Computational Tractability: The View From Mars

Rodney G. Downey, Michael R. Fellows, and Ulrike Stege

ABSTRACT. We describe a point of view about the parameterized computational complexity framework in the broad context of one of the central issues of theoretical computer science as a field: *the problem of systematically coping with computational intractability*. Those already familiar with the basic ideas of parameterized complexity will nevertheless find here something new: the emerging systematic connections between fixed-parameter tractability techniques and the design of useful heuristic algorithms, and also perhaps the philosophical maturation of the parameterized complexity program.

1. Introduction

There are two different ways that one can view the theory of parameterized complexity. The easiest 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 intractability as it is *classically* diagnosed.

The second way that one can view parameterized complexity is as a fundamentally richer and generally more productive *primary framework* for problem analysis and algorithm design, including the design of heuristic and approximation algorithms.

It is interesting and humbling to consider the big picture of the development of theoretical computer science over the past 30 years. Above all, this seems to be dominated by continent-sized discoveries and cartography of various kinds of computational intractability. For some time, the practical folks portrayed in the amusing cartoons of Garey and Johnson [**GJ79**], Chapter 1, have been somewhat disappointed with theoretical computer science, and there have been numerous incidents of "restless drums" in the jungles of computer applications. Such grumblings may have something to do with the recent spate of soul-searching among theorists [**HL92, PGWRS96, AFGPR96**]. Some examples:

Example 1: Yet Another Call for Reform. One of the plenary addresses at the AAAI meeting in 1996 was concerned with the broad theme of how computer science practice and theory interact [**DKL96**]. The discussion in [**DKL96**] can be summarized as:

Rodney G. Downey. Research supported by a grant from the United States/New Zealand Cooperative Science Foundation and by the New Zealand Marsden Fund for Basic Science.

Michael R. Fellows. Research supported by the National Science and Engineering Research Council of Canada.

This a shortened version of a survey that appeared in the AMS-DIMACS Proceedings Volume, *The Future of Discrete Mathematics*, edited by F. Roberts and J. Nešetřil .

(1) Pointing to a particular problem, STRIPS PLANNING, as central to the entire field of artificial intelligence.
(2) Proposing that practitioners and theorists collaborate in an intense analysis of this one problem, to understand what makes it hard, and to come up with something more useful than a *PSPACE*-completeness classification.
(3) Suggesting that the usual framework for concrete complexity analysis is wrong-headed, historically contingent, unnatural (especially worst-case asymptotic analysis), and reflects an unhappy state of interaction between computer science theory and practice.

Regarding the third point, most theorists have probably heard similar charges and complaints from colleagues in applied areas of computer science in their own departments.

Example 2: An Encounter With a Computational Biologist. In recent conversations with a biologist who is heavily involved in combinatorial computing [**Fel97**], the following summary was offered of his interaction with theoretical computer scientists.

> *"About ten years ago some computer scientists came by and said they heard we have some really cool problems. They showed that the problems are* NP*-complete and went away!"*

If this interaction had been more recent, then perhaps the computer scientists would also have proved that the problems are unlikely to have efficient approximation algorithms.

The Pervasive Nature of Computational Intractability and Various Coping Strategies. Naturally, many computer science practitioners would like to shoot the messenger who brings so much bad news! In this difficult situation, computer science theory has articulated a few general programs for systematically coping with the ubiquitous phenomena of computational intractability. We list these famous basic approaches, and add to this list our less well known candidate.

- 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 biomolecular chemistry, to create qualitatively more powerful computational mechanisms.
- The idea of devising fixed-parameter tractable algorithms for parameterizations of a problem.

A list such as this cannot be considered complete without including another coping strategy that mathematical theorists have frequently contributed to. This approach antedates modern complexity-theoretic frameworks and has recently regained some respectability among theorists as reflected in the DIMACS Challenges and in new journals such as the *Journal of Heuristics* and the ACM *Journal of Experimental Algorithms*:

- 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 generally there is not much systematic connection to work in theoretical computer science on algorithms and complexity. It is heuristic algorithms that are relied on in most applications.

2. Fixed-Parameter Tractability

The basic concept of the parameterized complexity framework is that of *fixed-parameter tractability*. The definition is best introduced through concrete examples.

VERTEX COVER
Instance: A graph $G = (V, E)$ and a positive integer k.
Parameter: k
Question: Does G have a vertex cover of size k? That is, is there a subset $V' \subseteq V$ of size at most k such that for every edge $uv \in E$, either $u \in V'$ or $v \in V'$?

DOMINATING SET
Instance: A graph $G = (V, E)$ and a positive integer k.
Parameter: k
Question: Does G have a dominating set of size k? That is, is there a subset $V' \subseteq V$ of size at most k such that every vertex $u \in V$ of G either belongs to V' or has a neighbor in V'?

THE STEINER PROBLEM FOR HYPERCUBES
Instance: A set $S = \{x_i : 1 \leq i \leq k\}$ of binary vectors, $x_i \in \{0,1\}^n$ for $i = 1, ..., k$, and a positive integer m.
Parameter: k
Question: Is there a tree $T = (V, E)$ and a labeling of the vertices of T with elements of $\{0,1\}^n$ such that the following conditions are satisfied? (1) The leaves are labeled 1:1 with the elements of S. (2) The sum over the edges uv of T of the Hamming distance between the labels $l(u) \in \{0,1\}^n$ and $l(v) \in \{0,1\}^n$ is at most m.

