Parameterized Complexity for the Skeptic

Rod Downey*
School of Mathematical and Computing Sciences
Victoria University
P. O. Box 600
Wellington, New Zealand

Abstract

The goal of this article is to provide a tourist guide, with an eye towards structural issues, to what I consider some of the major highlights of parameterized complexity.

1 Introduction

Anyone working in the area of parameterized complexity gets rather schizophrenic.[1] We submit papers that get reviews saying that parameterized complexity is now well known so why are you including this introductory stuff, to the other extreme where reviewers say that they have never heard about it.

It is fair to say that the groups that have picked up the material the best are those in rather applied areas where it is perceived as a method of systematic algorithm design : in computational biology, linguistics and the like. It is also why we have seen a number of recent surveys aimed at the heuristic/applied computing community, such as [Ra97, Nie98, DF99b, DFS99, AGN01, Gr02]. On the other hand the area has a lot of manageable and important problems of great significance to the COMPLEXITY community, and this is perhaps not widely known.

1.1 Introduction for the Totally Dubious or "what's in this for me?"

You are someone who is barely aware of this theory, and perhaps even slightly hostile, believing that it does not address any of the central concerns of traditional complexity. Thus I wish to being pointing out a classical problem in the traditional STOC/FOCS/STACS/CCC mould that this theory might offer some hope towards resolving, since it has proven useful in such circumstances before.

A lot of effort has gone into trying to combat intractability. As per Garey and Johnson [GJ79], polynomial time approximation schemes (PTAS's) are one of the main traditional methods. Many ingenious polynomial time approximation schemes have been invented for this reason. Often the wonderful PCP theorem of Arora *et al.* [ALMSS92] shows that no such approximation exists. But sometimes they do. Let's look at some recent examples, taken from some recent major conferences such as STOC, FOCS and SODA, etc.

- Arora [Ar96] gave a $\mathcal{O}(n^{\frac{3000}{\epsilon}})$ PTAS for EUCLIDEAN TSP
- Chekuri and Khanna [CK00] gave a $\mathcal{O}(n^{12(\log(1/\epsilon)/\epsilon^8)})$ PTAS for MULTIPLE KNAPSACK
- Shamir and Tsur [ST98] gave a $\mathcal{O}(n^{2^{2^{\frac{1}{\epsilon}}}-1})$ PTAS for MAXIMUM SUBFOREST
- Chen and Miranda [CM99] gave a $\mathcal{O}(n^{(3mm!)^{\frac{m}{\epsilon}+1}})$ PTAS for GENERAL MULTIPROCESSOR JOB SCHEDULING
- Erlebach *et al.* [EJS01] gave a $\mathcal{O}(n^{\frac{4}{\pi}(\frac{1}{\epsilon^2}+1)^2(\frac{1}{\epsilon^2}+2)^2})$ PTAS for MAXIMUM INDEPENDENT SET for geometric graphs.

Table 1 below calculates some running times for these PTAS's with a 20% error.

Reference	Running Time for a 20% Error
Arora [Ar96]	$\mathcal{O}(n^{15000})$
Chekuri and Khanna [CK00]	$\mathcal{O}(n^{9,375,000})$
Shamir and Tsur [ST98]	$\mathcal{O}(n^{958,267,391})$
Chen and Miranda [CM99]	$> \mathcal{O}(n^{10^{60}})$ (4 Processors)
Erlebach *et al.* [EJS01]	$\mathcal{O}(n^{523,804})$

Table 1. The Running Times for Some Recent PTAS's with 20% Error.

*Partially supported by the Marsden Fund of New Zealand. Thanks to Mike Fellows for helpful comment. Thanks also to Martin Grohe, Rolf Niedermeier, and Detlef Seese who supplied corrections. Finally thanks to Tandy Warnow and Luay Nakhleh for allowing me access to their slides from which I based Section 4.3.2.

[1] Some of us are already known as manic.

Now, by anyone's measure, a running time of $n^{500,000}$ is bad and $n^{9,000,000}$ is even worse. The optimist would argue that these examples are important in that they prove that PTAS's exist, and are but a first foray. The optimist would also argue that with more effort and better combinatorics, we will be able to come up with some $n \log n$ PTAS for the problems. For example, Arora [Ar97] also came up with another PTAS for EUCLIDEAN TSP, but this time it was nearly linear and practical.

But this situation is akin to P vs NP. Why not argue that some exponential algorithm is but a first one and with more effort and better combinatorics we will find a feasible algorithm for SATISFIABILITY. What if a lot of effort is spent in trying to find a practical PTAS's without success? As with P vs NP what is desired, is either an *efficient* PTAS (EPTAS), or a proof that no such PTAS exists. A primary use of NP completeness is to give compelling evidence that many problems are unlikely to have better than exponential algorithms generated by complete search. The trouble is, these examples are in polynomial time. Lower bounds are hard to come by there. Here's where you might make parameterized complexity your friend. Parameterized complexity will allow you in certain circumstances to show that the polynomial time algorithm with the horrible exponent probably has no *feasible* algorithm.

If the reader studies the examples above, they will realize that a source of the appalling running times is the $\frac{1}{\epsilon}$ in the exponent. One method that has worked in such examples is to parameterize the problem by taking $k = \frac{1}{\epsilon}$ as the relevant parameter. As will be shown in Section 5.1, if the underlying assumption of parameterized complexity is correct, a kind of miniaturized Cook's theorem, it is often possible to prove *that the $\frac{1}{\epsilon}$ cannot be removed and hence no EPTAS exists.* In the same way that the underlying assumption for $NP \neq P$ is that there is no way to efficiently decide if a nondeterministic Turing Machine accepts on some path, the underlying assumption in parametric complexity is that there is no way to decide if a nondeterministic Turing machine accepts *in $\leq k$ steps* save than by essentially trying them. We will look at this in section 3.

We will also point out a number of other interesting connections with classical complexity classes that might well be of use to you[2].

1.2 Introduction for the somewhat less dubious

You are a classically trained complexity theorist, and pretty skeptical about the whole thing: here's yet another article with an introduction saying that the authors have invented fire. You are not at all keen on spending a long time re-tooling. Perhaps you are mainly concerned with the deepest questions of complexity theory:

- will you find a decent thesis topic?
- will you get tenure?
- how can you renew your grant? etc

Whilst I certainly don't expect a mass migration into the area, my plan is to try to convince you that parameterized complexity at least can do the following.

(i) It can provide a very useful paradigm in algorithm design, particularly in practical applications of theoretical computer science.
(ii) It is an exceptionally applicable tool for systematically confronting computational intractability, and is one that is rather more easily applied and analyzed than many of the current coping strategies. In particular, it allows for an extended dialog with the problem systematically searching for tractability.
(iii) It focuses our attention on *how* the data is presented to us, and *what kinds* of data will be *relevant to instances of the problem that we are actually interested in.*
(iv) It is absolutely still in its infancy with a host of important open questions some of which might even be solvable, as distinct from many issues in structural complexity.
(v) It can provide very significant insight into things classical. Thus, even if you are a skeptic and are perhaps wedded to PCP, approximability, etc, then I would like to convince you that parametric complexity could add to your tool kit in a very useful way.
(vi) It is not actually that hard, nor that foreign, once you have a little paradigm shift.
(vii) It can provide an almost limitless supply of thesis topics.

1.3 What is parameterized complexity, and what motivates it?

Anybody working in software engineering will know that it is important to design tools specific to the type of problem at hand. Suppose that you are concerned with relational databases. Typically the database is huge, and the queries are relatively small. Moreover, "real life" queries are queries *people* actually ask. Hence, such queries tend to be also of low logical complexity. The *main idea* of parameterized complexity is to design a paradigm that will address complexity issues in the situation where we know in advance that certain parameters will be likely bounded and this might significantly affect the complexity. Thus in the database example, an algorithm that works very efficiently for small formulas with low logical depth might well be perfectly acceptable in practice.

Thus, parameterized complexity is a refined complexity analysis, driven by the idea that in real life data is often given to us naturally with an underlying structure which we

[2] For instance, under a parametric assumption a little weaker than the one above, (see Section 8) one can show that k-INDEPENDENT SET, and k-DOMINATING SET cannot be solved in $DTIME(2^{o(k)})$.

might profitably exploit. The idea is not to replace PTIME as the *underlying* paradigm of feasibility, but to provide a set of tools that refine this concept, allowing some exponential aspect in the running times by allowing us either to use the given structure of the input to arrive at feasibility, or to show that the kind of structure is not useful for this approach. For example, in the PTAS's above, all the algorithms are living in PTIME. It is how they live there that counts.

This simple idea is pretty obvious once you think about it. For example, when we teach a first course in automata theory we show the students that regular language acceptance is in linear time. But this is really not quite true: it is only true *if* the language is presented to us as, say, a regular expression, whereas it could be a language presented as the output of a Turing machine, in which case acceptance is *undecidable.* The *point* is that we only really care about regular languages when they are given to us in a structured way, namely via regular expressions.

Parameterized complexity seems widely applicable in both algorithm design and, we believe, it gives insight into structural complexity. This is a subject close to the author's heart. Sadly, however, there remain groups for whom the main ideas are still unknown.

Complexity theory evolved as a theory attempting to understand the resources needed for computational problems. I would argue that parameterized complexity can be a more suitable complexity theory for addressing the computational concerns arising from a number of important areas of computational problems. The main idea of parameterized complexity is that problems often come given with parameters to exploit, implicit or explicit underlying structure. Database theory is one such area. Computational biology is full of similar parameters to exploit. As we will see in section 2, there are also many situation with *hidden* parameters to exploit.

2 The Main Idea

In this section, I will look at our standard examples, and in the next section, we look at a couple of case studies drawn from the literature. When Mike Fellows and I were formulating this theory, we were really driven by our attempts to understand three examples: VERTEX COVER, DOMINATING SET,INDEPENDENT SET. The reader should recall that for a graph G a vertex cover is where vertices cover edges: that is $C = \{v_1, \ldots, v_k\}$ is a vertex cover iff for each $e \in E(G)$, there is a $v_i \in C$ such that $v_i \in e$. They should recall that a dominating set is where vertices cover vertices: $D = \{v_1, \ldots, v_k\}$ is a dominating set iff for all $v \in V(G)$, either $v \in D$ or there is an $e \in E(G)$ such that $e = \langle v_i, v \rangle$ for some $v_i \in D$. Finally an independent set is a collection of vertices no pair of which are connected.

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

Of course, these are some of the basic NP-complete problems identified in Garey and Johnson [GJ79], so are likely intractable.

Now we are analyzing data arising as, for instance, the conflict graph of some problem in, say, computational biology. Because of the nature of the data we know that it is likely the conflicts are at most about 50 or so, but the data set is large, maybe 10^8 points. We wish to eliminate the conflicts, by identifying those 50 or fewer points. Let's examine the problem depending on whether the identification turns out to be a dominating set problem or a vertex cover problem. (The role of INDEPENDENT SET comes later.)

DOMINATING SET. Essentially the only known algorithm for this problem is to try all possibilities. Since we are looking at subsets of size 50 or less then we will need to examine all $(10^8)^{50}$ many possibilities. Of course this is completely impossible.

VERTEX COVER There is now an algorithm running in time $\mathcal{O}(1.286^k + kn)$ ([CKJ01]) for determining if an G has a vertex cover of size k. This has been implemented and is practical for n of unlimited size and k up to around 400 [St00, DRST01].

The issue in a nutshell, is *the manner by which the running time for a fixed k depends on the k. Critically, is k in the exponent of the size of the problem, or independent from that?* Consider the situation of a running time of $\Omega(n^k)$ vs $2^k n$, as exhibited by Table 2 taken from Downey and Fellows [DF99a].

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.

Thus *one* parameterized version of VERTEX COVER is the following:

VERTEX COVER
Instance: A graph $G = (V, E)$.
Parameter: A positive integer k.

Question: Does G have a vertex cover of size $\leq k$?

One new aspect of the parametric approach for someone thinking in classical terms is that the objects under discussion are "two dimensional". Normally we would think of instances of the problems as being divided in increasing size. Now the idea is that we keep this on the horizontal axis, but we explicitly have the second coordinate, as coordinatizing the problem on the vertical axis. Each row might be in some class like PTIME, but the question is, how?. In the illustrative examples, as we go from slice k to slice $k+1$ we remain in linear time for VERTEX COVER, with only the constant varying, and move from $\Omega(n^k)$ to $\Omega(n^{k+1})$ for DOMINATING SET.

