Parameterized Complexity After (Almost) 10 Years: Review and Open Questions

Rodney G. Downey [1]
School of Mathematics and Computing Sciences
P.O. Box 600
Victoria University
Wellington, New Zealand

Michael R. Fellows [2]
Department of Computer Science
University of Victoria
Victoria, B.C. V8W 3P6, Canada

October 9, 1998

[1]Research supported by a grant from the United States/New Zealand Cooperative Science Foundation and by the New Zealand Marsden Fund for Basic Science.
Email address: `rod.downey@vuw.ac.nz`

[2]Research supported by grants from the National Science and Engineering Research Council of Canada.
Email address: `mfellows@csr.uvic.ca`

Abstract

We give a review of the development and some of the achievements of the theory of parameterized complexity in the last (nearly) 10 years. We highlight what we see as some of the major open questions and programmes for future development.

Contents

Chapter 1

Introduction

We begin our monograph [DF98] with the sentence

"The idea for this book was conceived over the second bottle of Villa Maria's Cabernet Merlot '89, at the dinner of the Australasian Combinatorics Conference held at Palmerston North, New Zealand in May 1990, where the authors first met and discovered they had a number of interests in common."

It seems therefore most appropriate in this New Zealand conference to look back at the achievements of the area as well as looking at future directions for research. Even at this point we'd like to remark that the opportunities for future research are very exciting indeed!

So what is parameterized complexity? Classical complexity theory views questions as either decision problems "Does a graph have a red bandersnatch?" or, to a lesser extent, optimization questions, "What is the biggest red bandersnatch in the graph?" One of the "triumphs" of the last 30 years of complexity theory is the realization that most interesting combinatorial problems are essentially intractable, being NP-complete or worse.

One of the points stressed in Garey and Johnson [GJ79], is that a hardness result such as NP-completeness should be just the beginning of an attack on a problem. All it says is that the initial hope for an efficient exact general algorithm is in vain. The principal idea of parameterized complexity is to try to look more deeply at the *structure* of the input. In parameterized complexity we consider the input not just as a single entity, for example, a graph — but as a graph with an additional parameter. So the underlying languages are subsets of $\Sigma^* \times \Sigma^*$. The typical problem will be of the form below.

Input: $\langle x, k\rangle$
Parameter: k.
Question: Is $\langle x, k\rangle \in L$?

The point is that we are trying to look at the problem *by the slice.* That is, we ask what is the behaviour for a *fixed* k? There may be many different ways to slice a particular problem. The following three examples were in fact our original motivation and serve well to demonstrate that there is something of interest here.

VERTEX COVER
Instance: A graph $G = (V, E)$.
Parameter: A positive integer k.
Question: Does G have a vertex cover of size $\leq k$? (A vertex cover of a graph G is a collection of vertices V' of G such that for all edges v_1v_2 of G either $v_1 \in V'$ or $v_2 \in V'$.)

DOMINATING SET
Input: A graph G.
Parameter: A positive integer k.
Question: Does G have a dominating set of size k? (A *dominating set* is a set $V' \subseteq V(G)$ where, for each $u \in V(G) - V'$ there is a $v \in V'$ such that $uv \in E(G)$.)

INDEPENDENT SET
Input: A graph G.
Parameter: A positive integer k.
Question: Does G have a set of k vertices $\{x_1, \ldots, x_k\}$ such that all $i \neq j$, x_i is not adjacent to x_j?

All of these can be solved trivially by brute force, trying all possible k-subsets, and hence in time $\Omega(|G|^{k+1})$. For INDEPENDENT SET and DOMINATING SET, this is essentially the best known algorithm, while VERTEX COVER can be solved in time $O(|G|)$ for any fixed k. This fact

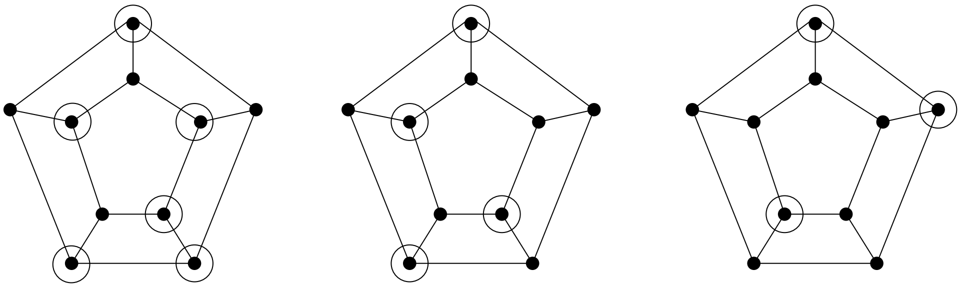

Figure 1.1: Three Vertex Set Problems

has obvious implications for applications where the range of the parameter is small. Table 1.1 illustrates the difference between a running time of $\Omega(|G|^{k+1})$ (that is, where complete search is necessary), and one running in time $2^k|G|$. The latter has been achieved for several natural parameterized problems. (In fact, as we see below the constant 2^k can sometimes be significantly improved.)

	$n = 50$	$n = 100$	$n = 150$
$k = 2$	625	2,500	5,625
$k = 3$	15,625	125,000	421,875
$k = 5$	390,625	6,250,000	31,640,625
$k = 10$	1.9×10^{12}	9.8×10^{14}	3.7×10^{16}
$k = 20$	1.8×10^{26}	9.5×10^{31}	2.1×10^{35}

Table 1.1: The Ratio $\frac{n^{k+1}}{2^k n}$ for Various Values of n and k.

So what is the global issue? The point is that for *any* combinatorial problem we will deal with only a small finite fraction of the universe. We tend to forget that the original suggestions that asymptotic analysis was a reasonable way to measure complexity, and polynomial time was a reasonable measure of feasibility, were initially quite controversial. The reasons that the now central ideas of asymptotic polynomial time and NP-completeness have survived the test of time are basically two. Firstly, the methodologies associated with P-time and P-time reductions have proved to be readily amenable to mathematical analysis in most situations. In a word, P has good closure properties. It offers a readily negotiable mathematical currency. Secondly, and even more importantly, for "natural" problems the universe seems *kind* in the sense that if a "natural" computational problem is in P then usually we can find an algorithm having a polynomial running time with small degree and small constants. This last point, while seeming a religious, rather than mathematical, statement, seems the key driving force behind the universal use of P as a classification tool for "natural" problems.

The same kind of concrete considerations are behind the idea of *fixed parameter tractability.* Here a parameterized problem is called *fixed parameter tractable* (FPT) if, like VERTEX COVER above, there is an algorithm Φ solving the problem "efficiently by the slice". That is there is a constant c (independent of k and a function f such that Φ accepts $\langle x, k\rangle$ in time $f(k)|x|^c$. For VERTEX COVER, we can take $c = 1$.

If the constant $f(k)$ can be made small enough, then for a reasonable range of k, the algorithm is feasible for a very large instance size.

This consideration is extremely useful if we can get our hands on a good parameter with which to partition off the usual combinatorial explosion. Are such parameters usually avaiable? Are there any standard techniques to achieve parametric tractability? What should I do if I can't seem to attack the question in this way? Is the phenomenom widespread? Are there open questions? In general, why should I be interested in this area?

We attempt to answer these questions below.

Chapter 2

Review

2.1 Why This Is Interesting: Applications Often Involve Natural Parameters

There are lots of areas of applications, and potential applications.

- Most computational problems involve several pieces of input, one or more of which may be a relevant parameter for various applications.
- For instance, VERTEX COVER can be often used to represent a *conflict graph* of some some sort, and the parameter k in this situation models the cost of eliminating the conflicts. The FPT result for VERTEX COVER has many different applications through this natural generality.
- *Graph linear layout width metrics* are of interest in VLSI layout and routing problems and have important applications for width values of $k \leq 10$. Interval graphs of pathwidth $k \leq 10$ have applications in DNA sequence reconstruction problems and in other evolutionary problems such as language evolution and phylogenic tree reconstruction.
- *Logic and database problems* frequently are defined as having input consisting a formula (which may be small and relatively invariant), and some other structure (such as a database) which is typically quite large

and changeable. Formula size, or other aspects of formula structure may be a relevant parameter.

- *Robotics.* The number of degrees of freedom in a robot motion-planning problem is commonly in the range $k \leq 10$.
- *VLSI.* The number of wiring layers in VLSI chip manufacture is typically bounded by $k \leq 30$.
- *Artificial Intelligence.* One of the central problems in Artificial Intelligence, STRIPS PLANNING, is known in general to PSPACE complete (Bylander [Byl94]) yet has parametric versions that are FPT (Downey-Fellows-Stege [DFS98]).
- *Network problems* may be naturally concerned with optimally locating a small number of facilities.

  These few examples are only suggestive and by no means exhaustive. There are myriad ways in which numbers that are small or moderately large (e.g., $k \leq 40$) arise naturally in problem specifications.

2.2 Why This Is Interesting: Parameters May Be Hidden and Implicit

The above gave a sample of possible applications. There are many parameters that arise naturally in practice. Some of them are hidden, and some have been the arenas of intensive investigation in recent years. In this section we will look at this issue.

In classical complexity a decision problem is specified by two items of information:
(1) The input to the problem.
(2) The question to be answered.

In parameterized complexity there are three parts of a problem specification:
(1) The input to the problem.
(2) The aspects of the input that constitute the parameter.
(3) The question

- We have already seen *Graph Width Metrics.* These can often be *hidden* parameters in some computational problem. Examples include TREEWIDTH, CUTWIDTH, BANDWIDTH and PATHWIDTH. For example, it has been proposed that the syntactic structure of sentences of natural languages can be modeled by dependency graphs of pathwidth less than 10 (Kornai and Tuza [KT92]). CUTWIDTH and PATHWIDTH have applications for small parameter ranges in VLSI design [FL92]. TREEWIDTH and PATHWIDTH are important hidden parameters arising from the work of Robertson and Seymour which provide a generic methodology of solving generally hard problems on a wide class of graphs. (More on this later.)

- We have already seen *Implicit Parameters in Logic Programming, Compiler Design and Database Applications.* In databases, the database is often much larger than the queries. This is an obvious arena for this kind of analysis. Sometimes the parameter can be less obvious. For example, the problem of type inference arises in implementations of programming languages such as ML that are based on the polymorphic typed λ-calculus. In Henglein and Mairson [HM91] it is shown that the problem is complete for deterministic exponential time, yet it has been widely noted that in practice the problem is efficiently solved by an algorithm that is known to be exponential in the worst case. An explanation for this discrepancy comes from noting that the logic formulas that occur in natural programs tend to have small bounded depth of *let*'s. For a parameter k bounding this depth, the problem is linear fixed-parameter tractable. This observation can be articulated in the point below.

- *Possible explanation of unpredicted tractability.* Problems can be much easier in practice than one would expect from theoretical analyses. One example is in the point above. Another example concerns the register allocation problem for imperative programming languages. This is a problem that is *NP*-complete in general. It has been shown that for structured programs (with short-circuit evaluation) the control-flow graphs have treewidth less than 10, which allows linear FPT algorithms based on this hidden structural parameter to be developed in, for instance, Alstrup, Lauridsen and Thorup [ALT96]. Such considerations give a different possible explanation from those currently offered such as probabilistic behaviour.