THE MAXIMUM AGREEMENT SUBTREE PROBLEM (MAST)
Instance: A set of rooted trees $T_1, ..., T_r$ $(r \geq 3)$ with the leaf set of each T_i labeled 1:1 with a set of species X, and a positive integer k.
Parameter: k
Question: Is there a subset $S \subseteq X$ of size at most k such that T_i restricted to the leaf set $X' = X - S$ is the same (up to label-preserving isomorphism and ignoring vertices of degree 2) for $i = 1, ..., r$?

All of these problems are *NP*-complete and they are described above in the standard way for the parameterized complexity framework. Part of the input (which may be some aggregate of various aspects) is identified as the *parameter* for the problem specification. (In order to consider a parameterized problem classically, just ignore the parameter part of the specification. Note that a single classical problem may give rise to many different parameterized problems.) All of the four problems above can be solved in time $O(n^{f(k)})$ by simple brute force algorithms.

Papadimitriou and Yannakakis showed that VERTEX COVER can be solved in time $O(3^k n)$ [**PY96**]. Balasubramanian, Fellows and Raman gave an algorithm with running time $O((53/40)^k k^2 + kn)$ [**BFR98**]. It is possible to do even better.

Note that since the exponential dependence on the parameter k in the last expression is *additive*, VERTEX COVER is *well-solved for input of any size so long as k is no more than around 60.* The difference between the situation for DOMINATING SET and VERTEX COVER is displayed in Table 1.

	$n = 50$	$n = 100$	$n = 150$
$k = 2$	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. The Ratio $\frac{n^{k+1}}{2^k n}$ for Various Values of n and k.

There are myriad ways in which numbers that are small or moderately large (e.g., $k \leq 40$) arise naturally in problem specifications. For example, 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 [**BDFHW95**]. *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 [**Yan95**]. *Hardware constraints* are a common source of natural parameters. The number of processors or machines to be scheduled may be bounded by a value such as $k \leq 20$. The number of degrees of freedom in a robot motion-planning problem is commonly in the range $k \leq 10$. The number of wiring layers in VLSI chip manufacture is typically bounded by $k \leq 30$. Important distributional parameters may also arise in ways that are not at all obvious. Thorup, for example, has shown that the flow graphs of structured programs for the major computer languages have treewidth $k \leq 7$ [**Th97**].

The Basic Definitions. The basic definitions of parameterized complexity are as follows.

Definition. A *parameterized language* L is a subset $L \subseteq \Sigma^* \times \Sigma^*$. If L is a parameterized language and $(x, y) \in L$ then we will refer to x as the *main part*, and refer to y as the *parameter*. It makes no difference to the theory and is occasionally more convenient to consider that y is an integer, or equivalently to define a parameterized language to be a subset of $\Sigma^* \times I\!N$.

Definition. A parameterized language L is *fixed-parameter tractable* if it can be determined in time $f(k)n^\alpha$ whether $(x, k) \in L$, where $|x| = n$, α is a constant independent of both n and k and f is an arbitrary function. The family of fixed-parameter tractable parameterized languages is denoted *FPT*.

It is somewhat surprising, although the argument is not hard, that *FPT* is unchanged if the definition above is modified by replacing $f(k)n^\alpha$ by $f(k) + n^\alpha$ [**CCDF97**].

"About half" of the naturally parameterized problems cataloged as *NP*-complete in the book by Garey and Johnson [**GJ79**] are in *FPT*, including three of the six basic problems singled out for attention in Chapter 3. It is always possible to parameterize a problem in various ways that are fixed-parameter tractable, yet it is not surprising that many parameterized problems apparently do not belong to *FPT*. The naturally parameterized DOMINATING SET problem defined above is one of these. We can find evidence for fixed-parameter *intractability* by studying the appropriate notion of problem transformation.

Definition. A *parametric transformation* from a parameterized language L to a parameterized language L' is an algorithm that computes from input consisting of a pair (x, k), a pair (x', k') such that:
(1) $(x, k) \in L$ if and only if $(x', k') \in L'$,
(2) $k' = g(k)$ is a function only of k, and
(3) the computation is accomplished in time $f(k)n^{\alpha}$, where $n = |x|$, α is a constant independent of both n and k, and f is an arbitrary function.

An Illustrative Non-Example. It can be helpful to observe how parametric transformations differ from ordinary polynomial-time reductions. Recall that for a graph $G = (V, E)$ on n vertices, a set of vertices $V' \subseteq V$ is a k-clique in G if and only if $V - V'$ is a vertex cover in the complementary graph G' where vertices are adjacent if and only if they are not adjacent in G. This gives an easy polynomial-time reduction of the naturally parameterized CLIQUE problem to the naturally parameterized VERTEX COVER problem, transforming the instance (G, k) of CLIQUE into the instance (G', k') of VERTEX COVER. But this is not a parametric transformation, since $k' = n - k$ is not purely a function of k. The available evidence suggests that there is no parametric transformation in this direction.
An Illustrative Example. There is a fairly elaborate parametric transformation from the naturally parameterized CLIQUE problem to the naturally parameterized DOMINATING SET problem, mapping (G, k) to (G', k') where $k' = 2k$ [**DF95a, DF98**]. The evidence is that there is no such parametric transformation in the other direction.

The essential property of parametric transformations is that if L transforms to L' and $L' \in FPT$, then $L \in FPT$. This naturally leads to a completeness program based on a hierarchy of parameterized problem classes:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[SAT] \subseteq W[P] \subseteq AW[P] \subseteq XP$$

The parameterized analog of *NP* is $W[1]$, and $W[1]$-hardness is the basic evidence that a parameterized problem is likely not to be fixed-parameter tractable. The k-STEP HALTING PROBLEM FOR NONDETERMINISTIC TURING MACHINES is $W[1]$-complete (where the amount of nondeterminism at each step of the Turing machine computation is unbounded). Since the $q(n)$-STEP HALTING PROBLEM is essentially the defining problem for *NP*, the analogy between *NP* and $W[1]$ is quite strong. The parameterized complexity class $W[P]$ is an analog of *PSPACE* [**ADF95**]. The class *XP* is the set of all parameterized languages L for which there are functions f and g and an algorithm to determine whether $(x, k) \in L$ in time $f(k)|x|^{g(k)}$.