A formal working definition[3] runs along the following lines:

A *parameterized language* is $L \subseteq \Sigma^* \times \Sigma^*$ where we refer to the second coordinate as the *parameter.* It does no harm to think of $L \subseteq \Sigma^* \times \mathbb{N}$.

Definition 2.1. *A parameterized language L is* (strongly) fixed parameter tractable (FPT)*, iff there is a computable function f, a constant c, and a (deterministic) algorithm M such that for all x, k,*

$$\langle x, k\rangle \in L \text{ iff } M(x,k) \text{ accepts},$$

and the running time of $M(x,k)$ is $\leq f(k)|x|^c$.

The thing to keep in mind is that an FPT language is in P "by the slice", *and more*: each k-slice is in the *same* polynomial time class via the *same* machine. Flum and Grohe built on the advice view of Cai, Chen, Downey and Fellows [CCDF97], to formalize this intuition by recasting this definition as follows: Let L_k denote the k-th slice of L and $L_k^{(>m)}$ denote $\{\langle x, k\rangle : |x| > m\}$, the part of L_k *from m onwards.* Then Flum and Grohe [FG02a] observed that L is FPT iff there is an algorithm M, a constant c, and a computable function g such that M witnesses that

$$L_k^{(>g(k))} \in DTIME(n^c).$$

The illustrative example is our parameterized VERTEX COVER where

$$k - \text{VERTEX COVER} \in DTIME(n),$$

from some point onwards with $g(k)$ about 2^k.

Naturally we can do this with other classical notions such as eventually LOGSPACE by the slice. See [CCDF97, FG02a], and Section 8.

Whenever this definition is introduced to unfamiliar audiences, there are several questions that arise.

[3] There are a number of different definitions we can use depending on the level of uniformity desired. We will choose the one of most relevance to practical considerations.

(i) *Question.* That's pretty weird. How can you have this arbitrary f in the definition of *parametric feasibility* when it could be Ackermann's function or worse?

Answer. When it was introduced, polynomial time as a central paradigm for analyzing computational feasibility was somewhat controversial. Edmonds makes a special point of discussing this in [Ed65]. The reader of course knows the usual criticisms: what about ridiculously large exponents, what about ridiculously large constants etc. The answer is kind of pragmatic. First PTIME has very nice closure properties that make it mathematically sound. Second, at least until the last 10 years, "real" problems that are in P have feasible solutions, by and large.

We would argue that the same is true of our mathematical idealization, FPT. There are a number of truly applied FPT algorithms, and they are reasonable. However, there are also some general techniques which give theoretical feasibility but for which there are no known really feasible algorithms, nor proofs that no such algorithms exist subject to any reasonable assumption even.

(ii) *Question.* How do you know which parameter to use? Is there a canonical choice?

Answer. Often there is no canonical parameter. Sometimes there is at least one obvious one. However, the beauty is, as we will see, we can often look at the type of data we are provided with and seek a parameter that makes the problem tractable. It is *good* that a problem can have *many* parameterizations. This is the idea behind using this technique as a systematic way to address feasibility.

3 Parametric Intractability

3.1 The basic class

Before we look at examples, and case studies, I would like to mention the basic hardness classes since that is the other key part of the theory. We know what the good is, what is the bad?

The keystone for the theory of NP completeness is the following:

NONDETERMINISTIC TURING MACHINE ACCEPTANCE
Input: A nondeterministic Turing Machine M and a number e.
Question: Does M have an accepting computation in $\leq |M|^e$ steps?

Cook's argument is that a Turing machine is such an opaque object that it seems that there would be no way to decide if M accepts, without essentially trying the paths. If we accept this thesis, then we probably should accept that the following problem is not $\mathcal{O}(|M|^c)$ for any fixed c and is probably $\Omega(|M|^k)$ since again our intuition would be that all paths would need to be tried:

SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE
Input: A nondeterministic Turing Machine M
Parameter: A number k.
Question: Does M have an accepting computation in $\leq k$ steps?

3.2 Reductions

Thus the idea would be to show that DOMINATING SET is likely not FPT by demonstrating that *if we could solve this in time $\mathcal{O}(n^c)$ by the slice, then we could have a $\mathcal{O}(n^c)$ for* SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE. To do this, we need reductions that work in FPT time and take parameters to parameters. The principal *working* definition for this is that of a parametric connection or transformation.

Definition 3.1. *Let L, L' be two parameterized languages. We say that $L \leq_{fpt} L'$ iff there is an algorithm M, a computable function f and a constant c, such that*

$$M : \langle G, k\rangle \mapsto \langle G', k'\rangle,$$

so that
(i) $M(\langle G, k\rangle)$ runs in time $\leq g(k)|G|^c$.
(ii) $k' \leq f(k)$.
(iii) $\langle G, k\rangle \in L$ iff $\langle G', k'\rangle \in L'$.

For simplicity, the reader can think of k' as $f(k)$, for some computable f. A simple example of a parametric reduction is from k-CLIQUE to k-INDEPENDENT SET, where the standard reduction is parametric (a situation not common). The following is a consequence of Cai, Chen, Downey and Fellows [CCDF96], and Downey and Fellows [DF95b].

Theorem 3.2. *The following are hard for* SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE*:* INDEPENDENT SET*,* DOMINATING SET.

The proof is involved, and is omitted. The reader might think that, as per the theory of NP completeness, that all of these problems are reducible to one another. In fact we can show that SHORT NONDETERMINISTIC TURING MACHINE ACCEPTANCE$\equiv_{fpt}$ INDEPENDENT SET. However, we *do not* think that DOMINATING SET$\leq_{fpt}$ INDEPENDENT SET.

Why might this be? The following might be instructive as an example of a distinctly non-parametric reduction. Suppose we are given a boolean formula F, in variables $x_1 \ldots, x_n$. Then the *weight* of an assignment is the number of variables made true by the assignment. Consider the following problem.

WEIGHTED CNF SAT
Input: A CNF formula X.
Parameter: A number k.
Question: Does X have a true assignment of weight k?

Similarly, we can define WEIGHTED 3 CNF SAT where the clauses have only 3 variables. Classically, using a padding argument, we know that CNF SAT$\equiv_m^P$3 CNF SAT. Recall that to do this for a clause of the form $\{q_1, \ldots, q_k\}$ we add extra variables z_j and turn the clause into several as per: $\{q_1, q_2, z_1\}$, $\{\overline{z_1}, q_3, z_2\}$, etc.

Now this is definitely *not* a parametric reduction from WEIGHTED CNF SAT to WEIGHTED 3 CNF SAT because a weight k assignment could go to any other weight assignment for the corresponding clause 3 version.

In fact Downey and Fellows conjecture that there is no reduction at all from WEIGHTED CNF SAT to WEIGHTED 3 CNF SAT. We can prove that DOMINATING SET$\equiv_{fpt}$WEIGHTED CNF SAT. Extending this reasoning further, we can view WEIGHTED CNF SAT as a formula that is a product of sums. We can similarly define WEIGHTED t-POS SAT as the weighted satisfiability problem for a formula X in product of sums of product of sums... with t alternations. And we can define WEIGHTED SAT if we have no restriction on the formula. Downey and Fellows [DF95a] called the collection of parameterized languages fpt-equivalent to WEIGHTED 3 CNF SAT $W[1]$, the collection of languages fpt-equivalent to WEIGHTED CNF SAT $W[2]$, the collection of languages fpt-equivalent to WEIGHTED t-POS SAT $W[t]$, and the collection of languages fpt-equivalent to WEIGHTED SAT $W[SAT]$. There are some other classes $W[P]$, the weighted circuit satisfiability class, and XP which has as its defining problem the class whose k-th slice is complete for $DTIME(n^k)$, this being provably distinct from FPT and akin to exponential time. This gave the W-hierarchy below

$$W[1] \subseteq W[2] \subseteq W[3] \ldots W[SAT] \subseteq W[P] \subseteq XP.$$

Each of these classes contains concrete problems. For instance, XP has k-CAT AND MOUSE GAME and some other games ([DF99a]), $W[P]$ has LINEAR INEQUALITIES, SHORT SATISFIABILITY, WEIGHTED CIRCUIT SATISFIABILITY ([ADF95]) and MINIMUM AXIOM SET([DFKHW94]). Then there are a number of quite important problems from combinatorial pattern matching which are $W[t]$ hard for all t: LONGEST COMMON SUBSEQUENCE (k = number of seqs.,$|\Sigma|$-two parameters) ([BDFHW95]), FEASIBLE REGISTER ASSIGNMENT, TRIANGULATING COLORED GRAPHS, BANDWIDTH, PROPER INTERVAL GRAPH COMPLETION ([BFH94]), DOMINO TREEWIDTH ([BE97]) and

BOUNDED PERSISTENCE PATHWIDTH ([McC03]). Some concrete problems complete for $W[2]$ include WEIGHTED $\{0,1\}$ INTEGER PROGRAMMING, DOMINATING SET ([DF95a]), TOURNAMENT DOMINATING SET ([DF95c]) UNIT LENGTH PRECEDENCE CONSTRAINED SCHEDULING (hard) ([BF95]), SHORTEST COMMON SUPERSEQUENCE (k)(hard) ([FHK95]), MAXIMUM LIKELIHOOD DECODING (hard), WEIGHT DISTRIBUTION IN LINEAR CODES (hard), NEAREST VECTOR IN INTEGER LATTICES (hard) ([DFVW99]), SHORT PERMUTATION GROUP FACTORIZATION (hard). Finally complete for $W[1]$ we have a collection including k-STEP DERIVATION FOR CONTEXT SENSITIVE GRAMMARS, SHORT NTM COMPUTATION, SHORT POST CORRESPONDENCE, SQUARE TILING ([CCDF96]), WEIGHTED q–CNF SATISFIABILITY ([DF95b]), VAPNIK–CHERVONENKIS DIMENSION ([DEF93]) LONGEST COMMON SUBSEQUENCE (k, $m =$ LENGTH OF COMMON SUBSEQ.) ([BDFW95]), CLIQUE, INDEPENDENT SET ([DF95b]), and MONOTONE DATA COMPLEXITY FOR RELATIONAL DATABASES ([DFT96]). This list is definitely not complete, and new arenas of application are being found all the time.

4 Some natural arenas of application

One thing that we did not expect when we began our studies in this area was the applicability of the ideas to many areas particularly as a common generalization of natural heuristics people were using anyway. Perhaps this is why a number of groups working in algorithms and applied computer science have taken up the ideas. I give a small sample in the subsections to follow.

4.1 Databases

As we alluded to in the introduction, databases provide a very natural arena for the applications of the theory. There have been some very attractive applications in this area such as Downey, Fellows, Taylor [DFT96], Papadimitriou and Yannakakis [PY97], and especially Grohe [Gr01a, Gr01b, Gr02]. Readers especially interested in this area are urged to read the entertaining introduction Grohe [Gr02]. There Grohe gives an introduction to the area as "A database theorist's nightmare."

Chandra and Merlin [CM77] introduced the study of the complexity of query languages in the study of database theory. Vardi [Va82] noted that what was important in the study of relational databases was the complexity of the evaluation of a query when the size of the query was fixed as a function of the size of the database, and since then this has been seen as a relevant measure for the study of the complexity of databases.

The point is that, again, the general problem of checking that a relational database satisfies some formula in some reasonable language is PSPACE complete. However, all of the problems for a fixed *size* of input formula are again in PTIME. Thus the standard kind of problem we might look at would be of the form.
Input: A boolean query φ and a database instance I.
Parameter: Some parameter of φ, such as the size of φ.
Problem: Evaluate φ in I.

The first suggestion that parameterized complexity would be a suitable way to address the issues in database query evaluation was in Yannakakis [Ya95].

In [DFT96], and [PY97], it is shown that there are relatively easy reductions to demonstrate even more bad news. The problems are $W[1]$ hard, and hence likely have no feasible algorithms. Papadimitriou and Yannakakis [PY97] systematically also looked at other parameters such as bounding the number of variables following ideas of Vardi [Va95]. They looked at positive queries, conjunctive queries, first order theories and datalog ones and found them to be all $W[1]$ hard and at various levels of the W-hierarchy. Other analyses look at other parametric aspects and give even more bad news. (e.g. Demri, Laroussinie and Schnoebelen [DLS02].)

You might well ask now, with "good" news like this provided by parameterized complexity, what use is it? You could argue that once we knew these problems were NP-hard and likely PSPACE complete. Now we know that even when you bound the obvious parameters then they are *still* hard!

One interpretation is that we should learn to live with this by searching for new coping strategies.