- *Parameters Introduced by Engineering Practices.* Some proposals for implementations of public key cryptosystems have considered limiting the size or Hamming weight of keys in order to obtain faster processing times. A cautionary note is sounded by the result Fellows and Koblitz [FK93] that for every fixed k, with high probability it can be determined in time $f(k)n^3$ whether an n-bit positive integer has a prime divisor less than n^k.

- *Approximation Parameters in Computational Biology.* A parameter can be composed of any relevant aspects of a problem. In Jiang, Lawler, and Wang [JLW94] it is shown that the problem of computing a tree alignment for DNA sequences of length n for k species of cost within a factor of $1+\epsilon$ of the cost of an optimal alignment can be accomplished in time $O((kn)^{c^{1/\epsilon}})$. It is natural to consider this problem with the parameter consisting of the pair $(k, 1/\epsilon)$ and to ask whether it is fixed-parameter tractable.

- *Problems considered* in practice *come from "people" situations and hence can often have very small parameters associated with them because of our "brain" limitations.*

2.3 Why This Is Interesting: A Rich Positive Toolkit

So far we have seen that (i) parameterized complexity has lots of potential areas of application and (ii) there are many natural, often hidden, parameters. Another very important cog in the machinery is that *there is a very distinctive positive toolkit of FPT algorithm design methods, begging for further development.* We discuss some of these basic methods below. Due to space limitations we will keep the discussion fairly incomplete and refer the reader to the monograph [DF98] for further details.

- *Well-Quasi-Ordering.* The spectacular results of Robertson and Seymour give a method that in its most general form yields $O(n^3)$ fixed-parameter tractability results that are supremely impractical, primarily because they involve astronomical hidden constants. This is still useful, however, as a complexity classification tool, pointing the way to better FPT results based on a more detailed problem analysis.

Briefly, a *quasi-order* on a set A is a relation $\preceq$ that is transitive and reflexive. $\preceq$ is a *well-quasi-order* (WQO) if every filter has a finite basis. That is, for any set $F \subseteq A$ such that $x \in F$ and $x \preceq y$ implies $y \in F$, there is a finite subset $F' \subseteq F$ such that $x \in F$ iff $\exists f \in F'(f \preceq x)$.

The crucial point is that if, for a fixed f, the relation $f \preceq x$ can be decided in polynomial time, then membership in a filter F can be determined in polynomial time by the finiteness of F'.

The best known example is Robertson and Seymour's proof that the minor relation on the class of finite graphs is a WQO. Here G is a minor of H iff G can be gotten from H be a finite sequence of edge contractions and vertex/edge deletions. Robertson and Seymour in the famous series of papers (e.g [RS85, RS96]) showed the WQO result and they also proved that for a fixed G the relation "G is a minor of H" can be decided in time $O(|H|^3)$. Thus, *any set of graphs closed under the minor relation has a cubic time recognition algorithm.* This allows is to immediately conclude that the following problems are FPT.

GRAPH GENUS
Instance: A graph $G = (V, E)$.
Parameter: A positive integer k.
Question: Does G have genus k? (That is, can G be embedded with no edges crossing on a surface with k handles?)

GRAPH LINKING NUMBER
Instance: A graph $G = (V, E)$.
Parameter: A positive integer k.
Question: Can G be embedded into 3-space such that the maximum size of a collection of topologically linked disjoint cycles is bounded by k?

- *Bounded Treewidth and Pathwidth.* These FPT methods are currently "approaching" practicality (see Bodlaender [Bod93]) with the best current parameter function being $f(k) = 2^{ck^2}$. Treewidth, for instance, measures how "treelike" a graph is. If a graph has bounded treewidth then many intractible problems become tractable by dynamic programming/automata techniques.

  In more detail, A *tree-decomposition* of a graph $G = (V, E)$ is a tree $\mathcal{T}$ together with a collection of subsets T_x (called *bags*) of V labelled by the vertices x of $\mathcal{T}$ such that $\cup_{x\in\mathcal{T}} T_x = V$ and 1. and 2. below hold:

1. For every edge uv of G there is some x such that $\{u, v\} \subseteq T_x$.

2. (Interpolation Property) If y is a vertex on the unique path in $\mathcal{T}$ from x to z then $T_x \cap T_z \subseteq T_y$.

(b) The *width* of a tree decomposition is the maximum value of $|T_x| - 1$ taken over all the vertices x of the tree $\mathcal{T}$ of the decomposition. Figure 2.1 gives an example of a tree decomposition of width 2.

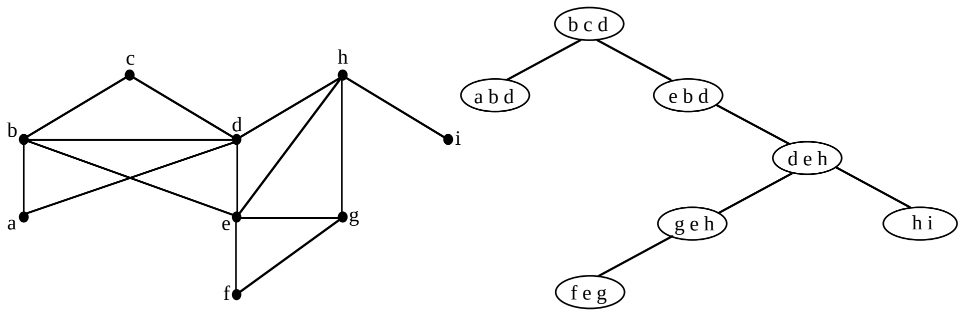

Figure 2.1: Example of Tree Decomposition of Width 2

Historically, authors often tried to get around classical intractability by looking at the problems at hand on special classes of graphs. Authors often discovered that intractable problems became tractable if the problems were restricted to say, "outerplanar" graphs. Such restriction is not purely an academic exercise since, in many practical situations, the graphs that arise do not in fact demonstrate the full pathology of the class of all graphs. [Consider, for instance TRAVELLING SALESMAN concerned with a given city. The graph will almost certainly be of small maximum clique size.] The table (from Van Leeuwin [vanL90]) below lists some families of graphs that have been studied and have treewidth.

Families of Graphs	Bound on Treewidth
Trees	1
Almost Trees (k)	$k+1$
Partial k-trees	k
Bandwidth k	k
Cutwidth k	k
Planar of Radius k	$3k$
Series Parrallel	2
Outerplanar	2
Halin	3
k-Outerplanar	$3k-1$
Chordal with Maximum Clique Size k	$k-1$
Undirected Path With Maximum Clique Size k	$k-1$
Directed Path With Maximum Clique Size k	$k-1$
Interval With Maximum Clique Size k	$k-1$
Proper Interval With Maximum Clique Size k	$k-1$
Circular Arc With Maximum Clique Size k	$2k-1$
Proper Circular Arc With Maximum Clique Size k	$2k-2$

Many of the algorithms for the classes above relied on heroic case analysis. We now know the key fact is that the bags of the tree decomposition give a kind of roadmap to the construction of the graph, and, more importantly, keep track of that part of the graph for which information must be kept in some algorithm. One can think of the algorithm as processing information in the bags down the branches of the tree decomposition. This picture can be formalized because we can associate with a graph of bounded treewidth a standard *parse tree*. The idea is that given some property such as HAMILTONICITY, we will build a kind of tree automaton. This will be used to accept or reject the relevant graph by feeding into the automaton the relevant parse tree of the graph. The culmination of this approach is Courcelle's Theorem [Co90] below. Recall the monadic second order logic is a two sorted logic with the usual logical connectives, vertex and edge variables, and variables for sets of vertices and sets of edges, together with incidence relations. Quantification is allowed over vertices and edges as well as over the sets of vertices and edges. Many properties are expressible in this logic. For instance, a graph G is Hamiltonian iff the edges of G can be partitioned into two sets *red* and *blue* such that each vertex has exactly two incident red edges, and the subgraph induced by the red edges spans G. It is routine to write this description out in monadic second order logic. This all leads to Courcelle's Theorem below.

Theorem 2.1 (Courcelle [Co90]) *Any property of graphs definable in monadic second order logic is recognizable for bounded treewidth, in the sense that we can build an automaton as above accepting graphs satisfying the property.*

Courcelle's Theorem gives a linear time FPT algorithm for such properties.

- *Color-Coding (Hashing).* This is a general method of devising FPT algorithms using k-perfect families of hash functions derived from sophisticated de-randomization techniques developed by Alon, Yuster and Zwick in [AYZ94] after earlier work of Dennenberg and Gurevich [DGS86]. The method works quite well for certain kinds of problems, such as finding small subgraphs in a graph (such as paths, cycles and matchings). The resulting algorithms offer significant practical parameter ranges and no sources of large hidden constants.

  The main result of [AYZ94] proves that for every k and n there is a family $\mathcal{H}$ of functions $h : \{1, ..., n\} \to \{1, ..., k\}$ with the property that for any k-element set $S \subseteq \{1, ..., n\}$ there is some $h \in \mathcal{H}$ that is injective on S. The size of of the family $\mathcal{H}$ of hash functions is bounded by $|\mathcal{H}| = O(2^{ck} \log n)$ where c is a small constant independent of k and n.

  For a simple example of how this method works, consider the problem of 3-DIMENSIONAL MATCHING, which is one of the six NP-complete problems singled out by Garey and Johnson as particularly useful starting points for NP-completeness reductions. This problem (essentially) takes as input a set T of triples $T \subseteq A \times B \times C$ and a positive integer k, and the question is whether there is a subset $S \subseteq T$ of k triples such that if $t = (a, b, c) \in S$ and $t' = (a', b', c') \in S$, then $a \neq a'$ and $b \neq b'$ and $c \neq c'$. In other words, the question is whether there is a set of k of the triples of T that are pairwise distinct in each component. Note that if we ask the same question about 2-tuples rather than 3-tuples, then the problem can be solved in polynomial time by computing a maximum matching in a graph.

To solve this question by the method of color-coding, note that if there is a solution set S of k triples, then this involves at most $3k$ elements in $A \cup B \cup C$. Let $n = |A \cup B \cup C|$, and let $\mathcal{H}$ be the set of perfect hash functions $h : \{1, ..., n\} \to \{1, ..., 3k\}$. If there is a solution set S, then at least one of the hash functions in $\mathcal{H}$ must assign each of the $3k$ elements of S a different "color". Conversely, suppose we choose a function $h \in \mathcal{H}$ and color the elements of $A \cup B \cup C$. After doing this, we can list all of the different color-triples that result. There can be at most $(3k)^3$ of these. In time that is just a function of k we can determine if there is a set of k of these that are pairwise "color-distinct" in each component. But this implies there is a set of k pre-images in T that are a solution. Checking each function in $\mathcal{H}$ in this way gives us an FPT algorithm for the problem.