How Compelling is Fixed-Parameter Tractability? The notion of *fixed-parameter tractability* is the basic concept of the theory — but how good, really, is this notion of *good complexity behaviour*? It might be objected that Table 1 is misleading, unless *FPT* parameter functions such as 2^k are typical. Certainly functions such as $2^{2^{2^k}}$ are allowed by the definition, and would be impractical for $k = 3$, which suggests that the basic definition allows too much pathology.

There are two main responses. First of all, we are already used to some risk-taking in definitions, since the notion of polynomial time allows for, e.g., $O(n^{12})$ and $O(n^{120})$ and $O(n^{1200})$, all of which are completely impractical. We forget how controversial the notion of asymptotic polynomial time was when first introduced, but the notion has thrived because the universe of *natural* computational problems has been *kind*, in some sense. A parameterized problem is just an ordinary problem for which some aspect of the input has been designated as the parameter. Ignoring the parameter, if the problem can be solved in polynomial time, that is, in time polynomial in both the total input size n and the parameter k, then trivially this is an *FPT* algorithm. In other words, considered classically, *FPT* is a superset of P, and it is intended to be a generalization that allows us to do something for problems that are not in P and that may even be *PSPACE* hard or undecidable. We have to expect to risk something in formulating such a general attack on intractability. The definition simply asks whether the difficulty of the problem can be confined to a function of the parameter, while the other costs are polynomial. How else would one naturally formulate a generalization of P having such ambitions?

The second response is that there are many good *FPT* examples other than VERTEX COVER suggesting that "reasonable" (e.g., single exponential) parameter functions are usually obtainable for natural problems (possibly after some rounds of improvement). For example, consider the problem MAXIMUM SATISFIABILITY where the parameter k denotes the number of clauses to be satisfied. This was shown by Cai and Chen [**CC97**] to be in *FPT* with the parameter function 2^{2ck}, when the clause size is bounded by c. The parameter function was improved by Mahajan and Raman [**MR98**] to ϕ^k (without assuming a bound on the clause size), where ϕ is the golden ratio $(1+\sqrt{5})/2$. Franco and others in [**Fr97**] have shown that the falsifiability problem for pure implicational formulas having k negations is in *FPT* with parameter function k^k. (Can this be improved to 2^k?) Although the type checking problem for the programming language ML is *PSPACE*-complete, this is handled in implementations in linear time with a parameter function of 2^k, where k is the nesting depth of *let*'s, a very natural parameter for this problem (one that explains why the problem did not seem hard in practice). We will describe an *FPT* algorithm for a natural parameterization of the MAXIMUM AGREEMENT SUBTREE problem having the parameter function 3^k. Many more examples can be found in [**DF98**]. The improvement of parameter functions for *FPT* problems seems to be a productive area for research, where many different ideas and techniques can be employed.

The point of view that parameterized complexity adopts can be summarized in a metaphorical picture. There is an assumption that most interesting problems are hard, so we can picture them as stones, or perhaps planets in the generally hostile environment of outer space. The trick is to identify and develop thin zones of computational viability, as suggested in Figure 1.

3. A Review of the Major Coping Strategies

In §1 we noted the fundamental problem that has emerged in the first decades of theoretical computer science:

> *The need to deal in some systematic way with the pervasive phenomena of computational intractability.*

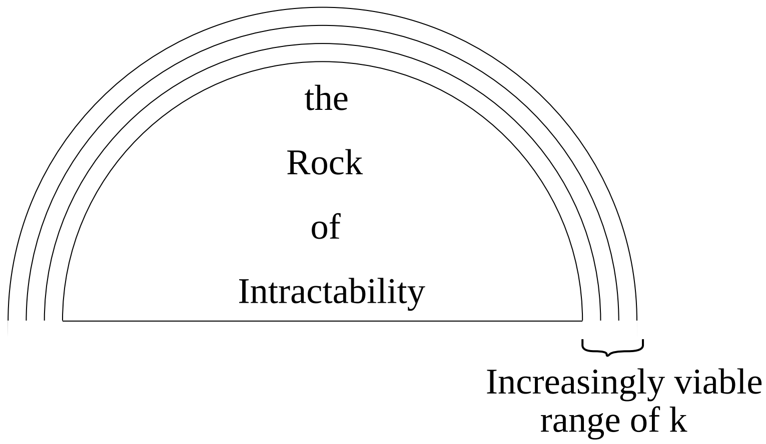

FIGURE 1. The view from Mars: interesting computations occur in thin zones of viability.

We also noted that it is possible to point to five general *mathematical* strategies that have been proposed so far: (1) average-case analysis, (2) approximation, (3) randomization, (4) fundamentally new kinds of computing devices, and (5) parameterization, as well as another, quasi-mathematical coping strategy, (6) heuristics.

In this section, we review the accomplishments and prospects of these programs.

Average-Case Analysis. In many applications practitioners would be happy with algorithms having good average-case performance. This criterion is implicit in the common practice of evaluating heuristic algorithms on sets of representative benchmarks. The idea that average-case analysis is more realistic than worst-case analysis has been around since the beginnings of theoretical computer science. Although we have a few gems of average-case analysis, in general the program seems to be too difficult to carry out for typical hard problems, realistic distributions and sophisticated algorithms. It is also frequently unclear what constitutes a reasonable assumption about the distribution of problem instances, apart from the analytical difficulties of the program.