The parametric point of view is to try to cope by finding new, and maybe more appropriate parameters. This is the point of view pursued by Grohe [Gr01a, Gr01b, Gr02]. In the next section, we will look at various graph width metrics, and see that they can be used as a systematic way to parametrically address intractability. The *Gaifman graph* of a relational database is the graph whose vertices are the elements of the active domain of the database instance I, with an edge between the vertices if they lie on the same row of some table for I. (More details can be found in the references below.)

Theorem 4.1 (Frick and Grohe [FrG02], Flum and Grohe [FG02a]). *Let C be a class of relational database instances such that underlying graph of instances in C are any of the following forms: bounded degree, bounded treewidth, bounded local treewidth, planar or have an excluded minor. Then the query evaluation problem for the relational calculus on C is FPT.*

The reader unfamiliar with the graph theoretical terms above is referred to Section 4.2 for more details.

Frick and Grohe [FrG02] have looked beyond relational databases. For instance, XML-documents can be viewed as colored trees, with the color representing the XML-tag. It is also known that the core of standard XML query languages is contained in monadic second order logic. (See Section 4.2 for details on monadic second order logic.)

Theorem 4.2 (attributed in [FrG02] to folklore). *The query evaluation problem for monadic second order logic on the class of colored trees is FPT.*

As we will see in Section 5, Theorem 4.2 has rather limited applicability. More on this later, when we see how parametric complexity connects to classical complexity.

Finally, as Grohe [Gr02] observed, there are also nice known parametric results for temporal logics. Temporal logics such as LTL and CTL* are used for specification languages for automated verification. Gottlob and Koch [GK02] have that the core of XPATH can be viewed as a fragment of CTL*. The following algorithm is practical.

Theorem 4.3 (Lichtenstein and Pneuli [LP85], Emerson and Lei [EL87]). *The evaluation problems for LTL and CTL* on the class of Kripke structures are FPT in time $\mathcal{O}(2^k n)$ where k is the size of the query, and n is the size of the input instance.*

Given that the material has only been investigated by only a few authors, and the area is definitely important, it is clearly one that would merit further attention, particularly how to make Theorem 4.1 practical. (But see Frick and Grohe [FrG02] for parameter dependence as in Section 8.)

4.2 Graph width metrics

Anyone who has done any course in algorithms has seen various algorithms for planar this and bounded degree, dimension, pathwidth, bandwidth, etc that. Clearly, what is going on is some kind of quest to try to map the boundary of intractability, and using some kind of regularity in the data to get tractability.

Planarity is natural since a road map of a city is more or less planar subject to a few exceptions. One could view the number of exceptions as a parameter, or simply view the every increasing genus as the relevant parameter. Similarly degree. How does the running time vary for the problem at hand as the degree varies.

Two sweeping generalizations of the notions of width metrics are found through treewidth and local treewidth. Treewidth is part of the change from *ad hoc* graph theory to structural, topological graph theory which has revolutionized the area in the last decade or so. If you have not seen this before here is the definition.

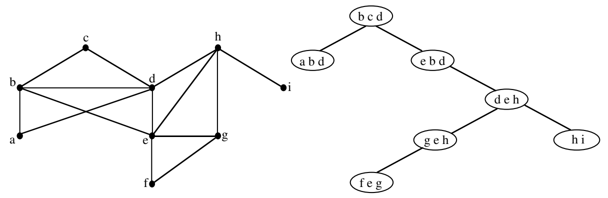

Figure 1. Example of Tree Decomposition of Width 2

Definition 4.4 (Robertson and Seymour [RS86a]). *(a)* *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 labeled by the vertices x of $\mathcal{T}$ such that $\cup_{x \in \mathcal{T}} T_x = V$ and (i) and (ii) below hold:*

(i) For every edge uv of G there is some x such that $\{u, v\} \subseteq T_x$.
(ii) (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.*

(c) The treewidth of a graph G is the minimum treewidth of all tree decompositions of G.

The point of the notion is that it is a measure of how tree-like the graph is. One can similarly define *path decomposition* where the tree $\mathcal{T}$ must be a path. A tree decomposition provides a road map as to how to build a graph from small pieces by gluing them together. Figure 1 gives an example of a tree decomposition of width 2.

Authors often discovered that intractable problems became tractable if the problems were restricted to say, "outerplanar" graphs. As we have seen, 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. Families of graph that have been studied which turn out to have bounded treewidth include Almost Trees (k) (width $k+1$), Bandwidth k (width k), Cutwidth k (width k), Planar of Radius k (width $3k$), Series Parallel (width 2), Outerplanar (width 2), Halin (width 3) k-Outerplanar (width $3k - 1$), Chordal with Maximum Clique Size k (width $k - 1$), and many others.

There is now a large industry devoted to treewidth, and it must be a basic tool now for someone working in algorithms. The principal reason for this is that, using dynamic programming, or automata, many problems which are otherwise intractable become tractable when restricted to the parameterized class of graphs of bounded treewidth. A high level version of this phenomenon is the following.

A standard version of *Monadic Second Order* logic of graphs is a two sorted logic with vertex variables, edge variables (denoted by lower case letters) and variables for sets of vertices and sets of edges (upper case) with incidence relations $v \in V$, $v \in e$, and quantification over these variables. This is a powerful language, where many standard properties of graphs such as Hamiltonicity etc are expressible. For instance, G is 3-colorable would be expressed as $\exists V_1 \exists V_2 \exists V_3 (\forall v [\wedge_{i \neq j} (v \notin V_i \vee v \notin V_j) \wedge (\vee_{i \in \{1,2,3\}} (v \in V_i)] \wedge \forall e (\forall v_1, v_2 (v_1 \neq v_2 \wedge v_1 \in e \wedge v_2 \in e \rightarrow (\wedge_{i \neq j} (v_1 \notin V_i \vee v_2 \notin V_j))))$.

Theorem 4.5 (Courcelle [Co87]). *Suppose that φ is any sentence in monadic second order logic, and $\mathcal{C}_k$ is the class of graphs of bounded treewidth k. Then there is a linear time algorithm deciding if a given graph in $\mathcal{C}_k$ satisfies φ. That is, the problem is FPT.*

Sometimes the application of this result is hidden. A recent example is Grohe's [Gr01c] proof that k-CROSSING NUMBER is quadratic FPT. This filters through proofs of Robertson and Seymour [RS86b, RS95] which says that either a graph G has bounded treewidth or G contains a big grid as a topological minor. In fact this can be achieved in linear time, by Bodlaender's algorithm (Theorem 4.7 below). Thus the proof runs as follows. Grohe proves that there is a linear time algorithm that either gives a certificate that the crossing number is too big, or find a *flat* embedding of a big grid into the graph, or demonstrates that the graph has low treewidth. If the graph has bounded treewidth, in which case we can apply Courcelle's theorem. If not, and the crossing number is $\leq k$ then Grohe argues that we can not only find a grid but remove part of the grid and apply the process recursively.

Interestingly, it is in some sense hard to find properties of graphs that correspond to NP complete problems which are not MSO. One example is *Bandwidth* which is $W[t]$-hard for all t even for trees [BFH94]. Also in terms of MSO, bounded treewidth is the boundary of tractability.

Theorem 4.6 (Seese [Se91]). *Suppose that $\mathcal{C}$ is any family of graphs with a* decidable *monadic second order (MS_2) theory. Then there is a number n such that for all $G \in \mathcal{F}$, the treewidth of G is less than n.*

Finally treewdith is of interest to us also because its recognition, despite being NP-complete, is FPT. The following is the best deterministic algorithm for treewdith, at least theoretically, improving earlier work beginning with Robertson and Seymour's original algorithm.

Theorem 4.7 (Bodlaender [Bod93, Bod96]). *There is a linear time FPT algorithm deciding if a graph has treewidth k.*

The algorithms and results above need to be taken with a grain of salt, and we will take up this issue when we look at implementations and heuristics in Section 6. The notion of treewidth and the related notion of branchwidth have been generalized to matroids by Geelen and Whittle such as [GW03]. Here the idea is that a representable matroid represented by $[v_1, \ldots, v_n]$ is of low branch width if the intersections of the subspaces generated by subsets of the columns has relatively low dimension. That is we thing of the dimension as being like the cardinality of a separation. Additionally, things like implementable algorithms for matroids have been developed and things like Courcelle's theorem proven such as Hlineny [Hl02a, Hl03].

Frick and Grohe [FG02a] identified a property of graphs like treewidth which makes *first order* properties tractable. The *d-neighborhood* of a graph vertex v in a graph G is the induced subgraph of distance d from v. We denote this by $d_G(v)$.

Definition 4.8 (Frick and Grohe [FrG02]). *A class of graphs $\mathcal{C}$ is said to have bounded local treewidth iff there is a function f such that for all $G \in \mathcal{C}$, and all $v \in G$, and all d, the treewidth of $d_G(v)$ is bounded by $f(d)$.*

Classes of graphs with bounded local treewidth include bounded genus (e.g. planar), bounded degree, and those that exclude a minor. The following again generalizes a whole suite of *ad hoc* results.

Theorem 4.9 (Frick and Grohe [FrG02]). *If $\mathcal{C}$ is a class of graphs of bounded local treewidth, and φ is a first order sentence, then for all $k \geq 1$, there is a $\mathcal{O}(n^{1+\frac{1}{k}})$ time FPT algorithm to decide φ for members of $\mathcal{C}$.*

An example of a NP complete property which is first order when parameterized is INDEPENDENT SET.

4.3 Phylogeny

4.3.1 General Phylogenics

One area where the methods of practical parametric complexity have penetrated to the extent they are regarded as relative mainstream is the area of combinatorial computational biology, particularly the area of phylogenic trees. I will look at *one* example, but there are many, particularly from Hallett's group at McGill, Mike Steel's work, and the group at ETH. This example is drawn from recent material, concerning historical linguistics, presented by Tandy Warnow at the annual NZMRI meeting in New Plymouth in New Zealand. (See `www.mcs.vuw.ac.nz/~mathmeet`). I thank Tandy and Luay Nakhleh for making their slides available, since I think that this material should be more widely advertised, particularly to groups such as this.

The underlying problem is similar to one encountered in biology. The *Perfect Phylogeny Problem* is to determine

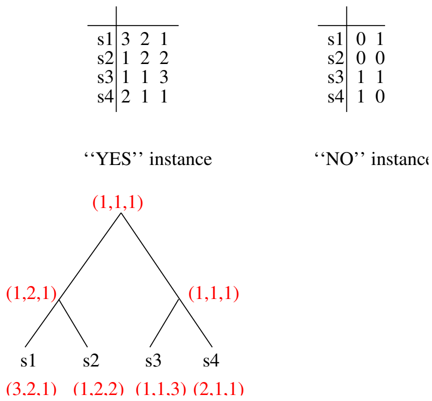

Figure 2. Perfect Phylogeny

whether a given set $S \subseteq \mathbb{Z}^k$ of n "taxa" (classifications of objects, which are *characters* in certain states) has a tree T with the following properties:

(C1) T is leaf-labeled by S, and

(C2) Each internal node v of T can be labeled by a vector in $\mathbb{Z}^k$ such that for every i, $1 \leq i \leq k$, and every $j \in \mathbb{Z}$, the set of all $u \in V(T)$ such that $u_i = j$ induces a subtree of T.

The tree T, if it exists, is called a *perfect phylogeny* (PP) for $\mathcal{S}$ and the set of characters $\mathcal{C}$ is said to be *compatible*.

Figure 2 below gives an example.

An example of taxa that give a yes instance would be $\mathcal{S} = \{(3,2,1),(1,2,2),(1,1,3),(2,1,1)\}$ which are compatible as leaves if you label in the given order. The internal nodes would be labeled $(1,2,1)$ above $(3,2,1),(1,2,2)$ and $(1,1,1)$ above $(1,1,3),(2,1,1)$ with the root labeled $(1,1,1)$. On the other hand, the set $\mathcal{T} = \{(0,1),(0,0),(1,1),(1,0)\}$ has no compatible leaf order. See Figures 2 and 3.

Recall that a graph is is *chordal* or *triangulated* if it contains no induced cycles of length four or more. The relevance of this concept to our studies is the old fact that *Every triangulated graph is the intersection graph of subtrees of a tree.* In fact we are more interested in (properly) colored graphs. A properly colored graph $G = (V, E)$ with coloring $c : V \to \mathbb{Z}$ can be *c-triangulated* if there exists a chordal graph $G' = (V, E')$ where $E \subset E'$ and c is proper on G'.