- *Elementary Methods: search trees* and *reductions to a problem kernel.* An interesting aspect of these two approaches is that while they are simple algorithmic strategies, they are in some sense "new ideas" that might not immediately come to mind, *because they involve costs that are exponential in the parameter.*

  The idea behind search trees is that we might be able to bound the the size of the nondeterminism in some search as a function of k. A simple minded approach using this idea for VERTEX COVER is to proceed as follows. Take any edge of G, say, xy. Now every vertex cover of G must include one of x or y. Begin a tree woth root λ and first nodes labelled respectively x and y. For the x branch remove all edges covered by x and consider the derived graph (and similarly for the y branch). Proceed inductively. The tree is binary and for a fixed k will have depth k. The time needed to see if this results in a vertex cover is $O(2^k|G|)$.

  The idea of the method of problem kernel is to define some sort of reduction which takes an instance G of size n to one of size $f(k)$. In the VERTEX COVER case, for instance, we can remove vertices of high degree since they must be in any vertex cover. When one removes all such verteices, of which there can be at most k one should get a relatively small graph, since one can demonstrate that if a graph has only vertices of low degree then it can only have a k element vertex cover if the number of vertices is bounded by something like k^2. If the kernel obtained by the first reduction is small, then one checks for a vertex cover of the relevant size by examining all subsets (or by some other means). The reader should note that this gives an algorithm with the parameter function $f(k)$ as an *additive* constant. The two methodologies are compatible in the sense that often one uses first kernalization and then search trees.

Just these naive methods gives an algorithm with running time $O(|G| + 2^k k^2)$ for VERTEX COVER. These simple strategies have been successfully applied to a wide variety of problems such as PLANAR DOMINATING SET, FEEDBACK VERTEX SET, FACE COVER NUMBER FOR PLANAR GRAPHS, (see [DF98]), MIMIMUM FILL-IN (Kaplan, Tarjan and Shamir [KST94]), for search trees, and k-LEAF SPANNING TREE, ([DF98]), various $\Pi_{i,j,k}$-GRAPH MODIFICATION PROBLEMS (Leizhen Cai [LeC95]), SET BASIS, UNIQUE HITTING SET ([DF98]), and various phylogenetic tree metric problems (Allen [Al98]).

2.4 Why This Is Interesting: The Deal with the Devil

The above discussion focuses on the fact that parameterized complexity describes a a new paradigm for seeking tractability in computational problems. We call this *The Deal with the Devil of Intractability.* The basic idea is that we are trying to solve some important natural problem, and these (as we now know) are almost always NP-complete or PSPACE complete, or worse. What to do? Lets *actively search for a parameter to absorb the intractability* and make the problem feasible by the slice.

Another fundamental aspect of this line of attack we call *Lichenism.* By this we mean that initial efforts may yield an FPT algorithm that is useful only for a small range of the parameter k, but this can (we have seen for many problems) be subsequently improved, as we gradually expand the "viable margin" on the "rock of intractability" that we are dealing with (perhaps a planetary metaphor would speak more eloquently of this point of view).

Given the general abundance of "bad news" about computational tractability, an effort to slice the problem in various ways and apply FPT methods *can't hurt!* Using these we techniques have improved many known algorithms. For instance, using kernelization we forst improved VERTEX COVER from $2^k|G|$ to $|G| + 2^k k^2$. If kernelization can be applied to a problem (see the above described kernelization for VERTEX COVER for example) then surely dealing with the smaller problem that results can only make our life easier.

Another point worth mentioning is that many heuristic algorithms in the literature *already use* these techniques. The FPT notion "points the way" and systematizes much current practice in practical computing.

2.5 Intractability and Its Applications

Before we look towards the future, and compare this approach with other coping strategies, it is appropriate to also examine the phenomenom of parametric intractability.

There are two ingredients for any intractability theory.

- The indentification of hard problems.
- Reductions to calibrate problems.

Reductions don't pose much of a problem. What is needed is a reduction that preserves the "slicewise" complexity of parameterized languages. The basic working definition used in concrete reductions is that there is a constant c, and three functions

- $k \mapsto k'$, an arbitary function,
- $k \mapsto f(k)$, another arbitary function,
- $\langle x, k\rangle \mapsto \langle x', k'\rangle$, a function computable in time $f(k)|x|^c$, such that

$$\langle x, k\rangle \in L \text{ iff } \langle x', k'\rangle \in L'$$

There are other types of reductions but the one above suffices for most concrete applications. We call this type a *standard reduction.* Here is an example of a standard parameterized reduction. Consider the following two problems.

(k–)INDEPENDENT SET
Input: A graph G.
Parameter: A positive integer k.
Question: Does G have a set of k vectors $\{x_1, \ldots, x_k\}$ such that all $i \neq j$, x_i is not adjacent to x_j?

(k–)MAXIMAL IRREDUNDANT SET
Input: A graph G.
Parameter: A positive integer k.
Question: Does G have a set X of k vertices such that for each member x of X there is a $y \in V(G)$ such that either $y = x$ or (x, y) is an edge, and y is not adjacent to nor a member of S, and furthermore S is maximal with this property?

We will show that INDEPENDENT SET $\leq$ MAXIMAL IRREDUNDANT SET via a standard reduction. We remark that H. Bodlaender and D. Kratsch have proven the apparently stronger result that WEIGHTED CNF SATISFIABILITY $\leq$ MAXIMAL IRREDUNDANT SET. (see below)

To prove our result, given a graph G and a k we construct a graph H and a k' so that G has a k element independent set iff H has a maximal irredundant set of size k'. H is constructed as follows. Construct k blocks consisting of $k+1$ columns each consisting of $n = |G|$ points. So column i in block j consists of n points $x(j, i, t)$ for $t = 1, ..., n$. The idea is that a block represents a choice for a vertex in the independent set. Now within a block join all vertices except those with the same last coordinate. (We refer to this as a *row*.) The first column is not connected to anything else and is called the *dummy column*. Now for each block i pick a column in each block $i' \neq i$ to identify with block i, and similarly identify a corresponding column i'' in block i. Do this in such a way that for each pair (i, j) there is exactly one column $c_i(i, j)$ identified with exactly one column $c_j(i, j)$ in block j, and these columns are used for no other identifications, for any other pair except (i, j). Let (q, r) be some such pair of columns. Now connect each $x(i, q, t)$ to $x(j, r, s)$ iff t and s are adjacent vertices in G.

Now let $k' = k(k+1)$. First if G has an independent set of size k. Let

$\{t_1, ...t_{k-1}\}$ be this independent set. The relevant irredundant set for H is obtained by taking the union of vertices of the t_i'th row of block i, for $i = 1, .., k$. They are each their own private neighbours as the corresponding set in G is independent, and it is maximal by the fact that the blocks are almost cliques. Conversely, take S a size k' maximal irredundant set in H. One can argue that it must be of the form above and hence corresponds to an independent set in G of size k. (There are 2 cases depending on whether some block B has more than k vertices of S. Remember that a maximal irredundant set must also be a dominating set.)

Actually, more instructive than the above example is the following *non*-example. Consider the two problems below.

WEIGHTED CNF SATISFIABILITY
Instance: A CNF formula X,
Parameter: A positive integer k.
Question: Does X have a satisfying assignment of weight k?

WEIGHTED q-CNF SATISFIABILITY
Instance: A q-CNF formula X, i.e., a CNF formula such that each clause has no more than q literals.
Parameter: A positive integer k.
Question: Does X have a satisfying assignment of weight k?

The *classical* reduction of Karp showing that CNF SATISFIABILITY is polynomial time reducible to 3SAT, (i.e. the unweighted versions of the above), works as follows. Let the given CNF formula be $X = c_1 \wedge c_2 \wedge ... \wedge c_n$, where the c_i represent clauses. For those clauses with size ≥ 4, say, $c_i = \{y_1, ..., y_m\}$, add $m - 3$ new dummy variables, and get the clauses

$$\{y_1, y_2, z_1\}, \{\overline{z_1}, y_3, z_2\}, \{\overline{z_2}, y_4, z_3, \}, ...$$

Then the original clause is satisfiable iff there is some assigmment of the new clauses that makes one of the original literals true. *The key thing is that this reduction does not preserve the Hamming weight of the truth assignments.* Thus, this is *not* a parameterized reduction.

We conjecture that there is *no* parameterized reduction between WEIGHTED CNF SATISFIABILITY and WEIGHTED q-CNF SATISFIABILITY. This is why we regard the Bodlaender-Kratsch result as being more significant than the one above.

However, all of this is somewhat irrelevant if all we are interested in is establishing fixed parameter intractability. What we need is a good measure of what we will regard as intractable. That is, what we need is an analog of Cook's Theorem. This leads us to the basic hardness class $W[1]$. Our analog of Cook's Theorem is based on the following problem.

SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE
Instance: A nondeterministic Turing Machine M and a positive integer k.
Parameter: k.
Question: Does M have a computation path accepting the empty string in at most k steps?

If one believes the philosophical argument that Cook's Theorem provides compelling evidence that SAT is intractible, then one surely must believe the same for the parametric intractability of SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE.

We now know that there are many natural problems of the same parameterized complexity as SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE. These include parameterized versions of CLIQUE, INDEPENDENT SET, and a number of other well studied problems. We remark that given the sensitive nature of the reductions, we seem to get a hierarchy of intractable problems, which can be investigated in terms of the logical depth of formulae and circuits that describe the problems. (See [DF98] or [DF95a, DF95b] for details.)

$$FPT \subseteq W[1] \subseteq W[2] \subseteq ... \subseteq W[SAT] \subseteq W[P] \subset XP.$$

The following table gives the classification of some concrete problems based upon this hierarchy.

2.6 Insight Into Classical Complexity

An interesting aspect of the table above is that it includes several problems that in their unparameterized form are considered unlikely to be NP-hard. An example is the problem of VC-DIMENSION, which is in DTIME($n^{\log n}$). If we believe that $W[1] \neq FPT$ then the $W[1]$-hardness result says that there is no polynomial time algorithm to solve this problem, which is interesting

$W[P]$	LINEAR INEQUALITIES	[ADF95]
	MINIMUM AXIOM SET	[DFKHW94]
	SHORT SATISFIABILITY	[ADF95]
	WEIGHTED CIRCUIT SATISFIABILITY	[ADF95]
	⋮	
$W[SAT]$	WEIGHTED SATISFIABILITY	[ADF95]
	⋮	
$W[t]$, for all t	LONGEST COMMON SUBSEQUENCE ($k =$ NUMBER OF SEQS.,$\|\Sigma\|$) (hard)	[BDFHW95]
	FEASIBLE REGISTER ASSIGNMENT (hard)	[BFH94]
	TRIANGULATING COLORED GRAPHS (hard)	[BFH94]
	BANDWIDTH (hard)	[BFH94]
	PROPER INTERVAL GRAPH COMPLETION (hard)	[BFH94]
	WEIGHTED t–NORMALIZED SATISFIABILITY	[DF95a]
	⋮	
$W[2]$	WEIGHTED $\{0,1\}$ INTEGER PROGRAMMING	[DF95a]
	DOMINATING SET	[DF95a]
	TOURNAMENT DOMINATING SET	[DF95c]
	UNIT LENGTH PRECEDENCE CONSTRAINED SCHEDULING (hard)	[BF95]
$W[1]$	SHORTEST COMMON SUPERSEQUENCE (k)(hard)	[FHK95]
	MAXIMUM LIKELIHOOD DECODING (hard)	[DFVW98]
	WEIGHT DISTRIBUTION IN LINEAR CODES (hard)	[DFVW98]
	NEAREST VECTOR IN INTEGER LATTICES (hard)	[DFVW98]
	SHORT PERMUTATION GROUP FACTORIZATION (hard)	[CCDF96]
	SHORT POST CORRESPONDENCE	[CCDF96]
	WEIGHTED q–CNF SATISFIABILITY	[DF95b]
	VAPNIK–CHERVONENKIS DIMENSION	[DEF93]
	LONGEST COMMON SUBSEQUENCE (k, $m =$ LENGTH OF COMMON SUBSEQ.)	[BDFW95]
	INDEPENDENT SET	[DF95b]
	SQUARE TILING	[CCDF96]
	MONOTONE DATA COMPLEXITY FOR RELATIONAL DATABASES	[DFT96]
	k-STEP DERIVATION FOR CONTEXT SENSITIVE GRAMMARS	[CCDF96]
	CLIQUE	[DF95b]
	SHORT NTM COMPUTATION	[CCDF96]
FPT	FEEDBACK VERTEX SET	[DF95c]
	GRAPH GENUS	[RS85]
	MINOR ORDER TEST	[RS85]
	TREEWIDTH	[Bod96]
	VERTEX COVER	[BFR98]

Table 2.1: A Sample of Parametric Complexity Classifications

new information.