A completeness notion for average-case complexity has been introduced by Levin [**Lev86**]. This also seems to be difficult to apply, and has only been demonstrated for a few problems. The main weaknesses of average-case analysis as a mathematical program for coping with intractability seem to be:

- In general, it seems to be too difficult to prove the mathematical results that the program calls for.
- The positive toolkit has not developed well. Few general methods are known for designing algorithms with provably good average-case performance. (One interesting example of a general method has been described by Gustedt and Steger [**GS94**].)

Approximation. Most of the discussion in the chapter on coping with *NP*-completeness in [**GJ79**] is devoted to explaining the basic ideas of *polynomial time approximation algorithms and schemes.*

Early on, important polynomial time approximation schemes were found for *NP*-complete problems such as BIN PACKING and KNAPSACK. The program has attracted an enormous amount of effort. 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. The vast majority of natural *NP*-hard optimization problems do not admit efficient approximation schemes under the usual assumptions (such as $P \neq NP$).

While approximation allows for the clever deployment of mathematics and can be a very effective cure for worst-case intractability when it can be applied, from a practical standpoint, most of the results are negative. The main weakness of the program is:

- Most unrestricted classically hard problems cannot be approximated very well.

Randomized Polynomial Time. 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 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 one that is guaranteed to be correct after what is likely to be a polynomial amount of time (Las Vegas randomization).

These seemed at first to be potentially powerful generalizations of polynomial time. Randomized Monte Carlo and Las Vegas algorithms are a workhorse of cryptography, and have important applications in computational geometry, pattern matching, on-line algorithms and computer algebra (see [**Karp86**] and [**MR95a**] for surveys), in part because they are often simple to program. Approximate counting is another area of notable success. Randomization is an important new idea that is now applied in many different kinds of algorithms, including approximations and heuristics.

Despite these successes, it now seems that randomized polynomial time is better at delivering good algorithms for difficult problems that "probably" are in P to begin with, than at providing a general means for dealing with intractable problems. There have recently been a number of important results replacing fast probabilistic algorithms with ordinary polynomial time algorithms through the use of sophisticated derandomization techniques [**Rag88, MR95b**]. Even more to the point, Impagliazzo and Wigderson have recently shown that if there merely exists a language in the exponential time class *EXP* that requires circuits of size $2^{\Omega(n)}$ then $BPP = P$ [**IW97**]. The main weaknesses of randomization (in the sense of algorithms with performance guarantees) as a general program for coping with intractability are:

- With a few exceptions, it does not seem that randomized polynomial time algorithms are any more effective against problems that are truly hard than ordinary polynomial time algorithms.
- Although the positive toolkit of methods for designing and analyzing randomized algorithms is rich, there is no specifically corresponding negative toolkit that can be used in tandem to negotiate problem complexity and guide the design of effective algorithms.

New Forms of Computation: DNA and Quantum Mechanics. Although these programs have been launched with great fanfare, they so far offer much less of substance than the other items on this list in terms of a general mathematically-powered program for coping with intractability. So far, DNA computing essentially amounts to computation by molecular brute force. Combinatorial explosion can quickly force one to contemplate a very large test tube for brute force computations, despite the fact that information can be stored in molecules with a factor of 10^{12} improved efficiency compared to magnetic tape storage. Mathematically, the program seems to come down to the potential for significant constant speed-ups by means of this physical miniaturization of computing.

It is still unclear whether quantum computers useful for any kind of computation can actually be built. The notion of quantum polynomial time is mathematically interesting, but so far appears to be applicable only to a few special kinds of problems.

The main weakness of these approaches are:

- Practical implementation of the new computing devices seems to be far in the future.
- Biomolecular computing is essentially a physical attack on intractability, not a mathematical one.
- It is unclear whether quantum polynomial time is a significant generalization of ordinary polynomial time, except for a few special kinds of problems.

Parameterization. We can trace the idea of coping with intractability through parameterization to early discussions in Garey and Johnson, 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. The internal structure of an intractable problem — the identification of those aspects (parameters) to which the non-polynomial complexity can be confined — is precisely what is at issue in parameterized complexity.

A weakness of the parameterized complexity program is that some of the most general and spectacular positive methods, such as the celebrated results of Robertson and Seymour [**RS85**], yield algorithms having parameter functions that are supremely impractical (e.g., towers of 2's of height described by towers of 2's ...). If *tractability* has friends like these, who needs enemies?

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 intractable classes, and it supports a rich toolkit of both positive and negative techniques. The crucial strike against the program seems to have been:

- The extent to which FPT is really useful is unclear.

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. There have been vigorous developments of new ideas for designing heuristic algorithms, particularly new ideas employing randomization in various ways. These approaches frequently have colorful and extravagant names based on far-fetched analogies in other sciences, such as *simulated annealing*, *genetic algorithms*, *neural nets*, *great deluge algorithms*, and *roaming ants*. Many of these can be considered as variants on the basic technique of local search.

The main problem with considering heuristics as a systematic program for coping with intractability is that it is not a coherent mathematical program. The positive toolkit properly includes voodoo and the kitchen sink. As a program, it doesn't call for any theorems, only empirical performance. The undeniable successes of sometimes "mindless" and generally unanalyzable heuristic algorithms puts computer science theory in an uncomfortable position.