We can associate a canonical graph with the taxa. Think of each row as a taxon and each column as a character. Then we define avertex for each state of each character. For example $\{s_1 = (3,2,1), s_2 = (1,2,2), s_3 = (1,1,3), s_4 = (2,1,1)\}$ has 3 characters the first having 3 states, the second 2, and the third 3, giving a total of 8 states. One can associate a set of taxa with each state. For instance, if $\alpha_{i,j}$ represents state i of character (column) j, then the set corresponding to $\alpha 1, 1$ would be $\{s_2, s_3\}$. The we form the *character intersection graph* by putting an edge between $\alpha_{i,j}$ and $\alpha_{i',j'}$ if the corresponding sets have nonempty intersection. The relevance of this is the following fundamental theorem.

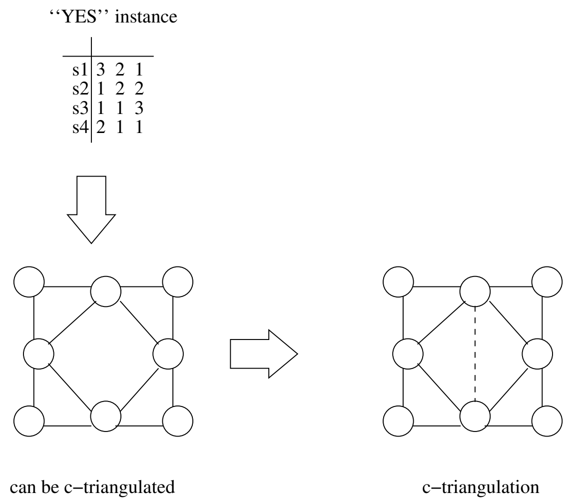

Figure 3. Using Bunemann's Theorem

Theorem 4.10 (Buneman's Theorem). *A set of characters is compatible on n species iff the associated character state intersection graph G can be c-triangulated.*

See Figure 3 for a spirit of the proof. The PP and TRIANGULATED COLORED GRAPH (TCG) problems are equivalent and NP-Complete. (Kannan and Warnow [KW90] and Steel [St92], also F. McMorris, T. Warnow, and T. Wimer [McMWW93]). Additionally they are parametrically hard by Bodlaender, Fellows and Hallett [BFH94].

The 2-character case of the perfect phylogeny problem is solvable in polynomial time. For triangulating colored graphs, McMorris, Warnow and Wimer [McMWW93] gave an $\mathcal{O}((n + m(k-2))^{k+1})$ algorithm, where the graph has n vertices, m edges, and k colors. The corresponding algorithm for PP runs in $\mathcal{O}(r^{k+1}k^{k+1} + nk^2)$ time, where n taxa are defined by k characters each having r states. It is impossible to get rid of the k in the exponent by the W-hardness result of Bodlaender, Fellows and Hallett [BFH94]. Nevertheless, parametric results are possible, as are heuristics. For perfect phylogeny, Gusfield gave an $\mathcal{O}(nk)$ algorithm for $r = 2$, the *binary* character case, Dress and Steel devised an $\mathcal{O}(nk^2)$ algorithm for $r \leq 3$, Kannan and Warnow gave an $\mathcal{O}(n^2k)$ algorithm for $r \leq 4$. Agarwala and Fernandez-Baca gave an $\mathcal{O}(2^{3r}(nk^3 + k^4))$ algorithm for any fixed r, and finally Kannan and Warnow improved it and gave a $\mathcal{O}(2^{2r}nk^2)$ algorithm. (References available on request.)

As we can see, this is just the tip of the iceberg. I have not even mentioned other combinatorial problems related to, for instance, breakpoint phylogenies (see Blanchette, Bourque and Sankoff [BBS97], Coster *et al.* [CJMRWWW00], B. Moret *et al.* [MWBWY01], and Fellows [Fe03]), or combinatorial sequence alignment (see, e.g. Bodlaender *et al.* [BDFHW95]). An excellent recent "real" FPT analysis for breakpoints can be found in Gramm and Niedermeier [GN02]. There are myriads of natural parameters in computational biology.

4.3.2 Applications to Historical Linguistics

We finish this section with a study in historical linguistics which is work of Ringe, Nakhleh, Taylor and Warnow. Some of this can be found in Nakhleh, Ringe, and Warnow [NRW03] and Warnow, Ringe and Taylor [WRT95]. Phylogenies of languages have as their leaves living (and maybe dead) languages such as English, German, French, etc. The "holy grail" of this area is to attempt to figure out the evolution of the languages, perhaps from an original source "proto-Indo-European". The language is represented, for instance, as a huge vector of many words and characters. The characters might be phonological (sound based), lexical (word based), or morphological (grammatical features). Historical linguists have made huge catalogs of such data, based on many techniques in the area. It is not important for our purposes here, save to say that they also provide many sanity checks for the results.

This study looks at the evolution of words through sound changes. A key concept in this area is the notion of a cognate class. Two words w_1 and w_2 are in the same cognate class if they evolved from the same word through sound changes. For example, French "champ" and Italian "champo" are both descendants of Latin "campus": the two words belong to the same cognate class whereas Spanish "mucho" and English "much" are not in the same cognate class. Again a lot is known about this through the work of historical linguists. Borrowing ideas from biology, the Ringe-Warnow Model (1993) of language evolution has a phylogenic tree as its paradigm. The nodes of the tree which contain elements of the same cognate class should form a rooted connected subgraph of the true tree. Hence the model is known as the CHARACTER COMPATIBILITY or another form of PERFECT PHYLOGENY. Ringe and Warnow postulated that all properly encoded characters for the Indo-European languages should be compatible on the true tree, if such a tree existed, would be a perfect phylogeny.

The dataset consisted of 24 languages. (*Anatolian:* Hittite (HI), Luvian (LU), Lycian (LY), *Tocharian:* Tocharian A (TA), Tocharian B (TB), *Celtic:* Old Irish (OI), Welsh (WE), *Italic:* Latin (LA), Oscan (OS), Umbrian (UM), *Germanic:* Gothic (GO), Old Norse (ON), Old English (OE), Old High German (OG), *Albanian*, *Armenian*, *Greek*, *Indo-*

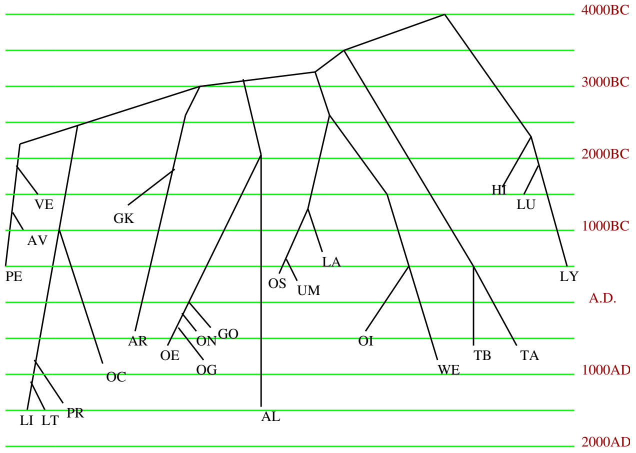

Figure 4. The Indo-European Tree

Iranian: Vedic Sanskrit (VE), Avestan (AV), Old Persian (PE), *Balto-Slavic:* Old Church Slavonic (OC), Old Prussian (PR), Lithuanian (LI), Latvian (LT)) The analysis used 22 phonological characters, 15 morphological characters, and 333 lexical characters, and the total number of working characters was 390. The tree in Figure 4 was obtained by heuristic methods for solving the MAXIMUM COMPATIBILITY problem on the described data.

The problem is Germanic. With Germanic: 372 characters are compatible on the tree and 18 are incompatible Without Germanic: 384 characters are compatible on the tree and 6 are incompatible. The conclusion was that *analysis of the IE dataset revealed that no perfect phylogeny for that dataset existed. Thus, whilst the the basic approach was good, new ideas were needed.* Again we can look towards biology for inspiration. We know so-called horizontal gene transfer happen. This is even easier to see with linguistics: Neighboring countries will affect each other and continue to interact.

Nakhleh *et al.* [NRW03] suggest that the tree model be replaced by *Phylogenetic Networks*. A phylogenetic network on a set L of languages is a rooted directed graph, or "digraph", $N = (V, E)$ with the following properties:
$V = L \cup I$, where I denotes added nodes which represent ancestral languages and $E = E_T \cup E_N$, where E_T are the edges of the tree $T = (V, E_T)$, where $L \subset V$ are the leaves of T, and E_N are the "non-tree" edges.
The edges in E_T are oriented from parent to child, and hence T is a rooted directed tree.
The edges in E_N are bidirectional N is "weakly acyclic", i.e., if N contains directed cycles, then those cycles contain only edges in E_N. (More details can be found in Nakhleh *et al.* [NRW03].

Figure 5 is a simple Phylogenic Network. Naturally, a phylogenetic network gives rise to a collection of trees

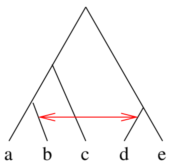

Figure 5. Phylogenic Network

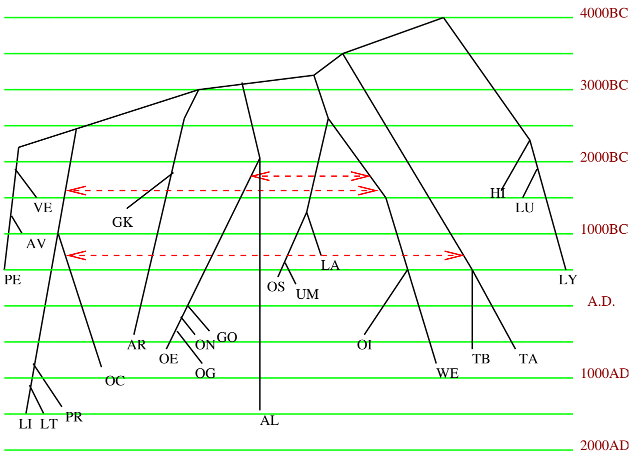

Figure 6. The Indo-European Network

that represent the evolutionary histories of the various characters. A character is compatible on a network N if it is compatible on at least one of the trees induced by N. Then is a short step to define PERFECT PHYLOGENIC NETWORK(PPN) in place of PP. We can test the compatibility of a character c on a tree T with n leaves in $\mathcal{O}(n)$ time, and hence in $\mathcal{O}(n3^B)$ on a network with B non-tree edges. The idea is to use B as a parameter. There are a number of results on the complexity of these analyses and the reader should turn to the papers of Ringe, Warnow, Nakhleh etc, for more on this topic. The authors use the heuristic Minimum Increment to PPN (MIPPN) First they obtained a tree T on L (e.g., using Maximum Compatibility), then add a minimum number of non-tree edges to T to obtain a PPN. They showed a number of parametric complexity results, including MIPPN is polynomial for a fixed number of non-tree edges ($\mathcal{O}(k2^{4B}n^{2B+1}$) where B is the number of non-tree edges, $n = |L|$, and $k = |C|$.).

Using this method, this particular story had a happy ending. A relatively small value for B was all that was necessary in the Indo-European case. Using this method, they obtained the network represented by Figure 6.

There is clearly a lot of new applications available for these ideas, and a lot of work still to be done.

4.4 Impractical, sometimes useful, but very attractive

There are some methods available such as *hashing*(at least hashing associated with color coding below), and *minors* which are mathematically important, but are presently wildly impractical, yet remain useful *since they can be used, often easily to quickly establish theoretical tractability.* We'll look at hashing when we look at the randomized version, called *color coding* later.

In a long series of difficult papers, collectively entitled "Graph Minors," almost all appearing in *Journal of Combinatorial Theory B*, Robertson and Seymour have revolutionized graph theory towards topological graph theory. Treewidth was but one by-product of their work. We say that a partial ordering $\preceq$ on graphs is a *well partial order* iff given any infinite sequence $G_1, G_2, \ldots$ of graphs, there is some $i < j$ such that $G_i \preceq G_j$.

The partial ordering of relevance to us is the *minor* order where we have $G \prec_M H$ iff G can be gotten from H by edge deletions and contractions.