That is, *you can use parameterized complexity it to prove unparameterized problems to not be feasible even though they are probably not NP-complete or are unknown.*

There are also deep connections with approximation issues for classical optimization problems. Recall that an *NP optimization problem* Q is either a *minimization problem* or a *maximization problem* and is given as a 4-tuple (I_Q, S_Q, f_Q, opt_Q), where

(i) I_Q is the set of *input instances.* I_Q is recognizable in polynomial time.

(ii) $S_Q(x)$ is the set of *feasible solutions* for the input $x \in I_Q$ such that there is a polynomial p and a polynomial-time computable relation Π (p and Π only depend on Q) such that for all $x \in I_Q$, $S_Q(x)$ can be expressed as $S_Q(x) = \{y : |y| \leq p(|x|) \wedge \Pi(x, y)\}$;

(iii) $f_Q(x, y) \in N$ is the *objective function* for each $x \in I_Q$ and $y \in S_Q(x)$. The function f_Q is computable in polynomial time;

(iv) $opt_Q \in \{\max, \min\}$.

An *optimal solution* for an input instance $x \in I_Q$ is a feasible solution $y \in S_Q(x)$ such that $f_Q(x, y) = opt_Q\{f_Q(x, z) : z \in S_Q(x)\}$. We write

$$opt_Q(x) = opt_Q\{f_Q(x, z) : z \in S_Q(x)\}.$$

The recent work of Liming Cai, Chen and Bagzan has shown that W[1] provides a vehicle for demonstrating that various problems do not have good approximation schemes of various types unless $W[1] = FPT$.

This is of more than passing interest. Aside from the original work in Garey and Johnson, there have been few advances which enable one to prove that an approximation problem has no good approximation algorithm, even assuming $P \neq NP$. The exception to this is the recent work of Arora et. al. [ALMSS92] who demonstrated that many NP optimization problems *with a certain syntactic description* don't have good approximation algorithms unless $P \neq NP$. The Cai-Chen-Bagzan material demonstrates that, assuming $W[1] \neq FPT$, fixed-parameter intractability of an NP optimization problem implies non-approximability of the problem. This result applies to many problems *not* covered by the Arora el. al. results. Thus, the study of fixed-parameter complexity provides a new and potentially powerful approach to proving non-approximability of NP optimization problems. The connections are not fully explored as yet.

For a relation R from (e.g.) $\{\geq, \leq, =\}$ we can associate a natural decision problem.

DECISION PROBLEM ASSOCIATED WITH THE NP OPTIMIZATION PROBLEM $Q = (I_Q, S_Q, f_Q, opt_Q)$.
Input: $x \in I_Q$.
Parameter: A positive integer k.
Question: Does $R(opt_Q(x), k)$ hold?

We write Q_R for the decision problem associated with R. The complexity of the parameterized problems $Q_=$, $Q_\leq$, and $Q_\geq$ have been studied in the literature. For example, Leggett and Moore [LM81] showed that for many NP optimization problems Q, the problems $Q_\leq$ and $Q_\geq$ are in NP $\cup$ coNP, while the problem $Q_=$ is not in NP $\cup$ coNP unless NP = coNP. Using a simple elementary argument, Liming Cai and Chen [CC97] demonstrated that the problems $Q_=$, $Q_\leq$, and $Q_\geq$ have the same complexity *from the viewpoint of fixed-parameter tractability.* That is, *if any of $Q_=$, $Q_\leq$, and $Q_\geq$ are FPT they all are.* The following theorem was the first to demonstrate a connection between FPT and approximation.

Theorem 2.2 (Cai [Ca92], Liming Cai and Chen [CC97]) *If an NP optimization problem has a fully polynomial-time approximation scheme, then it is fixed-parameter tractable.*

Proof. Suppose that an NP optimization problem $Q = (I_Q, S_Q, f_Q, opt_Q)$ has a fully polynomial-time approximation scheme A. Then, the algorithm A takes as input both an instance $x \in I_Q$ of Q and an accuracy requirement $\epsilon > 0$, and then outputs a feasible solution in time $O(p(1/\epsilon, |x|))$ such that the relative error $\Re_A(x)$ is bounded by ϵ, where p is a two-variable polynomial.

First we assume that Q is a maximization problem. By the remarks above, we only need to show that the problem $Q_\leq$ is fixed-parameter tractable.

Given an instance $\langle x, k\rangle$ for the parameterized problem $Q_{\leq}$, we run the algorithm A on input x and $1/2k$. In time $O(p(2k, |x|))$, the algorithm A produces a feasible solution $y \in S_Q(x)$ such that

$$\Re_A(x) = \frac{opt_Q(x) - f_Q(x,y)}{f_Q(x,y)} \leq \frac{1}{2k} < \frac{1}{k}$$

If $k \leq f_Q(x,y)$, then certainly $k \leq opt_Q(x)$ since Q is a maximization problem. On the other hand, if $k > f_Q(x,y)$, then $k - 1 \geq f_Q(x,y)$. Combining this with the inequality $(opt_Q(x) - f_Q(x,y))/f_Q(x,y) < 1/k$, we get immediately $k > opt_Q(x)$. Therefore, $k \leq opt_Q(x)$ if and only if $k \leq f_Q(x,y)$. Since the feasible solution y can be constructed by the algorithm A in time $O(p(2k, |x|))$, we conclude that the NP optimization problem Q is fixed-parameter tractable.

The case when Q is a minimization problem can be proved similarly by showing that the problem $Q_{\geq}$ is fixed-parameter tractable. $\square$

The most important Corollary to this result is in its application in negative form.

Corollary 2.3 (Cai and Chen [CC97]) *If a NP optimization problem is fixed parameter intractable, then it has no fully polynomial time approximation scheme. In particular under the current working hypothesis that $W[1] \neq FPT$, the NP optimization problems that are $W[1]$-hard under the uniform reduction have no fully polynomial-time approximation scheme.*

Using similar arguments, Bagzan has extended the the Cai-Chen Theorem to a more general class of nonuniform PTAS's. There are a number of applications of Theorem 2.2. One not covered by the classical literature is that of the VC-DIMENSION, which has applications in computational learning theory and has been studied extensively in the literature. VC-DIMENSION is $W[1]$-hard, and so cannot have a fully polynomial time approximation scheme unless $W[1] = FPT$.

As we now know, approximation schemes also have intimate connections with syntactic descriptions of problems. The class MAXSNP was introduced by Papadimitriou and Yannakakis [PY91]. The class comes from consideration of the syntactic form of NP problems via Fagin's Theorem below. Fagin's Theorem is known in finite model theory as $NP = \Sigma_1^1$.

Theorem 2.4 (Fagin's Theorem, Fagin [Fag74]) *NP is the class of all problems that can be put in the form*

$$\{I : \exists S \in 2^{[n]^k} \forall x \in [n]^p \exists y \in [n]^q \phi(I, S, x, y)\}.$$

Here $I \subseteq [n]^m$ is the input relation, x and y are tuples ranging over $[n] = \{1, 2, ..., n\}$, and ϕ is quantifier free.

For example, SATISFIABILITY can be written as

$$\exists T \forall c \exists x[(P(c, x) \wedge x \in T) \vee (N(c, x) \wedge x \notin T)]$$

Papadimitriou and Yannakakis realized that some problems have a form that is *simpler* than the basic Fagin Theorem. We can write 3SAT with one less alternation of quantifiers by assuming that the input consists of four relations C_0, C_1, C_2, C_3 where C_j consists of all clauses with exactly j negative literals. Thus $(x_1, x_2, x_3) \in C_j$ means that there is a clause with $x_1, ..., x_j$ occurring negatively and $x_{j+1}, ..., x_3$ positively. 3SAT is then

$$\exists T \forall (x_1, x_2, x_3)[\wedge_{j=0}^{3}((x_1, x_2, x_3) \in C_j \rightarrow (\wedge_{i \leq j} x_i \notin T \wedge \wedge_{i > j} x_i \in T.)].$$

The problems such as 3SAT which can be expressed as $\exists S \forall x \psi(x, G, S)$ constitute a a class called *simple* NP or SNP. For each $\Pi \in NP$, one can then define the *maximization version* $MAX\Pi$ as

$$\max_S |\{x : \exists y \psi(x, y, G, S)\}|.$$

The immediate relevance of all of the above to our studies comes from the following consequential definition of Papadimitriou and Yannakakis.

A maximization problem $Q = (I_Q, S_Q, f_Q, opt_Q)$ is in the class MAX SNP if its optimum $opt_Q(X)$, $X \in I_Q$, can be expressed as

$$opt_Q(X) = \max_S |\{v \ : \ \psi(v, X, S)\}|$$

where the input instance $X = (U, P^1, \cdots, P^b)$ is described by a finite structure over the finite universe U and P^i is a predicate of arity r_i for some integer $r_i \geq 1$, S is also a finite structure over the universe U, v is a vector of fixed arity of elements in U, and ψ is a quantifier-free formula.

Despite the fact that Papadimitriou and Yannakakis [PY91] proved that any problem in MAX SNP can be approximated to within some constant ratio, Arora et. al. [ALMSS92] proved the beautiful result *that MAX SNP-complete (under L-reductions) optimization problems have no fully polynomial-time approximation scheme unless P = NP.* Because of this we cannot use Theorem 2.2 to demonstrate that MAXSNP problems are FPT. Nevertheless, Liming Cai and Chen were able to establish the fixed-parameter tractability of all problems in MAX SNP.

Theorem 2.5 (Cai and Chen [CC97]) *All maximization problems in the class MAX SNP are fixed-parameter tractable.*

The point of this result is that we can see that some problems are FPT *from syntactic descriptions alone.* Cai and Chen also showed that the important class of minimization problems MIN $\mathrm{F}^{+}\Pi_1(h)$ introduced by Kolaitis and Thakur [KoT95] are fixed-parameter tractable. We refer the reader to [CC97] or [DF98] for further details.