Heuristics based on local search perhaps come the closest to constituting a mathematically articulated general coping strategy for intractability (see the articles in [**AL97**]). There is a negative toolkit (of sorts) based on the notion of *polynomial local search* problems and *PLS*-completeness, introduced by Johnson, Papadimitriou and Yannakakis [**JPY88**]. Although a number of local search problems have been shown to be complete, the reductions are quite demanding (so there aren't very many such results), and there is furthermore a notable peculiarity of this framework. For a concrete example, because the TRAVELING SALESMAN PROBLEM is *NP*-hard, one resorts to a local search heuristic based on the k-Opt neighborhood structure, such as the Lin-Kernighan algorithm, and this is considered to be a particularly successful local search heuristic. Yet, Krentel has shown that for $k = 8$, this neighborhood structure is *PLS*-complete [**Kr90, JMc97**]. This seems like a weapon firing in the wrong direction, or perhaps just a kind of *reiteration* that TSP is fundamentally a hard problem. It is unclear how *PLS*-completeness provides any guidance in designing local search heuristics.

The main difficulty is summarized:

- Although heuristics are the principal coin of the realm in practical computing, the design of heuristics is not well-organized as a mathematical program.

Some general remarks. If these research programs were the *Knights of Theoretical Computer Science* who would deal with the *Dragon of Intractability*, we would seem to have: one well-equipped for battle but immobilized by the weight of the armaments, one in impressive battlegear with mystic insignia riding backwards away from the fray, one armed with lucky charms effective against lizards, one in a baby carriage shaking a rattle, and one on horse, going in the right direction, and armed with a lance — but a lance that is effective only after stabbing the Dragon a number of times bounded by

$$2^{2^{2^{2^{2^{2^{2^{k}}}}}}}$$

where k is ... unfortunately, it doesn't matter ... while the townspeople are cheering on another, who is marching towards the Dragon on foot waving a wooden cudgel (not a knight, actually just a hired ruffian).

Leaving fanciful impressions aside, it seems fair to say that the central problems of theoretical computer science, both structural and concrete, have turned out to be *much* harder to crack than was hoped in the early years of the field.

4. Industrial Strength *FPT*

In this section we sketch the reasons why we think that parameterized complexity offers an inevitable, general, mathematical program for dealing with computational intractability. There are two main points to the argument.

1. The parameterized complexity perspective can lead to useful algorithms in several different ways, including:

- Directly and analytically, when the parameter function is reasonable (i.e., not too fast-growing) and the parameter describes a restriction of the general problem that is still useful.
- Directly and empirically, in cases where the analytic bound on the running time of the *FPT* algorithm turns out to be too crude or pessimistic, that is, where the algorithm turns out to be useful for larger parameter ranges than the parameter function would indicate.
- By supplying systematic guidance in the design of heuristic algorithms in the form of explicit general methods based on *FPT* techniques, using the theory to understand "the internal structure of the complexity of the problem" by identifying those parameterizations of the problem that are *FPT* and those that probably are not.
- Via methodological connections between *FPT* and the design of efficient polynomial-time approximation schemes (where the relevant implicit parameter is $k = 1/\epsilon$, for approximations to within a factor of $(1+\epsilon)$ of optimal) and other approximation heuristics.

2. The notion of *FPT*, in many cases, simply provides a new name and a new perspective on heuristic algorithms *already in use*. Where natural instance distributions exhibit limited parameter ranges, these have frequently been exploited in the design of useful heuristics in a way that the notion of *FPT* names and systematizes.

We have already reported on a steadily growing list of examples of *FPT* problems with "reasonable" parametric costs, such as 2^k. Currently the best known algorithm for VERTEX COVER runs in time $O(n + k^2(1.317303414)^k)$ which directly indicates that it is useful for graphs of any size, so long as $k \leq 70$ (the running time of $O(n + k^2(1.317303414)^k)$ is a result of a more carefully analysis [**Ste98**] of the algorithm presented in [**DFS98**]). In fact, experiments with the algorithm indicate that it is useful for k up to around 200 [**HGS98**].

FPT algorithm design has a distinctive toolkit of positive techniques, including bounded treewidth and pathwidth methods [**Bod96**], well-quasiordering [**RS85**], color-coding [**AYZ94**] and important elementary methods such as search trees and reduction to a problem kernel. This is unfortunately not the place to exposit this interesting collection of methods. What is important to realize, however, is that all of these methods can be systematically adapted in the design of heuristic algorithms. The essential theme is to obtain heuristics from *FPT* algorithms by strategies for *deflating the parametric costs* by truncating or sampling the search trees, obstruction sets, test sets, etc. This is summarized in the following table.

For a specific example, the GATE MATRIX LAYOUT problem is amenable to well-quasiordering methods. For width 3, there are 110 minor order obstructions. Langston et al. found that testing for just one of these gave a new heuristic algorithm superior to those then currently in use in VLSI applications [**LR91**]. Similar ideas have been explored by Gustedt and Steger [**GS94**].

4.1. A Useful Parameterized Algorithm for MAST. We next describe a useful direct *FPT* algorithm for the MAXIMUM AGREEMENT SUBTREE (MAST) problem defined in §1. Apart from the intrinsic interest of this result, it is a nice example of two important points concerning *FPT* algorithms.

FPT Technique	**Heuristic Design Strategy**
Reduction to a Problem Kernel	A useful pre-processing subroutine for any heuristic.
Search Tree	Explore only an affordable, heuristically chosen subtree.
Well-Quasiordering	Use a sample of the obstruction set.
Color-Coding	Use a sample of the hash functions.
Bounded Treewidth	Use Angluin's Theorem and a sample of the test set.