Theorem 4.11 (Robertson and Seymour). *Finite graphs are well partial ordered by the minor relation.*

How can this be used algorithmically? Well, Robertson and Seymour also proved that the minor relation $\prec_M$ is cubic time FPT. Here we parameterize by fixing a graph G as the parameter. For a fixed G, we ask is $G \prec_M H$?. For a fixed G, this question is $\mathcal{O}(|H|^3)$. The constants are astronomical with iterated stacks of powers of 2 in the exponent. There seems some evidence that this is necessary. But the point is that this means that *any* minor closed class has a theoretical cubic time recognition algorithm. For example, consider the question of whether a graph has genus k? Clearly genus k graphs are closed under the minor relation. This means that there are a finite number of graphs, called an *obstruction set* $G_1, \ldots, G_d$ such that H has genus k iff for all $1 \leq i \leq d$, $G_i \not\preceq_M H$. (In the planar case this is Kuratowski's Theorem, and the graphs are K_5 and $K_{3,3}$.) Testing for genus k amounts to d minor tests, which makes it $\mathcal{O}(|H|^3)$. We remark that the algorithm has been improved to linear time by Mohar [Mo99]. Fellows [Fe89] has a survey of old applications of this type. Many recent applications also apply to classes that *exclude a particular graph*, which has deep consequences. For instance, Robertson and Seymour [**?**, RS95] that excluding a planar graph, or even a grid makes the class have bounded treewidth and hence have a linear time recognition algorithm. We remark that the Robertson Seymour methods are inherently nonconstructive since there is no algorithm to generate the obstruction set in general. But again this situation is akin to that of regular languages where the data is presented in such a way to be useful. Some progress has been done towards understanding what information is needed for such effective generation of the obstruction sets. See, for example, [CDDFL00, CDF97].

5 Connections with Classical Complexity

5.1 PTAS's

In the first section, we were introduced to some rather impractical PTAS's. They were from STOC, SODA etc. They are in fact only a small sample of such. Here area couple of others. The PTAS for the UNBOUNDED BATCH SCHEDULING problem due to Deng, Feng, Zhang and Zhu [DFZZ01] has a running time of $O(n^{5\log_{1+\epsilon}(1+(1/\epsilon))})$ which for a 20% error we have an $O(n^{50})$ polynomial-time algorithm. Wu *et al.* [WLBCRT98] gave a $\mathcal{O}(n^{2\lceil\frac{2}{\epsilon}\rceil-2})$ PTAS for MINIMUM COST ROUTING SPANNING TREE giving an algorithm running in time $\mathcal{O}(n^8)$ for a 20% error. Similarly, the PTAS for TWO-VEHICLE SCHEDULING ON A PATH due to Karuno and Nagamochi [KN01] has a running time of $O(n^{8(1+(2/\epsilon))})$; thus $O(n^{88})$ for a 20% error. Finally, the PTAS for the CLASS-CONSTRAINED PACKING PROBLEM due to Shachnai and Tamir [ST00] has a running time (for 3 colors) of $O(n^{64/\epsilon+(log(1/\epsilon)/\epsilon^8)})$; thus for a 20% error (for 3 colors) we have a running time of $O(n^{1021570})$.

As we have observed, the problem is mainly with the $\frac{1}{\epsilon}$ in the exponent. We have the following.

Definition 5.1. *An optimization problem* Π *has an* efficient P-time approximation scheme e *(EPTAS) if it can be approximated to a goodness of* $(1+\epsilon)$ *of optimal in time* $f(k)n^c$ *where c is a constant t and* $k = 1/\epsilon$*.*

Arora gave an EPTAS for the EUCLIDEAN TSP [Ar97], but for all of the other PTAS's mentioned above, the possibility of such an improvement remains open.

But we have a strategy: Prove the problem is $W[1]$-hard parameterized by $k = \frac{1}{\epsilon}$ and you prove that the problem has no EPTAS, assuming that $W[1] \neq FPT$..

There are a number of nice case studies which support the thesis that this methodology might well bear fruit. Many can be gotten by applying a nice result connecting EPTAS and $W[1]$ first articulated by Bazgan:

Theorem 5.2 (Bazgan [Baz95], also Cai and Chen [CC97]). *Suppose that* Π_{opt} *is an optimization problem, and that* Π_{param} *is the corresponding parameterized problem, where the parameter is the value of an optimal solution. Then* Π_{param} *is fixed-parameter tractable if* Π_{opt} *has an EPTAS.*

(There is also other nice related work by Cesati and Trevisan [CT97].) Here is one recent application of Bazgan's Theorem taken from Fellows, Cai, Juedes and Rosamond [CFJR01]. In a well-known paper, Khanna and Motwani introduced three planar logic problems towards an explanation of PTAS-approximability. Their suggestion is that "hidden planar structure" in the logic of an optimization problem is what allows PTASs to be developed [KM96]. One of their core problems was the following.

PLANAR TMIN

Input: A collection of Boolean formulas in sum-of-products form, with all literals positive, where the associated bipartite graph is planar (this graph has a vertex for each formula and a vertex for each variable, and an edge between two such vertices if the variable occurs in the formula).

Output: A truth assignment of minimum weight (i.e., a minimum number of variables set to *true*) that satisfies all the formulas.

Theorem 5.3 (Fellows, Cai, Juedes and Rosamond [CFJR01]). PLANAR TMIN *is hard for* $W[1]$ *and therefore does not have an EPTAS unless* $FPT = W[1]$.

The method of proof is to show that CLIQUE is parameterized reducible to PLANAR TMIN with the parameter being the weight of a truth assignment. Since CLIQUE is W[1]-complete, it will follow that the parameterized form of PLANAR TMIN is W[1]-hard. This is a relatively straightforward reduction, the details being found in [CFJR01] and Fellows [Fe03]. Fellows *et al.* [CFJR01] also show that the other two core problems of Khanna and Motwani [KM96] are also $W[1]$ hard and hence have no EPTAS's. It seems to us that there is a pretty major project waiting here to understand when problems such as those in [ACGKMP99], can have real EPTAS's rather than just PTAS's which have unrealistic running times.

5.2 Other Connections

The reader might well ask whether one can see the W-hierarchy to say something else about classical notions aside from PTAS's. Parameterized complexity can also be used to address the probable *practical* intractability of other problems which are likely not NP complete. One such example was VAPNIK CHERVONENKIS DIMENSION which was shown to be in the time class DTIME($n^{\log n}$) by Papadimitriou and Yannakakis [PY93], and hence likely not NP complete unless NP$\subseteq$DTIME($n^{\log n}$). another classic example is UNIT LENGTH PRECEDENCE CONSTRAINED SCHEDULING which was shown $W[1]$-hard in [BF95], and whose unparameterized case is still OPEN. But this means that the general case almost certainly not in P. One might be able to apply this method to show important open problems are likely not in P such as GRAPH ISOMORPHISM, or COMPOSITE NUMBER by parameterizing (e.g.) the first by valence or treewidth.

A final application of this sort was given by Alehknovich and Razborov [AR01]. They were looking at proof systems studying the basic question of what can be achieved in a give proof system. In particular the studied proof systems P called in [BPR01] *axiomatizable* meaning that there is a deterministic algorithm A which, when give a tautology τ returns its shortest proof in time polynomial in the size of the shortest P-proof of τ. It is known that it is unlikely that tree-like resolution can be proved not axiomatizable using the assumption that $P \neq NP$, because such a proof would imply quasi-polynomial time algorithms for NP. Again we want a complexity theoretical hypothesis sensitive to the needs of PTIME.

Theorem 5.4 (Alekhnovich and Razborov [AR01]). *Neither resolution not tree-like resolution is axiomatizable unless $W[P]$ is randomized FPT by a randomized algorithm with one-sided error.*

Alekhnovich and Razborov remark that they were able to use these techniques to relate approximate solution to MONOTONE CIRCUIT SATISFIABILITY to an exact solution without going via PCP.

The reader might also ask does $FPT = W[1]$ imply anything classical? This was explored somewhat by Abrahamson, Downey, and Fellows [ADF95], and discussed in [DF99a]. Here is a sample result. We define the class SUBEXPTIME($f(n)$) to be the set of languages L for which there is a polynomial $p(n)$ such that L is accepted in DTIME($p(n)2^{g(n)}$) for some function g in $o(f(n))$.

Theorem 5.5 (Abrahamson, Downey, Fellows [ADF95]). *Let $NP[a]$ denote NP where at most a nondeterministic moves are allowed. Then $W[P] = FPT$ iff for every P-time function f with $f(n) \geq \log n$, there is a recursive function h such that for every $L \in NP[f(n)]$ $h(L)$ computes machine M which witnesses that $L \in SUBEXPTIME(f(n))$.*

One idea introduced by (the somewhat flawed) Cai and Juedes [CJ01] was to see what kinds of exponents FPT algorithms might have. An illustration of this is the following recent results.

Theorem 5.6 (Dehne, Fellows, and Rosamond [DFR03]). *There is a $\mathcal{O}(n^4+2^{\mathcal{O}(k)}n^{2.5})$ algorithm for* SET SPLITTING *parameterized by the number of sets to be split. However, there can be no FPT algorithm running in time $2^{o(k)}n^c$ unless the satisfiability of n variable 3SAT is solvable in time $2^{o(n)}$*

Another one is due to Cai and Jeudes (see their homepage) for PLANAR DOMINATING SET matching the upper bound of $c^{\sqrt{k}}$ from Alber *et al.* [ABFKN02]. There are some other relationships, but we explore some of these in the next sections.

Perhaps one of the most important paper concerning lower bounds here is the somewhat neglected paper of Impagliazzo, Paturi and Zane [IPZ01].

6 Implementations, Heuristics, and Reality

As we mentioned in the introduction, the methodology of parametric complexity has found its home in the heuristic and applied community to some extent. In a short survey like this, we cannot hope to give more than a sample like the Warnow material above, and really must refer the reader to recent surveys aimed in this direction for more details. These would especially include [Ra97, Nie98, DF99b, DFS99] and we'd strongly recommend that the reader get a copy of Rolf Niedermeier's Habilatationschrift called "An Invitation to Fixed-parameter Algorithms." There have also been a number of implementations and attempts at implementations of this material including [KBFvH03, McC03, Fo03, St00, AGN01].

Later in this section we will address the implementability of the general methods mentioned in earlier sections. We begin by looking at the kinds of techniques that have worked in practice. Generally, for exact algorithms, most of the general methods such as treewidth, local treewidth etc are currently useless; but more on this later.

Downey and Fellows (e.g. [DF99a]) pointed out two simple methods of obtaining FPT algorithms that often yield practical algorithms. One is called the method of bounded search trees. The simplest example of this is applied to k-VERTEX COVER. Given a graph G then start at any edge $e = v_1v_2$. Then any vertex cover must include one of v_1 or v_2. Then begin a tree of depth k branching at v_1 or v_2, and at each branch consider the subgraph of G not covered by v_i; then repeat. This gives a simple $\mathcal{O}(2^k|G|)$ algorithm. The other technique is very powerful and simple, and is called *Kernelization*. Here is an example by Sam Buss again for k-VERTEX COVER. Take G. If G has a vertex of degree $> k$ it must be in any k-vertex cover. Delete such a vertex. Apply this to the resulting graph. If there are more than k such high degree vertices then reject. Otherwise take the small degree graph that results, and if it has more than k^2 vertices reject since we can show that a large graph of only small degree vertices cannot have a k-vertex cover. Finally one could use complete search for the k^2 vertices, or even bounded search trees for the small kernel graph giving a $\mathcal{O}(n+2^kk^2)$ algorithm. (A better kernelization for VERTEX COVER is given by a result of Nemhauser and Trotter (size 2k kernel)- see Chen, Kanj and Jia [CKJ01].). More intricate versions of this method use repeated alternation of the techniques, more complicated trees branching on higher degree vertices etc. This is how the current "champion" algorithm was found. This methodology is clearly applicable in any number of applications. These include the work such as [FMcRS01, KR00, AFN01, AFN02, AGN01, Ar00, BR99, CJMRWWW00, CKJ01, CS97, PSta, EL87, GN00]. Even for $W[1]$ hard problems kernelization can be very powerful in practice. See, for example, Weihe [Wei00].

There are many others. One notable example is the one we met earlier CHORDAL GRAPH COMPLETION. Give a graph G can one add $\leq k$ vertices to get a triangulated graph. Yannakakis proved that the general problem is NP-complete. Kaplan, Shamir and Tarjan [KST94] used Kernelization to show that there is a $\mathcal{O}((k^5|V||E| + f(k))$ algorithm for suitably chosen computable f. Leizhen Cai [LeC96] also used the Kernelization method to give a $\mathcal{O}((4^k((k+1)^{-3/2})[|E(G)||V(G)| + |V(G)|^2])$ algorithm. Time might be appropriate to re-examine these for improvements.

Actually in [LeC96], Cai studied a class of problems for which this technique works in general. If Π is a property of graphs, we say that a property Π is a *hereditary property* is given any Π graph G (i.e. graph satisfying Π), if H is an induced subgraph of G then H is a Π graph. The literature is filled with many *graph modification* problems. As Leizhen Cai [LeC96] remarks, in general these problems can be placed in the categories below.