2.7 A Comparison With Other Coping Strategies

As we argued in Downey, Fellows, Stege [DFS98], there are two ways of viewing parametric tractability. The first is easiest to arrive at. In this view, FPT is as a kind of first aid that can sometimes be applied to problems that are NP-hard, PSPACE-hard or undecidable. That is, it can be viewed as a potential means of coping with *classical* intractability.

In [DF98], we argued that there is a more radical second way that one can view parameterized complexity. We can view it as a fundamentally richer and generally more productive *primary framework* for problem analysis and algorithm design, including the design of heuristic and approximation algorithms. We will amplify this point later.

In this section we will compare the parametric point of view with other current approaches of "coping with intractability." There is no doubt that there is a percieved gap between the work of theoretical computer scientists and the work of "practical" ones. It may indeed be a reflection of the societal mind set, but there have been a number of calls for reform especially among theorists ([HL92, Hart94, PGWRS96, AFGPR96]). In particular, there seems a desperate need for the development of general coping strategies for dealing with the ubiquitous nature of computational intractability, and the practical computer scientists don't seem to percieve the theory community as delivering the goods.

Of course, computer science theory has articulated a few general programs for systematically coping with the phenomenon of computational intractability. We list these basic approaches:

• The idea of focusing on average-case as opposed to worst-case analysis of problems.

• The idea of settling for approximate solutions to problems, and of looking for efficient approximation algorithms.

• The idea of using randomization in algorithms.

• The idea of harnessing quantum mechanics, or molecular chemistry, to create qualitatively more powerful computational mechanisms.

To this list of fundamentally mathematical strategies for coping with intractability, in [DF98], we argued the following should be added.

• The idea of devising FPT algorithms for parameterizations of a problem.

• The design of mathematically informed, but perhaps unanalyzable *heuristics*, that are empirically evaluated by their performance on sets of benchmark instances.

The actual state of the practical world of computing is that (with the exception of some areas) there is not much connection to work in theoretical computer science on algorithms and complexity. Overwhelmingly, heuristic algorithms are relied on to deal with the hard problems encountered in most applications. How do the above strategies compare?

Average-Case Analysis

In many applications practitioners would be happy with algorithms having good average-case performance, and this criterion is implicit in the common practice of evaluating heuristic algorithms on sets of benchmarks. The idea that average-case analysis is more realistic than worst-case analysis has been around since the beginnings of theoretical computer science, and its potential role as a method of coping with intractability is discussed by Garey and Johnson in their chapter on this subject in [GJ79].

As we noted in [DFS98], obtaining theorems about average-case complexity tends to be mathematically difficult, even for the simplest of algorithms and distributions. As a consequence, the methods are difficult to apply, and moreover, it is frequently unclear what constitutes a reasonable assumption about the distribution of problem instances. The completeness notion for average-case complexity introduced by Levin [Lev86] also seems to have limited applicability. Another drawback of the program is that it lacks a positive toolkit. If I want to design an algorithm with good average-case performance (however this is evaluated) — how do I do that?

The real strength of average case analysis as a means of coping is that for most applications of computing it is the *right idea* for how complexity should (usually) be measured, and it is what practitioners generally continue to *do* about complexity measurement in practice, though informally. Perhaps what is lacking is a general methodology relating benchmarks with average case behaviour.

Approximation

There has been enormous effort in the last decades towards the development of approximation schemes for combinatorial problems. Much of Garey and Johnson [GJ79] is devoted to explaining the basic ideas of *polynomial time approximation algorithms and schemes.* There were significant early successes for problems such as BIN PACKING and KNAPSACK. However, apart from a few similar results on problems mostly of this same general flavor, it now seems to be clear, on the basis of powerful new proof techniques [ALMSS92], that these results are *not* typical for *NP*-hard and otherwise intractable problems. It now seems to be the case that the majority of natural *NP*-hard optimization problems are also hard to approximate, under the usual assumptions (such as $P \neq NP$).

The great strength of polynomial time approximation as a program for coping with intractability is that for those problems to which it can be applied, it allows for the clever deployment of mathematics and can be a very effective cure for worst-case intractability. It is now also clear that the negative tools for showing non-approximability are mathematically deep and widely applicable.

Polynomial Time Randomization

Randomization is discussed in Chapter 6 of Garey and Johnson as a means of avoiding one of the weaknesses of average-case analysis as a coping strategy — the need to have some knowledge in advance of a realistic distribution of problem instances. An algorithm that flips coins as it works may be able to conform to whatever distribution it is given, and either produce an answer in polynomial-time that is correct with high probability (Monte Carlo randomization), or give an answer that is guaranteed to be correct after what is quite likely to be a polynomial amount of time (Las Vegas randomization).

These seemed at first to be potentially very powerful generalizations of polynomial time. Randomized Monte Carlo and Las Vegas algorithms are a workhorse of cryptography [SS77, GM84], and have important applications in computational geometry [Cl87], pattern matching, on-line algorithms and computer algebra (see [Karp86] and [MR95] for surveys), in part because they are often simple to program.

Despite these successes, it now seems that randomized polynomial time is better at delivering good algorithms for difficult problems that "probably" are in P anyway, than at providing a general means for dealing with intractable problems. There have recently been a a number of important results replacing fast probabilistic algorithms with ordinary polynomial time algorithms through the use of sophisticated derandomization techniques.

The main weakness of randomization (in the sense of algorithms with performance guarantees) as a general program for coping with intractability is that it is unclear whether randomization actually provides real progress against problems that are truly hard. One of the strengths of the program is that it supports a rich positive toolkit of algorithm design and analysis techniques (see [MR95]).

New Forms of Computation: DNA and Quantum Mechanics

Although these programs have been launched with great fanfare and have received headline coverage in the newspapers, they so far offer much less of substance than the other items on this list in terms of a general program for coping with intractability. DNA computing essentially comes down to computation by molecular brute force. Combinatorial explosion quickly forces one to contemplate a very large test tube. It is still unclear whether quantum computers useful for any kind of computation can actually be built. The notion of quantum polynomial time QP is mathematically interesting, but so far appears to be applicable only to a few very special kinds of problems. Expert opinion appears to be gathering around conjectures that QP is not much of an extension of P.

Probably neither of these programs can really be considered to be a serious contender for systematically dealing with intractability in ways that will have real applications in the foreseeable future.

Parameterization

We can trace the idea of coping with intractability through parameterization to early discussions in Garey and Johnson [GJ79], particularly Chapter 4, where it is pointed out that parameters associated with different parts of the input to a problem can interact in a wide variety of ways in producing non-polynomial complexity, and that some forms of intractability might be preferable to others.

¿From the current point of view, we see that the main historical drawback of parameterization as a convincing program has been the lack of *industrial strength FPT algorithms.* Certainly one of the original engines for FPT was provided by the mathematically very deep results of Robertson and Seymour. The resulting algorithms involved constants far larger than the size of the universe.

The main strengths of parameterization as a program are that it does seem to be very generally applicable to hard problems throughout the classical hierarchy of intractability, and it supports a rich toolkit of both positive and negative techniques.

Heuristics

Since heuristic algorithms that work well in practice are now, and have always been, the workhorses of industrial computing, there is no question about the ultimate significance of this program for dealing with intractability. There has recently been a revival of interest in obtaining systematic empirical performance evaluations of heuristic algorithms for hard problems [BGKRS95, Hoo95, JM93, JT96].

The main question is whether this can actually be considered a program of computer science theory in the sense intended for this list. It isn't fundamentally at present a *mathematical research program*, though theorists have sometimes contributed to the design of heuristic algorithms. Mathematical ideas are used, but there are frequently no theorems to prove! No theorems are called for — only empirical performance.

Mindless randomized heuristic algorithms such as simulated annealing are applicable to almost any problem and have become extremely popular. Although it is almost impossible to say anything about the performance of such algorithms from a mathematical point of view, these clearly have become the principle means by which practitioners do what they can about intractability.

Chapter 3

Open Questions and Future Work

3.1 Industrial Strength FPT

We see as a major research program the development of "industrial strength" FPT techniques. This can be illustrated by reviewing the history of results concerning the VERTEX COVER problem.

- 1986. Fellows-Langston used the Robertson-Seymour results to establish a complexity of $f(k)n^3$, where $f(k)$ was a tower of 2's $500k$ high. Furthermore the result was nonconstructive in that only the existence of an algorithm was established.
- 1987. Johnson constructively described an $f(k)n^3$ algorithm for VERTEX COVER based on a combination of tree-decomposition and finite-state dynamic programming techniques, with a much better $f(k)$.
- 1988. Fellows using the method of bounded search trees gave an algorithm running in time $2^k n$.
- 1989. S. Buss using a problem kernel approach gave an algorithm requiring time $O(kn + 2^k k^{2k+2})$.

- 1992. R. Balasubramanian, R. Downey, M. Fellows and V. Raman gave an algorithm with running time $O(kn + 2^k k^2)$ by combining the problem kernel and search tree approaches.

- 1993. Using maximum matching, Papadimitriou and Yannakakis gave an algorithm with running time $O(3^k n)$ and noted that this is P-time for the classical problem if k is $O(\log n)$.

- 1996. Balasubramanian, Fellows and Raman used kernelization and an improved search tree strategy to achieve $O(kn + (4/3)^k k^2)$.

- 1998. Downey, Fellows, Stege [DFS98] gave an $O(kn + 1.31951^k k^2)$ algorithm. The 1996 algorithm above actually gives a constant of 1.32472; this small improvement yields a 21% difference for $k = 60$.

Thus we would like to know whether the kind of progress illustrated above is particular to VERTEX COVER, and we believe the answer is "surely not." The following is the basic challenge.

The Challenge. *Develop methods to make the various FPT methods some of which are currently impractical all practical. Or show this is impossible in general and point out when it can be done.*

Looking at our general techniques for proving things to be FPT, we get the following picture.

- Good candidate: Bounded treewidth. Bring down Bodlaender's constant. Currently the constant in Bodlaender's bounded treewidth k recognition algorithm is 2^{32k^2}, which is impractical. However, experience with small treewidth seems to suggest that finding tree decompositions should be than this. Even an algorithm that quickly gave a treewidth $2k$ decomposition for a treewdith k graph would be very useful.

- Good candidate. More extensive use of Colour Coding and other hashing techniques to replace or improve other approaches. Color-Coding give constants around 2^k, which is not bad. Can they be further improved? Can we use Colour Coding to replace WQO methods? Can Color-Coding be used in the bounded treewidth situation?

- Good candidate. What about linear programming? Linear programming is a powerful engine for producing practical algorithms. It is a notable hole in the book [DF98], mainly because of the ignorance of the authors. Presumably the material from Grötschel, Lovasz, and Schrijver [1] is relevant here.

- Bad candidate. WQO methods. Given the generality and complexity of the proofs of these techniques, it does not seem reasonable that the will ever be *in general* feasible. What would be nice is a real proof of this, perhaps using methods from logic. (e.g. [FRS87]). Also a delineation of when they might be feasible would be nice.

- Good candidate. Kernelization and Bounded Search Trees. It would be nice to explore these methodologies further and figure out further applications for them.