TABLE 2. Some *FPT* Methods and Heuristic Strategies

- Our algorithm for MAST ultimately uses an algorithm for VERTEX COVER as a subroutine. Useful *FPT* algorithms lead to other useful *FPT* algorithms, as one might naturally expect.
- There is already a polynomial time algorithm for MAST for bounded degree trees, so why bother with an exponential *FPT* algorithm? The answer is that the polynomial time algorithm for MAST due to [**FPT95**] runs in time $O(rn^3 + n^d)$ for r phylogenetic trees of maximum degree d on a set of n species. The algorithm we describe requires time $O(c^k rn \log n)$ where c is a constant less than 3. Consequently, this is an example of a situation *where a classically exponential FPT algorithm may be preferable to a polynomial time algorithm.*

Theorem. The parameterized MAST problem can be solved in time $O(c^k rn \log n)$ for r trees on n species.

SKETCH. The input to the problem is a set of rooted binary trees $T_1, ..., T_r$ each having n leaves labeled in 1:1 correspondence with a set X of n species. The problem is to determine if it is possible to delete at most k species from X to obtain a set X' on which all of the trees *agree*. In considering this problem, there is an elegant point of view developed by Bryant [**Bry97**] based on *triples*. If $\{a, b, c\}$ is a set of three species in X, then the restriction of each of the trees T_i to these three species must be one of the three possible alternatives (using parenthetical notation to represent trees): $(a, (b, c))$, $(b, (a, c))$, $(c, (a, b))$, or (a, b, c). If two or more of these three possibilities arise among the T_i, then obviously it will be necessary to eliminate at least one of the species in $\{a, b, c\}$ from X in order to obtain an agreement subtree. In this situation we will refer to $\{a, b, c\}$ as a *conflicted triple* of species.

An argument due to Bryant [**Bry98**] shows that our problem can be reduced in this way to the well-known 3-HITTING SET problem which takes as input a collection $\mathcal{C}$ of 3-element subsets of a set X and a positive integer k, and must answer whether there is a subset $X' \subseteq X$ with $|X'| \leq k$ such that for each $A \in \mathcal{C}$, $A \cap X' \neq \emptyset$. Bryant's argument shows that our problem is equivalent to finding a k-element hitting set for the triples of X that are conflicted with respect to the T_i.

The set of conflicted triples could be computed by exhaustively computing the restrictions of the T_i for each 3-element subset of X, but this is not very efficient. In time $O(n \log n)$ it is possible to either determine that two trees are isomorphic, or identify a conflicted triple. Once a conflicted triple is identified, we can branch in a search tree based on 3 possibilities for resolving the conflict. For example, if the conflicted triple is $\{a, b, c\}$ then we create three branches in the search tree by deleting one element (say a) from X and from all of the trees T_i. We now recursively attempt to determine if the modified instance can be solved with $k' = k - 1$.

There will be at most $O(3^k)$ nodes in the search tree, and the running time due to each node is $O(rn \log n)$, which yields the claimed running time. □

The algorithm sketched above uses 3-HITTING SET implicitly as a means of solving the parameterized MAST problem. An improved but more elaborate *FPT* algorithm for this problem is described in [**BFRS98**], where we reduce MAST to 3-HITTING SET, and in turn devise a good algorithm for 3-HITTING SET using a subroutine for VERTEX COVER, to obtain an algorithm with $c = (1 + \sqrt{17})/2$. A heuristic algorithm that might be useful for a greater range of the parameter k can be adapted from the above *FPT* algorithm simply by not exploring all the branches of the search tree.

4.2. The Steiner Problem for Generalized Hypercubes. In this section we briefly describe another *FPT* result based on the elementary method of *reduction to a problem kernel.* The basic idea of this method, which is very widely applicable (see [**DF98**]) is that by various reduction rules that require time *polynomial in both n and k* it may be possible to reduce the input (x, k) for a parameterized problem to an instance (x', k') where $k' \leq k$ and $|x'|$ is bounded by a function of k. This immediately implies that the problem is in *FPT*, since the question can be answered by brute force for (x', k'). Surprisingly, it can be shown that *every* problem in *FPT* can be kernelized. Such reductions seem to be closely related to ideas explored in [**CKT91, KS94**]. Since such reductions simplify the input, it seems that they cannot hurt, and thus constitute a reasonable "preprocessing" heuristic regardless of what comes after that.

THE STEINER PROBLEM FOR HYPERCUBES is of interest to biologists in the computation of phylogenetic trees under the criterion of minimum evolution / maximum parsimony. The set S corresponds to a set of species, and the binary vectors correspond to information about the species, each component recording the answer to some question (as 0 or 1), such as: "Does it have wings?" or "Is there a thymine at a certain position in the DNA sequence?" Each such bit of information is termed a *character* of the species. In realistic applications the number k of species may usefully be around 40 or 60, while the number of characters n may be very large. We consider a slightly more general problem than the one described in §1.

THE STEINER PROBLEM FOR GENERALIZED HYPERCUBES
Instance: The input the problem consists of the following pieces of information:
(1) A set of complete weighted digraphs D_i for $i = 1, ..., n$, each described by a set of vertices V_i and a function

$$t_i : V_i \times V_i \to I\!N$$

(We refer to the vertices of D_i as *character states*, to D_i as the *character state digraph*, and to t_i as the *state transition cost function* for the ith character.)
(2) A positive integer k_1 such that $|V_i| \leq k_1$ for $i = 1, ..., n$.
(3) A set X of k_2 length n vectors x_j for $j = 1, ..., k_2$, where the ith component

$x_j[i] \in V_i$. That is, for $j = 1, ..., k_2$,

$$x_j \in \Omega = \prod_{i=1}^{n} V_i$$

(4) A positive integer M.
Parameter: (k_1, k_2)
Question: Is there a rooted tree $T = (V, E)$ and an assignment to each vertex $v \in V$ of T of an element $y_v \in \Omega$, such that:

- X is assigned 1:1 with the set of leaves of T,
- The sum over all parent-child edges uv of T, of the *total transition cost* for the edge, defined to be

$$\sum_{i=1}^{n} t_i(y_u[i], y_v[i])$$

is bounded by M?