1. The *edge deletion problem* : Find a set of edges of minimum cardinality whose removal results in a Π graph.
2. The *vertex deletion problem* : Find a set of vertices of minimum cardinality whose removal results in a Π graph.
3. The *edge/vertex deletion problem* : 1. and 2. combined.
4. The *edge addition problem* : find a set of new edges of minimum cardinality whose addition results in a Π graph.

It is known that 1, 2 and 3 are NP-hard for any nontrivial hereditary property. In [LeC96], Leizhen Cai considered the following general problem:

$\Pi_{i,j,k}$ GRAPH MODIFICATION PROBLEM.
Input : A graph G.
Parameters : Nonnegative integers i, j, k.
Question : Can we delete at most i vertices, j edges, and add at most k edges and get a Π graph?

We will say that a property Π has a (finite) *forbidden set characterization* iff there exist a (finite) set $\mathcal{F}$ of graphs such that G is a Π graph iff G does not contain a member of $\mathcal{F}$ as an induced subgraph.

Clearly if $\mathcal{F}$ is a finite set of size N characterizing Π then there is a a more or less trivial $\mathcal{O}(|G|^{i+2j+2k+N})$-time recognition algorithm for the $\Pi_{i,j,k}$ GRAPH MODIFICATION PROBLEM, for any fixed i, j, k. However, Cai used essentially the problem kernel method to prove the following.

Theorem 6.1 (Leizhen Cai [LeC96]). *Suppose that* Π *is any property with a finite forbidden set characterization. Then the* $\Pi_{i,j,k}$ GRAPH MODIFICATION PROBLEM *is* FPT *in time* $O(N^{i+2j+2k}|G|^{N+1})$ *where* N *denotes the maximum size of the vertex set of any graph in the forbidden set* $\mathcal{F}$.

We remark that the parameterized complexity of the bipartization problem (delete edges or vertices to make a graph bipartite) is an extremely 'important open problem (see, e.g., [MR99]), with applications in computational biology.

Now we turn to issues of practicality of the *general* theorems on tractability we have seen in previous sections. Let's begin with the methods based on treewidth.

There are two issues. The *first* is running things like automata on tree decompositions to apply things like Courcelle's Theorem. The good news is that *in practice* this is not too bad. But in theory it is very bad indeed. The general setting is to look at model checking for monadic second order logic: given a L-sentence and a class $\mathcal{C}$ seeing whether the sentence holds in a given structure in $\mathcal{C}$. Similarly for first order logic.

Theorem 6.2 (Frick and Grohe [FrG02]). (i) *If* $NP \neq P$*, then the model checking problem for finite trees (in fact words) is not solvable in time* $f(k)p(n)$*, where* n *is the size of the word and* k *the sentence, for any polynomial* p *and* elementary *computable function* f.
(ii) *If* $W[1] \neq FPT$*, then the model checking for first order logic on structures of degree 2 and bounded degree* $d \geq 3$ *is not solvable in time* $2^{2^{o(k)}}p(n)$ *(degree 2) and* $2^{2^{2^{o(k)}}}p(n)$ *for any polynomial* p *unless.*

The *second* issue is *how* to get the tree decomposition in the first place. Now one point if view is to look at graphs that are naturally given to us with fairly small treewidth. For instance, think of a train network in a country. It is surely fairly treelike as it is given to you. The other point of view concerns what can be done with graphs given in some more or less random fashion and we want to apply something like Bodlaender's Theorem. Unfortunately, for a fixed k, Bodlaender's Theorem actually runs in time $\mathcal{O}(2^{32k^2}|G|)$. Even worse, the recursive structure of the algorithm means that not only is it theoretically bad but it falls over for any attempted implementation[4] This has lead to some attempts at heuristics for treewidth. One example of such computational experiments is in Fouhy [Fo03] and Koster *et al.* [KBFvH03]. For graphs of size less than 140 results seem to be encouraging.

There are other natural candidates for possible general techniques for FPT algorithms. One such is relevant to *randomized* FPT algorithms. This is the *color coding* technique of Alon, Yuster and Zwick [AYZ94]. To the author's knowledge this technique has not been implemented. Here is a

[4]There is an issue here. Algorithms can run in practice much faster than they "should". Sometimes the reason is the parametric nature, such as the number of "lets" in some structured program. Sometimes there are other reasons, a really nice example being Abdulla and Nylen's methods (e.g. [AN00]) using methods such as well-quasi-ordering and Better-quasi-ordering theory to verify infinite-state systems. Such methods *should* involve constants like Ackermann's function, and so give rise to running times like the life of the universe, but work well in practice, running in microseconds.

brief description of how the method works. We will apply the problem to k-PATH which seeks to find a (simple) path of k vertices in G. What we do is to *randomly* color the whole graph with k colors, and look for a *colorful* solution, namely one with k vertices of one of each color.

The two keys to this idea are
(i) we can check for colorful paths quickly.
(ii) if there is a simple path then the probability that it will have k colors for a random coloring is $\frac{k!}{k^k}$ which is bounded by e^{-k}.

Then, given (i) and (ii), we only need repeat process enough to fast probabilistic algorithm. We prove (i) by using dynamic programming: simply add a vertex v_0 with color 0, connect to those of color 1, then generate the colorful paths of length i starting from v_0 inductively, rather like Dijkstra's algorithm, the running time being $\mathcal{O}(k2^k|E|)$.

Theorem 6.3 (Alon, Yuster and Zwick [AYZ94]). *k-PATH can be solved in expected time $2^{\mathcal{O}(k)}|E|$.*

Alon, Yuster and Zwick demonstrated that this technique could be applied to a number of problems of the form asking "is G' a subgraph of G?" Note that the method does not allow for things like k-CLIQUE to be shown randomized FPT because (i) above *fails*. The important part of the dynamic programming method was that a path was represented by its beginning v_0 and some vertex v_i, and to extend the path only needed *local knowledge*; namely the colors used so far and v_i. This fails for CLIQUE, and would need $\binom{n}{i}$ at step i in the clique case.

Color coding would also seem to have a lot of unexplored uses, perhaps in relation to randomized treewidth algorithms. I think there is huge potential here. Certainly there is no general theory of randomized FPT, and this waits for development.

Finally we should mention that we can get at least theoretical FPT algorithms by derandomizing color coding algorithms. A *k-perfect family of hash functions* is a family $\mathcal{F}$ of functions (colorings) taking $[n] = \{1, \dots n\}$ onto $[k]$, such that for all $S \subseteq [n]$ of size k there is a $f \in \mathcal{F}$ whose restriction to is bijective (colorful). It is known that k-perfect families of $2^{\mathcal{O}(k)} \log n$ linear time hash functions. This gives a deterministic $2^{\mathcal{O}(k)}|E| \log |V|$ algorithm for k-PATH. More such applications can be found in Downey and Fellows [DF99a]. The $\mathcal{O}(k)$ in the exponent hides evil, and the derandomization method at present seems far from practical.

One final technique we have not discussed in [DF99a], is the use of INTEGER PROGRAMMING in the design of FP algorithms. This is discussed in Niedermeier [Nie02].

Theorem 6.4 (Lenstra [Le83]). *The integer programming feasibility problem can be solved with $\mathcal{O}(p^{\frac{9p}{2}}L)$ arithmetical operations in $\mathbb{Z}$ of $\mathcal{O}(p^{2p}L)$ bits in size, where p is the number of variables, and L the number of bits of the input.*

Niedermeier [Nie02] gave one example of the use of this method for establishing parametric tractability. He showed that the following problem is FPT.

CLOSEST STRING (parameterized by the number of strings and length)
Input: k strings $s_1, \dots s_k$ over an alphabet Σ each having length L, and a nonnegative integer d.
Parameter: k, L, d.
Question: is there a string s of distance $\leq d$ from s_i for all i?

We remark that the method's practicality is far from explored. We also refer the reader to Gramm, Niedermeier and Rossmanith [GNR01].

7 Other areas of application

There are a host of new arenas of applications of these ideas waiting. Here is one : *Online Algorithms*. An online structure such as a graph is given vertex by vertex, $\{v_1, v_2, \dots\}$ so that when v_i is given we are told the v_j for $j < i$ incident with v_i. There are many variations, but an online algorithm for a graph property Π, takes the online presentation of the graph and produces a Π-set in step i for vertex i, so that the algorithm is a suitably chosen function f acting on vertices presented one at a time. For instance, online coloring must color the graph vertex by vertex. Thus we must color the object quickly with only local information. This is an important area of algorithmics and is a really good candiate for parametric analysis.

The point here is that in an online algorithm, we cannot wait till we see the whole structure to assign a color. For example, it is easy to show that there is an online presentation of a k-colorable graph such that any online algorithm needs $\Omega(2^k)$ colors. Indeed there is a family of planar graphs needing arbitarily large numbers of colors to color online. The online community is very concerned with scheduling, etc, and look at what is called *performance ratios* where we look at the non-online value vs the online one. One of the basic online algorithms is *first fit* which greedily assigns colors as best it can.

We are interested in how presentations might affect the performance of online algorithms. Irani [Ir94] looked at what are called *k-inductive* graphs. $G = (V, E)$ is k inductive iff there is *some* ordering $v_1, v_2, \dots$ of the vertices such that $|\{v_j : j > i \wedge v_i v_j \in E\}| \leq k$. For instance, any planar graph has a vertex of degree 5, and hence all planar graphs are 5-inductive. Similarly any bounded degree graph is d-inductive for some d. It is a nice exercise to show that any graph of bounded treewidth is d-inductive for some d.

Theorem 7.1 (Irani [Ir94]). *First fit online colors any online presentation of a d-inductive graph in at most $d \log n$ colors. This bound is sharp.*

The notion of k-inductive is interesting in its own right. It might well have parametric applications elsewhere. For instance, Martin Grohe asks if the work on first order model checking can be extended to k-inductive graphs. What about online model checking etc? It would seem rather fruitful to pursure this area with an eye towards topological graph theory. Downey, Fouhy and McCartin [DFMcC03] have some recent work on this, and especially looking at how the presentation of the data affects the performance of the algorithms. For instance a pathwidth k graph presented by its path decomposition can be online colored with $k + 1$ colors. However, this bound fails if the graphs is presented in some random fashion. If a graph of pathwidth k is presented in any fashion it can still be colored by $3k + 1$ colors by some online algorithm and by 25.72(k+1) colors using first-fit. (Kierstead and Qin see [KQ95]). Slusarek [Sl93] has proven a 4.4(k+1) lower bound for first-fit. Computational experiments suggest that $2k + 2$ is a typical bound which we would expect. There seems little work concerning the effect of differing types of presentations.

Another more or less unexplored is n^k crypto using reasonable "no-ftp" security guarantees. While the cryptosystems might be in PTIME, $W[1] \neq FPT$ would be a reasonable working guarantee. The only work I am aware of here is Fellows and Koblitz [FK93].

8 Structural Issues

Aside from understanding the issues relating parametric complexity with classical complexity there are many poorly understood aspects of parametric complexity, waiting for eager graduate students. We can easily form analogs of many of the classical complexity notions. For instance, Flum and Grohe [FG02a], and Cai, Chen, Downey, and Fellows [CCDF96] have looked at parameterized LOGSPACE, where VERTEX COVER turns out to be in parameterized LOGSPACE, and as Flum and Grohe observed a wide class of problems for graphs of bounded degree also belong to this class. Another area looked at is parametric counting by Flum and Grohe [FG02b], Arvind [Ar00], and McCartin [McC02], who looked at $\#W[t]$ for instance, proving basic results. For instance, as Ventakesh Raman observed counting k-vertex covers is FPT. Similarly, counting in bounded treewidth is also FPT. Grohe showed that the the parameterized problem of counting k-cycles in a graph is $\#W[1]$ complete, an analog of Valiant's Theorem. The methods are quite different. Abrahamson, Downey, and Fellows [ADF95], and Chen, Flum and Grohe [CFG03, FG02a] have also looked at alternation, with two distinct hierarchies the $A[t]$ hierarchy and the $AW[t]$ hierarchy.

Lots of analog questions are wide open. What about a PCP theorem? What about Toda's Theorem? Can the switching lemma be applied? What about parameterized average case complexity?

The basic parameterized theory itself generates structural questions of independent interest. For instance, it is conjectured that $W[t] \neq W[t + 1]$, but it is not known if e.g. $W[1] = FPT$ implies *anything* about the rest of the W-hierarchy. Oracle results have been explored in [DF93, DF99a], but they are somewhat unsatisfactory. Even an analog of Ladner's theorem (that the polynomial time degrees are dense, or that there are intermediate problems if $P \neq NP$) is not fully answered (see [DF99a]). The parameterized reducibilities are much more complex than the standard ones. There are a lot of questions like "does BANDWIDTH $\in W[1]$ imply *anything*?"