3.2 Connections With Heuristics.

¿From our point of view the principal drawback of heuristics is that they are hand-crafted *ad hoc* techniques. Perhaps that is necessary. But perhaps not. What is interesting is that we have found many instances in the literature and in working with, e.g. computational biologists, where *existing heuristics are in fact* FPT algorithms! For instance, an important problem in computational biology is the STEINER PROBLEM FOR HYPERCUBES. It is not important precisely what this problem is, but after one of the authors described an FPT algorithm for this problem, the biologist replied simply

That's what I already do!

It is our belief that there is something much deeper here.

Challenge. *Based on FPT techniques, develop general methodologies for building industrial strength heuristic algorithms for combinatorial problems.*

As an example of what such a general methodology might look like, note that many FPT algorithms work because of some finite combinatorial basis or bounded search tree. Why do we need to use all the basis? For instance, we know that a graph is planar iff it has K_5 or $K_{3,3}$ as a minor. But in general, for almost all graphs, testing with $K_{3,3}$ suffices.

Can we work this sort of observation into a practical, general methodology? Use partial bases, partial obstruction sets, partial automata?

Similarly, because the kernelization phase simplifies and decreases the size of the problem instance, it is a reasonable first step for any general attack on the unparameterized problem. It would be interesting to know if any systematic connection can be made between the "optimum" size of a problem kernel (which requires a definition) and the *complexity threshold* for the problem [KS94].

3.3 Connections With Classical Complexity

We have already mentioned some connections in an earlier section. The general area we wish to point to is:

Challenge. *Explore how the parametric and the classical complexity frameworks intertwine.*

Already in [CC97] and [DF98] we have some connections. Here are some particular suggestions for further exploration.

- What's the best we can do? Can VERTEX COVER be improved to $O(kn + (1 + \epsilon)^k)$ for arbitrarily small ϵ, or would there be some unexpected consequence?
- Can we use parameterized complexity to establish something about the complexity of concrete problems that are classically unresolved. For instance, can we prove, as we did for VC DIMENSION, that GRAPH ISOMORPHISM is unlikely to be in P?
- Can we reverse any of the approximation connections? Is there a class of problems that have approximation algorithms iff W[1]=FPT?
- Can we develop a parametric notion of approximability? For a concrete example, we can ask if there an fast algorithm for DOMINATING SET that either gives an answer "no dominating set of size k" or produces a dominating set of size $2k$.

3.4 Unknown classifications

There a number of interesting and important parametric problems that have not been classified. Here are some. (We refer the reader to [DF98] for others.)

- GRAPH ISOMORPHISM
  Instance: Graphs G and H.
  Parameter: (Various, e.g. Max degree, max treewidth)
  Question: Is G isomorphic to H.

- (FPT?) TOPOLOGICAL CONTAINMENT
  Instance: A graph $G = (V, E)$.
  Parameter: A graph H.
  Question: Is H topologically contained in G?

- (FPT?) IMMERSION ORDER TEST
  Instance: A graph $G = (V, E)$ and a graph $H = (V', E')$.
  Parameter: A graph H.
  Question: Is $H \leq_{immersion} G$ where $\leq_{immersion}$ denotes th e immersion ordering?

- (FPT?) DIRECTED FEEDBACK VERTEX SET
  Instance: A directed graph $D = (V, A)$.
  Parameter: A positive integer k.
  Question: Is there a set S of k vertices such that each directed cycle of G contains a member of S?

- (FPT?) PLANAR DIRECTED DISJOINT PATHS
  Input: A directed planar graph G and k pairs $\langle r_1, s_1\rangle, ..., \langle r_k, s_k\rangle$ of vertices of G.
  Parameter: A positive integer k.
  Question: Does G have k vertex-disjoint paths $P_1, ..., P_k$ with P_i running from $r - i$ to s_i?

  Known to be solvable in $O(n^{f(k)})$ time by Schrijver.

- (FPT?) PLANAR MULTIWAY CUT
  Instance: A weighted planar graph $G = (V, E)$ with terminals $\{x_1, \ldots, x_k\}$.
  Parameter: A positive integer k.

Question: Is there a set of edges of total weight $\leq k'$ whose removal disconnects each terminal from all the others?

The general version of this problem is NP-complete by Dahlhaus et. al. [DJPSY92]. Best known complexity is $O((4^k)^k n^{2k-1} \log n)$ by [DJPSY92] where it is asked if the problem is FPT

- (W[1] Hard?) FIXED ALPHABET LONGEST COMMON SUBSEQUENCE (LCS)
  Instance: k sequences X_i over an alphabet Σ of fixed size (such as $|\Sigma| = 4$ for DNA sequences), and a positive integer m.
  Parameter: A positive integer k.
  Question: Is there a string $X \in \Sigma^*$ of length m that is a subsequence of each of the X_i

  $W[t]$ hard for all t when the alphabet is bounded by the parameter k. Unfortunately this is not the problem that is really of interest to biologists. A simple dynamic programming algorithm that is widely used in the packages available to molecular biologists runs in time $O(n^k)$, and is frequently applied with $k = 5$ (with the help of supercomputers).

- (W[1] Hard?) BOUNDED HAMMING WEIGHT DISCRETE LOGARITHM
  Instance: An n-bit prime, a generator g of F_p^*, an element $a \in F_p^*$, a positive integer k.
  Parameter: A positive integer k.
  Question: Is there a positive integer x whose binary representation has at most k 1's (that is, x has a Hamming weight of k) such that $a = g$?

- (W[1] Hard?) CROSSING NUMBER
  Instance: A graph $G = (V, E)$.
  Parameter: k
  Question: Can G be embedded in the plane with at most k edges crossing?

- (W[1] Hard?) SHORT CHEAP TOUR
  Instance: A graph $G = (V, E)$, an edge weighting $w : E \to \mathbf{Z}$.
  Parameter: A positive integer k.
  Question: Is there a tour through at least k nodes of G of cost at most S?

- FIXED ALPHABET LONGEST COMMON SUBSEQUENCE (LCS)
  Instance: k sequences X_i over an alphabet Σ of fixed size (such as $|\Sigma| = 4$ for DNA sequences), and a positive integer m.
  Parameter: A positive integer k.
  Question: Is there a string $X \in \Sigma^*$ of length m that is a subsequence of each of the X_i.

  The LCS problem for k sequences is known to be hard for $W[t]$ for all t when the alphabet is bounded by the parameter k by a very intricate reduction — but this is not the problem that is really of interest to biologists. A simple dynamic programming algorithm that is widely used in the packages available to molecular biologists runs in time $O(n^k)$, and is frequently applied with $k = 5$ (with the help of supercomputers). This is an almost ideal target for parameterized complexity analysis or algorithm design — yet it seems to be very difficult to resolve.

3.5 Structural Issues and Analogs of Classical Results

FPT and the parametric universe in general throw up a whole new world to explore. One natural thing to do is to look for inspiration towards the classical world, and then investigate the possibility of analogous structural results in the parametric world. Here are some examples.

One of the deep results of the theory is a *partial* analog of Ladner's theorem [Lad75] that the polynomial time degrees are dense [DF93]. Obtaining a full analog — if it is even true — remains a significant and very challenging open problem.

We currently don't even know a reasonable hypothesis implying that the $W[t]$-hierarchy is infinite. If the hierarchy collapses, does collapse propogate upwards? That is, if $W[1] = W[2]$ for instance, then does entail $W[2] = W[3]$? Or would collapse propagate downwards? Does the hierarch collapse under *randomized reductions*? What seems to be needed are uniform versions of Hastad's switching lemma or Razborov's techniques.

It is important for the reader interested in structural issues to realize that proving parameterized analogs of classical theorems is not at all routine, and that few analogs are currently known. Some would have very interesting concrete consequences. For example, if there is a parameterized analog of Toda's theorem, then the problem UNIQUE CLIQUE of determining whether a graph has a unique k-clique (for parameter k) is as hard as any problem at any level of the $W[t]$ hierarchy [DFR98a]. Currently, however, we lack even the basic probability amplification results that such an analog would presumably require.

What about Oracle results? The ones from [DF93] are somehow unsatisfying. Perhaps there are better models.

What about highness/lowness? This might be relevant to GRAPH ISOMORPHISM.

Finally, what about the structure of FPT? For instance, define a parameterized reduction $\langle x, k\rangle \mapsto x', k'\rangle$ to be (poly, poly) if the function $k \mapsto k'$ is also polynomial time. Then we could look at problems in FPT that only had FPT algorithms with the constant 2^k or 2^{2^k}. Are there complete problems etc? The reductions would be (poly,poly) ones.

Chapter 4

A Final Word

The previous sections have explored the FPT world as a method of coping with classical infeasibility, and thus as a kind of fall-back after an initial pessimistic classification such as NP-completeness. In [DFS98], the authors finsihed with a much more radical view which we would like to summarize below. The view is that perhaps parametric analysis is a better *primary classification tool* than P for combinatorial problems.

The current approach to the analysis of concrete computational problems is dominated by two kinds of effort:

(1) The search for asymptotic worst-case polynomial-time algorithms.

(2) Alternatively, proofs of classical hardness results, particularly NP-hardness.

We expect that these will become substantially supplemented by:

($1'$) The design of FPT algorithms for various parameterizations of a given problem, and the development of associated heuristics.

($2'$) Alternatively, demonstrations of $W[1]$-hardness.

We think this will happen because we are inevitably forced towards something like an *ultrafinitist* [YV70] outlook concerning computational complexity because of the nature of the universe of interesting yet feasible computation. The main point of this outlook is that numbers in different ranges of magnitude should be treated in qualitatively different ways.

The pair of notions ($1'$) and ($2'$) are actually rather straightforward mutations of (1) and (2), and they inherit many of the properties that have made the framework provided by (1) and (2) so successful. We note the following in support of this position.

• The enrichment of the dialogue between practice and theory that parameterized complexity is based on always makes sense. It *always* makes sense to ask the users of algorithms, "Are there aspects of your problem that may typically belong to limited distributional ranges?"

• Fixed-parameter tractability is a more accurate notion of "the good". If you were concerned with inverting very large matrices and could identify a bounded structural parameter k for your application that allows this to be done in time $O(2^k n^2)$, then you might well prefer this *classically exponential-time* algorithm to the usual $O(n^3)$ polynomial-time algorithm.

• The "bad", $W[1]$-hardness, is based on a miniaturization of Cook's Theorem in a way that establishes a strong analogy between *NP* and $W[1]$. Proofs of $W[1]$-hardness are generally more challenging than *NP*-completeness, but it is obvious by now that this is a very applicable complexity measurement.

• Problems that are hard do not just go away. Parameterization allows for several kinds of sustained dialogue with a single problem, in ways that allow finer distinctions about the causes of intractability (and opportunities for practical algorithms, including systematically designed heuristics) to be made than the exploration of the "*NP*-completeness boundary" described in [GJ79].

• Polynomial time has thrived because it is a mathematically rich and productive notion allowing for a wide variety of algorithm design techniques. FPT seems to offer an even richer field of play, in part because it encompasses polynomial time as usually the best kind of FPT result. Beyond this, the FPT objective encompasses a rich and distinctive positive toolkit, including novel ways of defining and exploiting parameters.

