each remaining column the missing data is then tried to be resolved. The dynamic programming phase takes $O(3^{O(k^3 \cdot 6^k \cdot k!)} \cdot n^2)$ time.[GNT04]

Conclusion The complete algorithm runs in $O(4^k m^3 n + 3^{O(k^3 \cdot 6^k \cdot k!)} \cdot n^2)$ time, where $n \times m$ is the size of the genotype matrix A, and k is the number of missing entries per column, which makes it tractable for very small values of k.[GNT04]

The next step of Gramm's et al. research will be to remove the focus on directed phylogenes and try to get a fixed-parameter tractable version with undirected phylogenes.[GNT04]

4 Conclusion

FPT is used in many areas of computation. NP-hard problems can sometimes be solved efficiently when one has some knowledge about the input, for instance in graph theory the depth of a tree, in networking the number of processors in a parallel processing system, or with formula satisfiability the number of variables in a logical formula.

The topics dealt with in this survey paper are only lightly touched upon and the actual algorithms and proofs aren't given. The field of FPT is huge and the examples given are but the tip of the ice berg of fixed-parameterized problems.

References

[DF92] R.G. Downey and M.R. Fellows. Fixed-parameter intractability (extended abstract). In *Proceedings of the Seventh Annual Structure in Complexity Theory Conference, 1992*, pages 36–49, jun 1992.

[DF94] R.G. Downey and M.R. Fellows. *Fixed-parameter tractability and completeness II: On completeness for W[1]*, volume 141 of *Theoretical Computer Science*, pages 109–131. Elsevier Science B.V., 1994.

[DHOW04] Vladimir G. Deneĭko, Michael Hoffmann, Yoshio Okamoto, and Gerhard J. Woeginger. The traveling salesman problem with few inner points. In *Computing and Combinatorics: Proceedings of the 10th Annual International Conference, COCOON 2004, Jeju Island, Korea, August 17-20, 2004*, volume 3106 of *Lecture Notes in Computer Science*, pages 268–277, 2004.

[FD98] Michael Fellows and Rodney Downey. Parameterized complexity after (almost) 10 years: Review and open questions, 1998.

[FK93] Michael R. Fellows and Neal Koblitz. Fixed-parameter complexity and cryptography. In *Proceedings of the 10th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 673 of *Lecture Notes in Computer Science*, pages 121–131. Springer-Verlag, 1993.

[GNT04] Jens Gramm, Till Nierhoff, and Till Tantau. Perfect path phylogeny haplotyping with missing data is fixed-parameter tractable. In *Parameterized and Exact Computation: Proceedings of the First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004*, volume 3162 of *Lecture Notes in Computer Science*, pages 174–186. Springer-Verlag, 2004.

[HO04] Michael Hoffmann and Yoshio Okamoto. The minimum weight triangulation problem with few inner points. In *Parameterized and Exact Computation: Proceedings of the First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004*, volume 3162 of *Lecture Notes in Computer Science*, pages 200–212. Springer-Verlag, 2004.

[Slo01] Christian Sloper. Parameterized complexity and the method of test sets. Master's thesis, University of Bergen, 2001.

[Sze03] Stefan Szeider. *Minimal Unsatisfiable Formulas with Bounded Clause-Variable Difference are Fixed-Parameter Tractable*, volume 2697 of *Lecture Notes in Computer Science*, pages 548–558. Springer-Verlag, 2003.