One recent interesting development has been to challenge the class $W[1]$ as the basic "infeasible" class. In Downey *et al.* [DEFPR03], the authors introduced a new idea. That is they asked that we parameterize the size of the input, rather than the aspect of the output we are interested in. This forms what they call the "mini-classes". For example MINI-VERTEX COVER is the problem that takes as input a graph of size $k \log n$, and asks what is the minimum vertex cover of the graph. The basic hardness kernel is the problem MINI-CIRCUIT SATISFIABILITY. The gives a new intermediate class

$$FPT \subseteq M[1] \subseteq W[1].$$

The mini-classes include *Mini-Vertex Cover*, MINI-DOMINATING SET, MINI-3SAT, MINI-SET SPLITTING, MINI-NOT ALL EQUAL 3SAT, MINI-INDEPENDENT SET, etc. The beauty of the mini-classes is that the reductions look more like "regular" ones, provided that they preserve $k \log$'s. The non-optimization theorem, Theorem 5.6, for SET COVER of Dehne, Fellows and Rosamond follows from material related to this.

Theorem 8.1 (Downey, Estivill-Castro, Fellows, Prieto-Rodriguez and Rosamond [DEFPR03]). $FPT = M[1]$ *iff* n*-variable* $3SAT$ *can be solved in time* $2^{o(n)}$.

The method of proof was inspired by Cai and Juedes [CJ01]. Can the hierarchy be extended? Where does MINI-TURING MACHINE ACCEPTANCE fall in all of this? There are many questions of this form. A final result of this ilk is the following.

Theorem 8.2 (Chor, Fellows, Juedes [Fe]). *If* $FPT = M[1]$ *then* k-INDEPENDENT SET *and* k-DOMINATING SET *are in* $DTIME(2^{o(k)})$.

One could well view $W[1]$ hardness as simply meaning that some form of k must stay in the exponent, and does not rule out $n^{k \log \log n}$ or something. These last results say something like if you believe the n variable SAT not in $2^{o(n)}$ then $W[1]$ hardness is serious bad news. By the way, we don't have much evidence that $M[1] \neq W[1]$, but any proof would need something clever. One can show that $M[1] = W[1]$ iff the parameterized problem asking if a graph has vertex cover of sized $k \log n$ is $W[1]$ hard.

References

[AN00] P. Abdulla and A. Nylen, "Better is Better than Well: On Efficient Verification of Infinite-State Systems." *Proc. LICS'2000, 14th IEEE Int. Symp. on Logic in Computer Science.*

[ABFKN02] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks and R. Niedermeier. "Fixed Parameter Algorithms for Dominating Set and Related Problems on Planar Graphs," *Algorithmica*, 33:461–493, 2002.

[ACGKMP99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela and M. Protasi, *Complexity and Approximation*, Springer-Verlag, 1999.

[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.

[AFN01] J. Alber, H. Fernau and R. Niedermeier, "Parameterized Complexity: Exponential Speed-Up for Planar Graph Problems," in: Proceedings of ICALP 2001, Crete, Greece, *Lecture Notes in Computer Science* vol. 2076 (2001), 261-272.

[AFN02] J. Alber, M. Fellows and R. Niedermeier, "Efficient Data Reduction for Dominating Set: A Linear Problem Kernel for the Planar Case," in *Proc. 8th SWAT*, Springer-Verlag LNCS 2368, pp. 150–159, 2002.

[AGN01] J. Alber, J. Gramm and R. Niedermeier, "Faster Exact Algorithms for Hard Problems: A Parameterized Point of View," *Discrete Mathematics* 229 (2001), 3–27.

[AR01] M. Alekhnovich and A. Razborov, "Resolution is Not Automatizable Unless W[P] is Tractable," *Proc. of the 42nd IEEE FOCS,* 2001, 210-219.

[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, pp. 2–12.

[Ar97] S. Arora, "Nearly Linear Time Approximation Schemes for Euclidean TSP and Other Geometric Problems," *Proc. 38th Annual IEEE Symposium on the Foundations of Computing* (FOCS'97), IEEE Press (1997), 554-563.

[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).

[Ar00] V. Arvind, "On the Parameterized Complexity of Counting Problems," *Workshop on Parameterized Complexity*, Madras, India, Dec. 2000.

[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.

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

[BR99] N. Bansal and V. Raman, "Upper Bounds for MaxSat: Further Improved," *Proc. 10th International Symposium on Algorithms and Computation (ISAAC '99)* , Springer-Verlag, *Lecture Notes in Computer Science* 1741 (1999), 247–258.

[BBS97] M. Blanchette, G. Bourque and D. Sankoff, "Breakpoint Phylogenies," in: *Genome Informatics 1997* (S. Miyano and T. Tagaki, eds.), Universal Academy Press, Tokyo, 1997, pp. 25–34.

[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.

[Bod96] H. L. Bodlaender. "A linear time algorithm for finding tree-decompositions of small treewidth," *SIAM Journal on Computing*, 25:1305–1317, 1996.

[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.

[BE97] H. Bodlaender and J. Engelfreit, "Domino Treewidth,"*J. Algorithms*, 24 (1997),94-127.

[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.

[BPR01] M. Bonnet, T. Pitazzi, and R. Raz, "On Interpolation and Axiomatization for Frege Systems," *SIAM J. Comput,* 29 (2000), 1939-1967.

[LeC96] Leizhen Cai, "Fixed-parameter tractability of graph modification problems for hereditary properties," *Information Processing Letters*, 58(4), 171-176, 1996.

[Cai01] Leizhen Cai, "Parameterized Complexity of Vertex Coloring," to appear in *Discrete Applied Math.*

[CS97] Leizhen Cai and B. Schieber, "A Linear Time Algorithm for Computing the Intersection of All Odd Cycles in a Graph," *Discrete Applied Math.* 73 (1997), 27-34.

[CC97] Liming 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.

[CFJR01] Liming Cai, M. Fellows, D. Juedes and F. Rosamond, "Efficient Polynomial-Time Approximation Schemes for Problems on Planar Structures: Upper and Lower Bounds," manuscript, 2001.

[CJ01] L. Cai and D. Jeudes, "Subexponential parameterized algorithms collapse the W-hierarchy," in *ICALP, 2001* LNCS 2076.

[CDDFL00] K. Cattell, M. Dinneen, R. Downey, M. Fellows, and M. Langston, "On Computing Graph Minor Obstruction Sets," *Theor. Comp. Science.* Vol. 233 (2000), 107-127.

[CM77] A. Chandra and P. Merlin,"Optimal Implementation of Conjunctive Queries in Relational Databases," in *Proceedings STOC*, 1977, 77-90.

[CK00] C. Chekuri and S. Khanna, "A PTAS for the Multiple Knapsack Problem," *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms* (SODA 2000), pp. 213-222.

[CKJ01] J. Chen, I.A. Kanj and W. Jia, "Vertex Cover: Further Observations and Further Improvements," *Journal of Algorithms*, 41:280–301, 2001.

[CFG03] Y. Chen, J. Flum, and M. Grohe, "Bounded Nondeterminism and Alternation in Parameterized Complexity Theory," to appear.

[Co87] B. Courcelle, "Recognizability and Second-Order Definability for Sets of Finite Graphs," Technical Report I-8634, Universite de Bordeaux, 1987.

[CDF97] B. Courcelle, R. Downey, and M. Fellows "A Note on the Computability of Graph Minor Obstruction Sets for Monadic Second Order Ideals," *Journal of Universal Computer Science,* Vol. 3 (1997), 1194-1198

[CM99] J. Chen and A. Miranda, "A Polynomial-Time Approximation Scheme for General Multiprocessor Scheduling," *Proc. ACM Symposium on Theory of Computing* (STOC '99), ACM Press (1999), 418–427.

[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: Volume 1*, IEEE Press, Los Alamitos, CA (1995), 880-885.

[CJMRWWW00] M. Cosner, R. Jansen, B. Moret, L. Raubeson, L. Wang, T. Warnow and S. Wyman, "A New Fast Heuristic for Computing the Breakpoint Phylogeny and Experimental P hylogenetic Analyses of Real and Synthetic Data," in: *Proceedings of the 8th International Conf. o n Intelligent Systems for Molecular Biology* (ISMB 2000), San Diego, 2000, pp. 104–115.

[DFR03] F. Dehne, M. Fellows and F. Rosamond, "An FPT algorithm for Set Splitting," to appear.

[DLS02] S. Demri, F. Laroussinie and Ph. Schnoebelen, "A parametric Analysis of the State Explosion problem in Model Checking (Extended Abstract)" in *Proc STACS 2002* LNCS 2285, Springer-Verlag, 2002, 620-631.

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

[DEFPR03] R. Downey, V. Estivill-Castro, M. Fellows, E. Prieto-Rodriguez and F. Rosamond, "Cutting Up Is Hard To Do: the Parameterized Complexity of k-Cut and Related Problems," in *Proc. Australian Theory Symposium, CATS 2003*, Elsevier Electronic Notes In Computer Science.

[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.

[DF99a] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer-Verlag, 1999.

[DF99b] R. Downey and M. Fellows, "Parameterized Complexity After Almost Ten Years: Review and Open Questions," in: *Combinatorics, Computation and Logic, DMTCS'99 and CATS'99*, Australian Computer Science Communications, Springer-Verlag Singapore, vol. 21 (1999), 1–33.

[DFS98] R. Downey, M. Fellows and U. Stege, "Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability," *Contemporary Trends in Discrete Mathematics,* (R. Graham, J. Krachovil, J. Nesetril, and F. Roberts, eds) DIMACS Vol. 49, American Mathematical Society, (1999) 49-100.

[DFS99] R. Downey, M. Fellows and U. Stege, "Computational Tractability: the View from Mars," *Bulletin of the European Association for Theoretical Computer Science,* No. 69, (1999), 73-97.

[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*, Vol. 209 (1998), 123-140.

[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.

[DFVW99] R. Downey, M. Fellows, A. Vardy and G. Whittle, "The Parameterized Complexity of Some Fundamental Problems in Coding Theory," *SIAM J. Comput.* Vol. 29 (1999), 545-570.

[DFMcC03] R. Downey, J. Fouhy and C. McCartin, " Pathwidth, Online Algorithms, and Presentations I," in preparation.

[DFZZ01] X. Deng, H. Feng, P. Zhang and H. Zhu, "A Polynomial Time Approximation Scheme for Minimizing Total Completion Time of Unbounded Batch Scheduling," *Proc. 12th International Symposium on Algorithms and Computation* (ISAAC '01), Springer-Verlag, *Lecture Notes in Computer Science* 2223 (2001), 26–35.

[DRST01] F. Dehne, A. Rau-Chaplin, U. Stege and P. Taillon, "Solving Large FPT Problems on Coarse Grained Parallel Machines," manuscript, 2001.

[DFHKLMcCRRSWW01] V. Dujmovic, M. Fellows, M. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. Rosamond, M. Suderman, S. Whitesides, D. Wood, "On the Parameterized Complexity of Layered Graph Drawing." *ESA 2001* 488-499

[Ed65] J. Edmonds, "Minimum Partition of a Matriod into Independent Sets," *J. Res. Nat. Bur. Stand. Sect. B.*, Vol. 69 (1965), 67-72.

[EL87] E. Emerson and C. Lei, "Modalities for Model Checking: Branching Time Logic Strikes Back," *Scence of Computer Programming*, 8 (1987), 275-306.

[EJS01] T. Erlebach, K. Jansen and E. Seidel, "Polynomial Time Approximation Schemes for Geometric Graphs," *Proc. ACM Symposium on Discrete Algorithms* (SODA'01), 2001, pp. 671–679.

[Fe89] M. R. Fellows, "The Robertson-Seymour Theorems: a Survey of Applications,"in *Contemporary Mathematics* Vol. 89, AMS, (1989) 1-18.

[Fe03] M. Fellows, "Parameterized complexity: the main ideas and connections to practical computing." In: R. Fleischer et al. (Eds.) *Experimental Algorithmics,* LNCS 2547 (2002), 51–77.

[Fe] M. Fellows, Personal communication, March 2003.

[FK93] M. Fellows and N. Koblitz, "Fixed-Parameter Complexity and Cryptography," *Proceedings of the Tenth International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC'93)*, Springer-Verlag, Berlin, Lecture Notes in Computer Science vol. 673 (1993), 121–131.