• There is some evidence that not only are small polynomial exponents generally available when problems are FPT, but also that simple exponential parameter functions such as 2^k are frequently achievable, and that many of the problems in *FPT* admit kernelization algorithms that provide useful start-ups for *any* algorithmic attack on the problem.

• The complexity of approximation is handled more elegantly than in the classical theory, with $W[1]$-hardness immediately implying that there is no efficient PTAS. Moreover, *FPT* algorithm design techniques appear to be fruitful in the design of approximation algorithms.

• Parameterization is a very broad idea. It is possible to formulate and explore notions such as randomized *FPT* [FK93], parameterized parallel complexity [Ces96], parameterized learning complexity [DEF93], parameterized approximation [BFH97], parameterized cryptosystems based on $O(n^k)$ security, etc.

Finally, we feel that the methodology is widely applicable because of our very nature as people. The relevant parameter may not be obvious, but because of the inherent bounded depth of our minds, the natural situations that give rise to computational problems frequently exhibit distinct regularities and parameterizable constraints. It seems just a matter of looking for the correct parameter.

Bibliography

[ADF95] K. Abrahamson, R. Downey and M. Fellows, "Fixed Parameter Tractability and Completeness IV: On Completeness for $W[P]$ and *PSPACE* Analogs," *Annals of Pure and Applied Logic* 73 (1995), 235–276.

[AF93] K. Abrahamson and M. Fellows, "Finite Automata, Bounded Treewidth and Wellquasiordering," In: *Graph Structure Theory*, American Mathematical Society, Contemporary Mathematics Series, vol. 147 (1993), 539–564.

[Al98] B. Allen, *Subtree Transfer Operations and their Induced Metrics on Evolutionary Trees*, MSc. Thesis, University of Canterbury, 1998.

[AFGPR96] E. Allender, J. Feigenbaum, J. Goldsmith, T. Pitassi and S. Rudich, "The Future of Computational Complexity Theory: Part II," *SIGACT News* 27 (1996), 3–7.

[ALT96] S. Alstrup, P.W. Lauridsen and M. Thorup. "Generalized Dominators for Structured Programs." *Proc. 3rd Static Analysis Symp.*, Springer Verlag Lecture Notes in Computer Science vol. 1145 (1996), 42–51.

[AMOV91] G. B. Agnew, R. C. Mullin, I. M. Onyszchuk and S. A. Vanstone, "An Implementation for a Fast Public-Key Cryptosystem," *J. Cryptology* 3 (1991), 63-79.

[ALMSS92] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, "Proof Verification and Intractability of Approximation Algorithms," *Proceedings of the IEEE Symposium on the Foundations of Computer Science* (1992).

[Ar96] S. Arora, "Polynomial Time Approximation Schemes for Euclidean TSP and Other Geometric Problems," In: *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, 1996.

[AYZ94] N. Alon, R. Yuster and U. Zwick, "Color-Coding: A New Method for Finding Simple Paths, Cycles and Other Small Subgraphs Within Large Graphs," *Proc. Symp. Theory of Computing (STOC)*, ACM (1994), 326–335.

[Ba94] B. Baker, "Approximation Algorithms for NP-Complete Problems on Planar Graphs," *J.A.C.M.* 41 (1994), 153–180.

[Baz95] C. Bazgan, "Schémas d'approximation et complexité paramétrée," Rapport de stage de DEA d'Informatique à Orsay, 1995.

[Bod93] H. L. Bodlaender. "A linear time algorithm for finding tree-decompositions of small treewidth," In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 226–234, 1993.

[BDFHW95] H. Bodlaender, R. Downey, M. Fellows, M. Hallett and H. T. Wareham, "Parameterized Complexity Analysis in Computational Biology," *Computer Applications in the Biosciences* 11 (1995), 49–57.

[BDFW95] H. Bodlaender, R. Downey, M. Fellows and H.T. Wareham, "The Parameterized Complexity of the Longest Common Subsequence Problem," *Theoretical Computer Science A* 147 (1995), 31-54.

[BF95] H. Bodlaender and M. Fellows, "On the Complexity of k-Processor Scheduling," *Operations Research Letters* 18 (1995), 93–98.

[BFH94] H. Bodlaender, M. R. Fellows and M. T. Hallett, "Beyond NP-completeness for Problems of Bounded Width: Hardness for the W Hierarchy," *Proc. ACM Symp. on Theory of Computing (STOC)* (1994), 449–458.

[BFH97] H. Bodlaender, M. Fellows, M. Hallett, "Parameterized Complexity and Parameterized Approximation for Some Problems About Trees," manuscript, 1997.

[BFR98] R. Balasubramanian, M. Fellows and V. Raman, "An Improved Fixed-Parameter Algorithm for Vertex Cover," *Information Processing Letters*, to appear.

[BFRS98] D. Bryant, M. Fellows, V. Raman and U. Stege, "On the Parameterized Complexity of MAST and 3-Hitting Sets," manuscript, 1998.

[BFW92] H. Bodlaender, M. Fellows and T. Warnow, "Two Strikes Against Perfect Phylogeny," in: *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, Springer-Verlag, Lecture Notes in Computer Science vol. 623 (1992), 273–283.

[BGKRS95] R. S. Barr, B. L. Golden, J. P. Kelly, M.G.C. Resende and W. R. Stewart, "Designing and Reporting on Computational Experiments with Heuristic Methods," *J. Heuristics* 1 (1995), 9–32.

[Bod96] H. Bodlaender, "A Linear Time Algorithm for Finding Tree Decompositions of Small Treewidth," *SIAM J. Comp.* 25 (1996), 1305–1317.

[Bry97] D. Bryant. Dissertation.

[Byl94] T. Bylander, "The Computational Complexity of Propositional STRIPS Planning," *Artificial Intelligence* 69 (1994), 165–204.

[LM81] E. Leggett and D. Moore, "Optimization problems and the polynomial time hierarchy," *Theoretical Computer Science,* Vol. 15 (1981), 279-289.

[LeC95] Leizhen Cai, "Fixed-parameter tractability of graph modification problems for hereditary properties," Tecnical Report, Department of Computer Science, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, to appear in *Information Processing Letters*, 1995.

[Ca92] Liming Cai, "Fixed parameter tractability and approximation problems," Project Report, June 1992.

[CC97] L. Cai and J. Chen. "On Fixed-Parameter Tractability and Approximability of NP-Hard Optimization Problems," *J. Computer and Systems Sciences* 54 (1997), 465–474.

[CCDF96] L. Cai, J. Chen, R. G. Downey amd M. R. Fellows, "On the Parameterized Complexity of Short Computation and Factorization," *Arch. for Math. Logic* 36 (1997), 321–337.

[CCDF97] L. Cai, J. Chen, R. Downey and M. Fellows, "Advice Classes of Parameterized Tractability," *Annals of Pure and Applied Logic* 84 (1997), 119–138.

[CD94] K. Cattell and M. J. Dinneen, "A Characterization of Graphs with Vertex Cover up to Five," *Proceedings ORDAL'94*, Springer Verlag, Lecture Notes in Computer Science, vol. 831 (1994), 86–99.

[CDDFL98] K. Cattell, M. Dinneen, R. Downey and M. Fellows, "On Computing Graph Minor Obstruction Sets," *Theoretical Computer Science A*, to appear.

[Ces96] M. Cesati, "Structural Aspects of Parameterized Complexity," Ph.D. dissertation, University of Rome, 1995.

[CF96] M. Cesati and M. Fellows, "Sparse Parameterized Problems," *Annals of Pure and Applied Logic* 62 (1996).

[Cl87] K. L. Clarkson, "New Applications of Random Sampling in Computational Geometry," *Discrete and Computational Geometry* 2 (1987), 195–222.

[Co90] B. Courcelle, "Graph Rewriting: An Algebraic and Logical Approach," in: *Handbook of Theoretical Computer Science, vol. B*, J. van Leeuwen, ed., North Holland (1990), Chapter 5.

[CL95] B. Courcelle and J. Lagergren, "Equivalent Definitions of Recognizability for Sets of Graphs of Bounded Treewidth," *Mathematical Structure in Computer Science* ????

[CT97] M. Cesati and L. Trevisan, "On the Efficiency of Polynomial Time Approximation Schemes," *Information Processing Letters* 64 (1997), 165–171.

[CW95] M. Cesati and H. T. Wareham, "Parameterized Complexity Analysis in Robot Motion Planning," *Proceedings 25th IEEE Intl. Conf. on Systems, Man and Cybernetics.*

[DJPSY92] E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour,and M. Yannakakis, "The complexity of multiway cuts," *STOC* (1992) 241–251

[DGS86] L. Dennenberg, Y. Gurevich and S. Shelah. Definability by constant-depth polynomial-size circuits. *Information and Control* 70 (1986), 216–240.

[DEF93] R. Downey, P. Evans and M. Fellows, "Parameterized Learning Complexity," *Proc. 6th ACM Workshop on Computational Learning Theory* (1993), 51–57.

[DF93] R. Downey and M. Fellows, "Fixed Parameter Tractability and Completeness III: Some Structural Aspects of the W-Hierarchy," in: K. Ambos-Spies, S. Homer and U. Schöning, editors, *Complexity Theory: Current Research*, Cambridge Univ. Press (1993), 166–191.

[DF95a] R. G. Downey and M. R. Fellows, "Fixed Parameter Tractability and Completeness I: Basic Theory," *SIAM Journal of Computing* 24 (1995), 873-921.

[DF95b] R. G. Downey and M. R. Fellows, "Fixed Parameter Tractability and Completeness II: Completeness for W[1]," *Theoretical Computer Science A* 141 (1995), 109-131.

[DF95c] R. G. Downey and M. R. Fellows, "Parametrized Computational Feasibility," in: *Feasible Mathematics II, P. Clote and J. Remmel (eds.)* Birkhauser, Boston (1995) 219-244.

[DF98] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer-Verlag, 1998.

[DFS98] R. Downey, M. Fellows and U. Stege, Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability, (To appear) DIMACS series on Combinatorics in the 21st Century, AMS Publ.

[DFKHW94] R. G. Downey, M. Fellows, B. Kapron, M. Hallett, and H. T. Wareham. "The Parameterized Complexity of Some Problems in Logic and Linguistics," *Proceedings Symposium on Logical Foundations of Computer Science (LFCS)*, Springer-Verlag, Lecture Notes in Computer Science vol. 813 (1994), 89–100.

[DFR98a] R. G. Downey, M. R. Fellows and K. W. Regan, "Parameterized Circuit Complexity and the W Hierarchy," *Theoretical Computer Science A* 191 (1998), 91–115,

[DFR98b] R. G. Downey, M. Fellows and K. Regan. "Threshold Dominating Sets and an Improved Characterization of $W[2]$," *Theoretical Computer Science A*, to appear.

[DFT96] R. G. Downey, M. Fellows and U. Taylor, "The Parameterized Complexity of Relational Database Queries and an Improved Characterization of $W[1]$," in: *Combinatorics, Complexity and Logic: Proceedings of DMTCS'96*, Springer-Verlag (1997), 194–213.