Theorem 1. The Steiner Problem for Generalized Hypercubes is fixed-parameter tractable.

Proof. We define an equivalence relation $i \sim j$ on the index space $\{1, ..., n\}$ that allows us to combine D_i and D_j and obtain an equivalent smaller instance. In order to define $\sim$ we first define some other equivalences.

Fix $m \leq k_1$ and let l be an integer edge labeling of the complete digraph K_m on m vertices. Let $v_1, ..., v_m$ denote the vertices of K_m. Let T be a rooted tree with k_2 leaves labeled from $v_1, ..., v_m$. Define the *cost* of T with respect to l to be the minimum, over all possible labelings s of the internal vertices of T with labels taken from $\{v_1, ..., v_m\}$, of the sum over the parent-child edges of T of the transition costs given by l on the labels, and write this as

$$\mathrm{cost}(T, l) = \min_s\{\mathrm{cost}(T, s, l)\}$$

If l and l' are integer edge labelings of K_m and T is as above, then define $l \sim_T l'$ if and only if $\exists s$ such that

$$\mathrm{cost}(T, l) = \mathrm{cost}(T, s, l) = \mathrm{cost}(T, s, l') = \mathrm{cost}(T, l')$$

and define $l \sim l'$ if and only $l \sim_T l'$ for all such trees T.

For $i, i' \in \{1, ..., n\}$ define $i \sim i'$ if and only if:

1. $|V_i| = |V_{i'}| = m$ so that the only difference between D_i and $D_{i'}$ is in their arc-labelings l and l', and
2. $l \sim l'$.

The kernelization algorithm can now be described quite simply. Let $\mathcal{I}$ be an instance of the problem. If there are indices $i \neq i'$ for which $i \sim i'$, then modify $\mathcal{I}$ by combining these into one character state digraph with the state transition cost function given by the arc-labeling given $l + l'$, where these are the cost functions for D_i and $D_{i'}$, respectively. Let $\mathcal{I}'$ denote the modified instance.

The correctness of the reduction to the smaller instance is obvious. We need only to note that the equivalence $i \sim i'$ can be determined in time bounded by a function of the parameter and that number of equivalence classes is similarly bounded by a function of the parameter. □

The parameter function for this simple kernelization algorithm is not very good and can probably be much improved. We remark that most of the expense is in determining when two transition digraph indices i and i' are equivalent by testing them on all possible trees with k_2 leaves. This suggests a heuristic algorithm that combines indices when they fail to be distinguished by a (much smaller) random sample of trees and leaf-labelings.

In §1 we reported an encounter with an evolutionary biologist who reported earlier, rather fruitless interactions with theoretical computer scientists who proved that his problems were *NP*-complete and "went away". We claimed that we were different! and that we had a result on one of his computational problems (The Steiner Problem for Hypercubes) that might be of interest. After we described the *FPT* algorithm he said simply [**Fel97**]:

"That's what I already do!"

5. Parametric Intractability

The main classes of parametric complexity are described in the tower:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[SAT] \subseteq W[P] \subseteq AW[P] \subseteq XP$$

As in the theory of *NP*-completeness, there are two kinds of evidence indicating, when a parameterized problem is hard for $W[1]$, that it is unlikely to be fixed-parameter tractable. The first is that given a sufficient amount of unsuccessful effort to demonstrate tractability for various problems in a class, the knowledge that a problem is hard for the class offers a cautionary sociological message.

A second reason for the belief that $W[1]$-hardness implies parametric intractability, is rooted in the following fundamental theorem [**DFKHW94, CCDF96**].

Theorem (Downey-Fellows). The k-Step Halting Problem for Nondeterministic Turing Machines is complete for $W[1]$.

On input consisting of a Turing machine M and a positive integer k (with k being the parameter), the question is whether M can reach a halting configuration in at most k steps. This problem is so generic and opaque that it is hard to imagine that there is any algorithm for it that radically improves on simply exploring the n-branching depth k tree of allowed nondeterministic transitions exhaustively. The theorem can be viewed as essentially a miniaturization of Cook's Theorem.

Clique is a typical starting point for $W[1]$-hardness arguments. It is interesting that most natural parameterized problems seem to belong to a small number of degrees (*FPT*, $W[1]$, $W[2]$, $W[P]$, $AW[*]$ and $AW[P]$; for details see [**DF98**]).

$W[1]$-Hard Means No Good PTAS. One might suspect that parameterized complexity is related to the complexity of approximation. A very good connection is supplied by the following theorem first proved by Bazgan [**Baz95**], and later independently by Cesati and Trevisan [**CT97**], strengthening an earlier result of Cai and Chen [**CC97**].

Definition. An approximation algorithm has an *efficient PTAS* if it computes a solution within a factor of $(1+\epsilon)$ of optimal in time $O(f(\epsilon)n^c)$ for some constant c.

Definition. For a maximization (resp. minimization) problem A, the *induced language* L_A is the parameterized language consisting of all pairs (x, k) where x is an instance of A and $\mathrm{opt}(x) \geq k$ (resp. $\mathrm{opt}(x) \leq k$).

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

TABLE 3. A Sample of Parametric Complexity Classifications (References in [DF98])

Theorem (Bazgan). If A has an efficient PTAS then $L_A \in FPT$.