[FMcRS01] M. Fellows, C. McCartin, F. Rosamond and U. Stege, "Trees with Few and Many Leaves," manuscript, full version of the paper: "Coordinatized kernels and catalytic reductions: An improved FPT algorithm for max leaf spanning tree and other problems," *Proceedings of the 20th FST TCS Conference*, New Delhi, India, Lecture Notes in Computer Science vol. 1974, Springer Verlag (2000), 240-251.

[FHK95] M. Fellows, M. Hallett and D. Kirby, "The Parameterized Complexity of the Shortest Common Supersequence Problem," manuscript, 1995.

[FG02a] J. Flum and M. Grohe, "Describing Parameterized Complexity Classes," Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS'02), Lecture Notes in Computer Science 2285, pp.359-371, Springer-Verlag 2002. (Abstract)

[FG02b] J. Flum and M. Grohe, "The Parameterized Complexity of Counting Problems," Conference version appeared in Proceedings of the 43rd IEEE Symposium on Foundations of Comupter Science (FOCS'02), pp. 538-547, 2002.

[FFGta] J. Flum, M. Frick and M. Grohe, "Query Evaluations via Tree-Decompositions," preliminary version in *Procs 8th International Conf. on Database Theory,* LNCS 1973, Springer-Verlag, 2001.

[FrG01] M. Frick and M. Grohe, "Deciding First Order Properties of Locally Tree-Decomposable Structures," *J. ACM*, 48 (2001), 1184-1206.

[FrG02] M. Frick and M. Grohe, "The Complexity of First-Order and Monadic Second-Order Logic Revisited," in *LICS*, 2002, 215-224.

[Fo03] J. Fouhy, "Computational Experiments on Graph Width Metrics," MSc Thesis, Victoria University, Wellington, 2003.

[GGRV01] M. Galota, C. Glasser, S. Reith and H. Vollmer, "A Polynomial Time Approximation Scheme for Base Station Positioning in UMTS Networks," *Proc. Discrete Algorithms and Methods for Mobile Computing and Communication*, 2001.

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

[GW03] J. Geelen and G. Whittle, "Branch-Width and Rota's Conjecture," *J. Comb. Theory B.*, to appear.

[GGKS95] P. Goldberg, M. Golumbic, H. Kaplan, and R. Shamir, "Four Strikes Against DNA Physical mapping," *Journal of Computational Biology*, 2(1), 139-152, 1995.

[GN00] J. Gramm and R. Niedermeier, "Faster Exact Algorithms for Max2Sat," *Proc. 4th Italian Conference on Algorithms and Complexity*, Springer-Verlag, *Lecture Notes in Computer Science* 1767 (2000), 174–186.

[GNR01] J. Gramm, R. Niedermeier, and P. Rossmanith. "Exact Solutions for Closest String and Related Problems," in *Proc. 12th ISAAC*, Springer-Verlag LNCS 2223, pp. 441–453, 2001. Long version to appear in *Algorithmica* 2003.

[GK02] G. Gottlob and C. Koch, "Monadic Queries over Tree Structured Data," *LICS, 2002*, 189-202.

[GN02] J. Gramm and R. Niedermeier. "Breakpoint Medians and Breakpoint Phylogenies: a Fixed-Parameter Approach," to appear in *Proc. First European Conference on Computational Biology*, October 2002, Saarbrücken. Supplement to *Bioinformatics* 18 (Supplement 2): S:128–S139, Oxford University Press, 2002.

[Gr01a] M. Grohe, "Generalized Model-Checking Problems for First-Order Logic," *Proc. STACS 2001*, Springer-Verlag LNCS vol. 2001 (2001), 12–26.

[Gr01b] M. Grohe, "The Parameterized Complexity of Database Queries," *Proc. PODS 2001*, ACM Press (2001), 82–92.

[Gr01c] M. Grohe, "Computing Crossing Numbers in Quadratic Time," Conference version appeared in *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC'01),* pp.231-236, 2001. (Abstract)

[Gr02] M. Grohe, "Parameterized Complexity for the Database Theorist," SIGMOD Record 31(4), 2002.

[GSS01] M. Grohe, T. Schwentick and L. Segoufin. "When is the Evaluation of Conjunctive Queries Tractable ?" In *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC'01)*, pp.657-666, 2001. (Abstract)

[GoSS01] G. Gottlob, F. Scarcello and M. Sideri, "Fixed Parameter Complexity in AI and Nonmonotonic Reasoning," to appear in *The Artificial Intelligence Journal*. Conference version in: *Proc. of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning* (LPNMR'99), vol. 1730 of *Lecture Notes in Artificial Intelligence* (1999), 1–18.

[Hl02a] Petr Hlineny, "Practical Computation with Representable Matroids (the Macek Program)," to appear.

[Hl03] Petr Hlineny, "Branch-Width, Parse Trees, and Second-Order Monadic Logic for Matroids" (Extended Abstract). In: *Theoretical Aspects of Computer Science STACS* 2003 (H. Alt and M. Habib Eds.), Lecture Notes in Computer Science 2607, Springer Verlag (2003), 319–330.

[IPZ01] R. Impagliazzo, R. Paturi and F. Zane, "Which problems have strongly exponential complexity?", JCSS 63(4),: 512–530, 2001.

[Ir94] S. Irani "Coloring Inductive Graphs On-Line," *Algorithmica*, vol.11, (1994), pp.53-72.

[KW90] S. Kannan and T. Warnow, "Inferring Evolutionary History from DNA Sequences," in *Proceedings of the 31st Annual Symposium on the Theory of Computing,* 1990, 362-378.

[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.

[KN01] Y. Karuno and H. Nagamochi, "A Polynomial Time Approximation Scheme for the Multi-vehicle Scheduling Problem on a Path with Release and Handling Times," *Proc. 12th International Symposium on Algorithms and Computation* (ISAAC '01), Springer-Verlag, *Lecture Notes in Computer Science* 2223 (2001), 36–47.

[KM96] S. Khanna and R. Motwani, "Towards a Syntactic Characterization of PTAS," in: *Proc. STOC 1996*, ACM Press (1996), 329–337.

[KR00] S. Khot and V. Raman, 'Parameterized Complexity of Finding Subgraphs with Hereditary properties', *Proceedings of the Sixth Annual International Computing and Combinatorics Conference (COCOON 2000)* July 2000, Sydney, Australia, Lecture Notes in Computer Science, Springer Verlag **1858** (2000) 137-147. (TCS 2002)

[KQ95] H. Kierstead and Qin Jun, "Coloring interval graphs with First-Fit," *Discrete Math.* 144 (1995) 47-57.

[KBFvH03] A. Koster, H. Bodlaender, J. Fouhy, S. van Hoesel, "Treewidth: Computational Experiments," to appear.

[Le83] H. Lenstra, "Integer Programming with a Fixed Number of Variables," *Mathematics of Operations Research,* 8 (1983), 538-548.

[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.

[McC01] C. McCartin, "An improved algorithm for the jump number problem," *Information Processing Letters* 79(2) (2001), 87-92.

[McC02] C. McCartin, "Parameterized Counting Problems," in *Proceedings Mathematical Foundations of Computer Science,* 2002, 556-567.

[McC03] C. McCartin, "Contributions to Parameterized Complexity," PhD Thesis, Victoria University, Wellington, 2003.

[McMWW93] F. McMorris, T. Warnow, and T. Wimer, "Triangulated Vertex Coloured Graphs," in *Proceedings of the 4th Annual Symposium on Discrete Algorithms,* 1993, Austin, Texas, 120-127.

[MR99] M. Mahajan and V. Raman, "Parameterizing Above Guaranteed Values: MaxSat and MaxCut," *J. Algorithms* 31 (1999), 335-354.

[Mo99] B. Mohar, “A Linear Time Algorithm for Embedding Graphs in an Arbitrary Surface,” *SIAM J. Discrete Math* 12 (1999) 6-26

[MWBWY01] B. Moret, S. Wyman, D. Bader, T. Warnow and M. Yan, “A New Implementation and Detailed Study of Breakpoint Analysis,” *Proc. 6th Pacific Symposium on Biocomputing* (PSB 2001), pp. 583–594. to appear.

[NRW03] L. Nakhleh, D. Ringe, and T. Warnow ”Perfect Phylogenetic Networks: A New Methodology for Reconstructing the Evolutionary History of Natural Languages.” Submitted for publication, 2003.

[Nie98] R. Niedermeier, “Some Prospects for Efficient Fixed-Parameter Algorithms,” *Proc. SOFSEM’98* Springer-Verlag LNCS 1521 (1998), 168–185.

[Nie02] R. Niedermeier, “Invitation to Fixed-Parameter Algorithms,” Habilitationschrift, University of Tübingen, September, 2002.

[NR03] R. Niedermeier and P. Rossmanith, “An Efficient Fixed Parameter Algorithm for 3-Hitting Set,” *Journal of Discrete Algorithms* to appear 2003.

[NSS98] A. Natanzon, R. Shamir and R. Sharan, “A Polynomial-Time Approximation Algorithm for Minimum Fill-In,” *Proc. ACM Symposium on the Theory of Computing* (STOC’98), ACM Press (1998), 41–47.

[PY93] C. Papadimitriou and M. Yannakakis, “On Limited Nondeterminism and the Complexity of the V-C Dimension”, *Eight Annual Conference on Structure in Complexity Theory*, 12–18, 1993.

[PY97] C. Papadimitriou and M. Yannakakis, “On the Complexity of Database Queries,” *Proc. ACM Symp. on Principles of Database Systems* (1997), 12–19. Journal version in *Journal of Computer System Sciences*, 58 (1999), 407-427.

[PSta] E. Prieto and C. Sloper “Either/Or: Using VERTEX COVER Structure in Designing FPT Algorithms -The Case of k-Internal Spanning Tree,” to appear.

[Se91] D. Seese, “The structure of Models of Decidable Monadic Theories of Graphs,” *Ann. Pure and Appl. Logic*, Vol. 53, (1991) 169-195.

[ST98] R. Shamir and D. Tzur, “The Maximum Subforest Problem: Approximation and Exact Algorithms,” *Proc. ACM Symposium on Discrete Algorithms* (SODA’98), ACM Press (1998), 394–399.

[ST00] H. Shachnai and T. Tamir, “Polynomial Time Approximation Schemes for Class-Constrained Packing Problems,” *Proc. APPROX 2000*.

[Sl93] M.Slusarek, “A Lower Bound to the First-Fit Coloring of Interval Graphs,” Universitas Iagellonica Acta Scientiarum Litterarumque, Schedae Informaticae, vol 5 (1993), 25-32

[St92] M. Steel, “The Complexity of Reconstructing Trees from Qualitative Characters and Subtrees,” *J. Classification,* Vol. 9 (1992), 91-116.

[St00] U. Stege, “Resolving Conflicts in Problems in Computational Biochemistry,” Ph.D. dissertation, ETH, 2000.

[Ra97] V. Raman, “Parameterized Complexity,” in: *Proceedings of the 7th National Seminar on Theoretical Computer Science*, Chennai, India (1997), 1–18.

[RS86a] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[RS86b] N. Robertson and P. Seymour, “Graph Minors V : Excluding a Planar Graph,” *J. Comb. Theory B.*, 41 (1986), 92-114.

[RS95] N. Robertson and P. Seymour, “Graph Minors XIII : The Disjoint Paths Problem,” *J. Comb. Theory B.*, 63 (1995), 65-110.

[Va82] M. Vardi, “The Complexity of Relational Database Queries,” *Proc. STOC*, 1982, 137-146.

[Va95] M. Vardi, “On the Complexity of Bounded-Variable Queries,” *Proc. STOC*, 1995, 266-276.

[WRT95] T. Warnow, D. Ringe, and A. Taylor, “Reconstructing the Evolutionary Hiestory of Natural Languages,” IRCS Report 95-16, University of Pennsylvania, 1995.

[Wei00] K. Weihe, “On the Differences Between ‘Practical’ and ‘Applied’ (invited paper),” *Proc. WAE 2000*, Springer-Verlag, *Lecture Notes in Computer Science* 1982 (2001), 1–10.

[WLBCRT98] B. Wu, G. Lancia, V. Bafna, K-M. Chao, R. Ravi and C. Tang, “A Polynomial Time Approximation Scheme for Minimum Routing Cost Spanning Trees,” *Proc. SODA ’98*, 1998, pp. 21–32.

[Ya95] M. Yannakakis,“Perspectives in Database Theory,” in *FOCS*, 1995, 224-246.