[DFVW98] R. Downey, M. Fellows, A. Vardy and G. Whittle, "The Parameterized Complexity of Some Fundamental Problems in Coding Theory," *SIAM J. Computing*, to appear.

[DKL96] T. Dean, J. Kirman and S.-H. Lin, "Theory and Practice in Planning," Technical Report, Computer Science Department, Brown University, 1996.

[Fag74] R. Fagin, "Generalized first order spectra and polynomial time recognizable languages," in *Complexity of Computations,* (R. Karp, ed.) SIAM-AMS proceedings, Vol. 7 (1974), 43-73.

[Fel97] J. Felsenstein. Private communication, 1997.

[FHK95] M. Fellows, M. Hallett and D. Kirby, "The Parameterized Complexity of the Shortest Common Supersequence Problem," manuscript, 1985.

[FHW93] M. Fellows, M. Hallett and H. T. Wareham, "DNA Physical Mapping: Three Ways Difficult," in: *Algorithms – ESA'93: Proceedings of the First European Symposium on Algorithms*, Springer-Verlag, Lecture Notes in Computer Science vol. 726 (1993), 157–168.

[FK93] M. Fellows and N. Koblitz, "Fixed-Parameter Complexity and Cryptography," *Proceedings of the 10th Intl. Symp. on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, Springer-Verlag, Berlin, Lecture Notes in Computer Science vol. 673 (1993), 121–131.

[FKS98] M. Fellows, C. Korostensky and U. Stege, "Theory-Based Heuristics for Editing Gaps in Multiple Sequence Alignments," manuscript, 1998.

[FL87] M. Fellows and M. Langston, "Nonconstructive Proofs of Polynomial-Time Complexity," *Information Processing Letters* 26(1987/88), 157–162.

[FL88] M. Fellows and M. Langston, "Nonconstructive Tools for Proving Polynomial-Time Complexity," *Journal of the Association for Computing Machinery* 35 (1988) 727–739.

[FL89] M. R. Fellows and M. A. Langston, "An Analogue of the Myhill-Nerode Theorem and its Use in Computing Finite-Basis Characterizations," *Proceedings of the IEEE Symposium on the Foundations of Computer Science* (1989), 520–525.

[FL92] M. R. Fellows and M. A. Langston, "On well-partial-ordering theory and its applications to combinatorial problems in VLSI design," *SIAM J. of Discrete Math.* 5 (1992) 117-126.

[FL94] M. R. Fellows and M. A. Langston, "On Search, Decision and the Efficiency of Polynomial-Time Algorithms," *Journal of Computer and Systems Science* 49 (1994), 769–779.

[FN71] R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence* 2 (1971), 189–208.

[FPT95] M. Farach, T. Przytycka, and M. Thorup. "On the agreement of many trees" *Information Processing Letters 55* (1995), 297–301.

[Fr97] J. Franco, J. Goldsmith, J. Schlipf, E. Speckenmeyer and R.P. Swaminathan, "An Algorithm for the Class of Pure Implicational Formulas,"

[FRS87] H. Friedman, N. Robertson, and Seymour, "The metamathematics of the graph minor theorem," *Logic and Combinatorics* (Arcata, Calif., 1985), 229–261, Contemp. Math., 65, Amer. Math. Soc., Providence, R.I., 1987.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W.H. Freeman, San Francisco, 1979.

[GM75] G. R. Grimmet and C. H. J. McDiarmid, "On Colouring Random Graphs," *Math. Proc. Cambridge Philos. Soc.* 77 (1975), 313–324.

[1] M. Grötschel, L. Lovasz and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization,* Springer–Verlag, (1988).

[GM84] S. Goldwasser and S. Micali, "Probabilistic Encryption," *J. Computer and Systems Science* 28 (1984), 270–299.

[Gur89] Y. Gurevich, "The Challenger-Solver Game: Variations on the Theme of P=?NP," *Bulletin EATCS* 39 (1989), 112–121.

[Hal96] M. Hallett. "An Integrated Complexity Analysis of Some Problems in Computational Biology," Ph.D. dissertation, Department of Computer Science, University of Victoria, 1996.

[Hart94] J. Hartmanis, "About the Nature of Computer Science," *Bulletin EATCS* 53 (1994), 170–190.

[HL92] J. Hartmanis and H. Lin, editors, *Computing the Future: A Broader Agenda for Computer Science and Engineering*, National Academy Press, 1992.

[HM91] F. Henglein and H. G. Mairson, "The Complexity of Type Inference for Higher-Order Typed Lambda Calculi." In *Proc. Symp. on Principles of Programming Languages (POPL)* (1991), 119-130.

[Hoo95] J. N. Hooker, "Testing Heuristics: We Have It All Wrong," *J. Heuristics* 1 (1995), 33–42.

[GHS98] G. Gonnet, M. Hallett and U. Stege, "Vertex Cover Revisited: A Hybrid Algorithm of Theory and Heuristic," to appear.

[IK75] O. H. Ibarra and C. E. Kim, "Fast Approximation for the Knapsack and Sum of Subset Problems," *J. Assoc. Computing Machinery* 22 (1975), 463–468.

[JDUGG74] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey and R. L. Graham, "Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms," *SIAM J. Computing* 3 (1974), 299–325.

[Jo87] D. S. Johnson, "The NP-Completeness Column: An Ongoing Guide," *Journal of Algorithms* 1987.

[JM93] D. S. Johnson and C. McGeoch (eds.), *Network Flows and Matching: First DIMACS Implementation Challenge*, American Mathematical Society, 1993.

[JT96] D. S. Johnson and M. Trick (eds.), *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*, American Mathematical Society, 1996.

[Karp72] R. M. Karp, "Reducibility Among Combinatorial Problems," in: *Complexity of Computer Applications*, Plenum Press, New York (1972), 85–103.

[Karp86] R. M. Karp, "Combinatorics, Complexity and Randomness," *Communications of the ACM* 29 (1986), 98–109.

[KS94] S. Kirkpatrick and B. Selman, "Critical Behavior in the Satisfiability of Boolean Formulae," *Science* 264 (1994), 1297–1301.

[KST94] H. Kaplan, R. Shamir and R. E. Tarjan, "Tractability of Parameterized Completion Problems on Chordal and Interval Graphs: Minimum Fill-In and DNA Physical Mapping," in: *Proc. 35th Annual Symposium on the Foundations of Computer Science (FOCS)*, IEEE Press (1994), 780–791.

[KoT95] P. Kolaitis and M. Thakur, "Approximation properties of NP minimization classes," *J. C. S. S.*, Vol. 50 (1995), 391-411.

[KT92] A. Kornai and Z. Tuza, "Narrowness, Pathwidth and Their Application in Natural Language Processing," *Discrete Applied Mathematics* 36 (1992), 87-92.

[JLW94] T. Jiang, E. Lawler, and L. Wang, "Aligning sequences via an evolutionary tree," in *Proceedings of the 26th Annual Symposium on the Theory of Computing,* ACM Press, 1994, 760-769.

[Lad75] R. Ladner, "On the structure of polynomial time reducibility," *J. Assoc. Comput. Mach.*, Vol. 22 (1975), 155-171.

[Len90] T. Lengauer, "VLSI Theory," in: *Handbook of Theoretical Computer Science vol. A*, Elsevier (1990), 835–868.

[Lev86] L. Levin, "Average Case Complete Problems," *SIAM J. Computing* 15 (1986), 285–286.

[Luks82] E. Luks, "Isomorphism of Graphs of Bounded Valence Can Be Tested in Polynomial Time," *J. Comput. and Systems Sci.* 25 (1982), 42–65.

[LP85] O. Lichtenstein and A. Pneuli. "Checking That Finite-State Concurrents Programs Satisfy Their Linear Specification." In: *Proceedings of the 12th ACM Symposium on Principles of Programming Languages* (1985), 97–107.

[MP94] S. Mahajan and J. G. Peters, "Regularity and Locality in k-Terminal Graphs," *Discrete Applied Mathematics* 54 (1994), 229–250.

[MR98] M. Mahajan and V. Raman, "Parameterizing Above the Guarantee: MaxSat and MaxCut," to appear in *J. Algorithms.*

[MR95] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge Univ. Press, 1995.

[MSL92] D. Mitchell, B. Selman and H. Levesque, "Hard and Easy Distributions of SAT Problems," in: *Proceedings AAAI-92*, AAAI (1992), 459–465.

[NP85] J. Nešetřil and S. Poljak, "On the Complexity of the Subgraph Problem," *Comm. Math. Univ. Carol.* 26 (1985), 415–419.

[PGWRS96] C. Papadimitriou, O. Goldreich, A. Wigderson, A. Razborov and M. Sipser, "The Future of Computational Complexity Theory: Part I," *SIGACT News* 27 (1996), 6–12.

[PY91] C. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," *Journal Comput. Sys. Sci* Vol. 43 (1991) 425–440.

[PY96] C. Papadimitriou and M. Yannakakis, "On Limited Nondeterminism and the Complexity of the VC Dimension," *J. Computer and Systems Sciences* 53 (1996), 161–170.

[RS85] N. Robertson and P. D. Seymour, "Graph Minors: A Survey," in *Surveys in Combinatorics*, I. Anderson, ed. (Cambridge University Press: Cambridge, 1985), 153-171.

[RS96] N. Robertson and P. D. Seymour, "Graph Minors XV. Giant Steps," *J. Comb. Theory Ser. B* 68 (1996), 112–148.

[Sha97] R. Shamir. Private communication, 1997.

[SOWH96] D. L. Swofford, G. J. Olsen, P. J. Waddell and D. M. Hillis, "Phylogenetic Inference," in: D. M. Hillis, C. Moritz and B. K. Mable (eds.) *Molecular Systematics*. Sinauer Associates, Inc. (1996), 407–514.

[SS77] R. Solovay and V. Strassen, "A Fast Monte Carlo Test for Primality," *SIAM J. Computing* (1977), 84–85.

[SW97] M. Steel and T. Warnow,

[Th97] M. Thorup, "Structured Programs Have Small Tree-Width and Good Register Allocation," *Proceedings 23rd International Workshop on Graph-Theoretic Concepts in Computer Science, WG'97*, R.

Möhring (ed.), Springer Verlag, Lecture Notes in Computer Science vol. 1335 (1997), 318–332.

[To91] S. Toda, "PP is as Hard as the Polynomial Time Hierarchy," *SIAM J. Comput.* (1991), 865–877.

[vanL90] L. Van Leeuwin, *Handbook of Theoretical Computer Science, Vol. A*. North Holland, 1990.

[Wilf85] H. Wilf, "Some Examples of Combinatorial Averaging," *Amer. Math. Monthly* 92 (1985), 250–261.

[Yan95] M. Yannakakis. "Perspectives on Database Theory," *Proceedings of the IEEE Symposium on the Foundations of Computer Science* (1995), 224–246.

[YV70] A. S. Yessenin-Volpin, "The Ultraintuitionistic Criticism and the Antitraditional Program for Foundations of Mathematics," in: *Intuitionism and Proof Theory: Proceedings of the Summer Conference at Buffalo, N.Y., 1968*, A. Kino, J. Myhill and R. E. Vesley (eds.), North-Holland, 1970.