Thus if the parameterized problem naturally associated with an optimization problem A is hard for $W[1]$, then A cannot have an efficient PTAS unless $FPT = W[1]$. For an example of the power of this result, we can conclude that VC DIMENSION is unlikely to have an efficient PTAS. It is worth noting that some (but by no means all) *NP*-completeness reductions are serendipitously parametric and thus provide demonstrations of $W[1]$-hardness and non-approximability "for free". An important optimization problem that has a PTAS but is not known to have an efficient

PTAS is the Euclidean Traveling Salesman Problem. The PTAS for this problem due to Arora runs in time $O(n^{30/\epsilon})$.

6. The Role of Parameterized Complexity Analysis

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 eventually these will be routinely 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 are inevitably forced towards something like an *ultrafinitist* [**YV70**] outlook concerning computational complexity because of the very nature of the universe of interesting yet feasible computation. The main point of this outlook is that numbers in different ranges of magnitude must 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 (see Table 3) 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 of the empirical circumstance that when polynomial-time algorithms can be devised, one almost always has small exponent polynomials. This is also true for *FPT* algorithms.
- Polynomial time is robust in that it seems to support a strong form of Church's thesis, i.e., that polynomial time on Turing machines is the same as polynomial time on any reasonable computing device. This also seems to be true for *FPT*.

- 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 good 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 (e.g., bounded treewidth techniques in the planar graph PTAS results of Baker [**Ba94**]).
- 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, parameterized cryptosystems based on $O(n^k)$ security, etc.

We feel that the parametric complexity notions, with their implicit ultrafinitism, correspond better to the natural universe of computational complexity, where we find ourselves overwhelmingly among hard problems, dependent on identifying and exploiting thin zones of computational viability. Many natural problem distributions are generated by processes that inhabit such zones themselves (e.g., computer code that is written in a structured manner so that it can be comprehensible to the programmer), and these distributions then inherit limited parameter ranges because of the computational parameters that implicitly govern the feasibility of the generative processes, though the relevant parameters may not be immediately obvious.

It seems that we have a whole new world of complexity issues to explore!

References

[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.

[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.

[AL97] E. Aarts and J. K. Lenstra (eds.), *Local Search in Combinatorial Optimization*, John Wiley and Sons, 1997.

[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*, 13–22, 1992.

[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.

[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.

[BFR98] R. Balasubramanian, M. Fellows and V. Raman, "An Improved Fixed-Parameter Algorithm for Vertex Cover," *Information Processing Letters* 65 (1998), 163–168.

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

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

[Bry97] D. Bryant, "Building Trees, Hunting for Trees, and Comparing Trees," Ph.D. dissertation, Department of Mathematics, Univ. Canterbury, Christchurch, New Zealand, 1997.

[Bry98] D. Bryant, private communication, 1998.

[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 and 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.

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

[CKT91] P. Cheeseman, B. Kanefsky and W. Taylor, "Where the Really Hard Problems Are," *Proc. 12th International Joint Conference on Artificial Intelligence* (1991), 331–337.

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

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

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

[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.

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

[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.

[DFS98] R. G. Downey, M. R. Fellows, and U. Stege. "Parameterized Complexity: A Framework for Systematically Confronting Parameterized Complexity," in *The Future of Discrete Mathematics: Proceedings of the First DIMATIA Symposium, Stirin Castle, Czech Republic, June, 1997*, F. Roberts, J. Kratochvil and J. Nešetřil (eds.), AMS-DIMACS Proceedings Series, 1998.

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

[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.

[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," to appear in *Discrete Applied Mathematics*.

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

[GS94] J. Gustedt and A. Steger, "Testing Hereditary Properties Efficiently on Average," *Orders, Algorithms and Applications: Proceedings of the International Workshop ORDAL'94*, Springer-Verlag, Lecture Notes in Computer Science, vol. 831 (1994), 100–116.

[HGS98] M. Hallett, G. Gonnet, and U. Stege, "Vertex Cover Revisited: A Hybrid Algorithm of Theory and Heuristic," manuscript 1998.

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

[IW97] R. Impagliazzo and A. Wigderson, *Proc. ACM Symposium on Theory of Computing*, 1997.

[JMc97] D. S. Johnson and L. A. McGeoch, "The Traveling Salesman Problem: A Case Study," in: Aarts and Lenstra (eds.), *Local Search in Combinatorial Optimization*, John Wiley and Sons, 1997.

[JPY88] D. S. Johnson, C. H. Papadimitriou and M. Yannakakis, "How Easy is Local Search?" *Journal of Computer and System Sciences* 37 (1988), 79–100.

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

[Kr90] M. W. Krentel, "On Finding and Verifying Locally Optimal Solutions," *SIAM Journal on Computing* 19 (1990), 742–751.

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

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

[LR91] M. Langston and S. Ramachandramurthi, "Dense Layouts for Series-Parallel Graphs," *Proc. Great Lakes Symposium on VLSI* (1991), 14–17.

[MR95a] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge Univ. Press, 1995.

[MR95b] S. Mahajan and H. Ramesh, "Derandomizing Semidefinite Programming Based on Approximation Algorithms," in *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science* (1995), 162–169.

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

[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.

[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.

[Rag88] P. Raghavan, "Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs," *J. Computer and Systems Sciences* 37 (1988), 130–143.

[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.

[Ste98] U. Stege, "Parameterized Complexity in Computational Biology", manuscript (dissertation draft), 1998.

[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.

[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.

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

E-mail address: `rod.downey@vuw.ac.nz`

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

E-mail address: `mfellows@csr.uvic.ca`

Ulrike Stege, Computational Biochemistry Research Group, ETH Zürich, CH-8092 Zürich, Switzerland

E-mail address: `stege@inf.ethz.ch`