A Basic Parameterized Complexity Primer

Rod Downey*

School of Mathematics, Statistics and Operations Research
Victoria University
P. O. Box 600, Wellington, New Zealand.
rod.downey@vuw.ac.nz

Abstract. This article was prepared for Mike Fellows Festschrift for his 60th Birthday. Since many of the contributed articles revolve around the concept of parameterized complexity, it seems reasonable to give the reader a (short) primer to this area. It is not intended as a complete survey of this very broad area in its current state; rather it is intended to give a flavour of the techniques used and the directions taken. Whilst not doing the area justice, the basics of the techniques for proving tractability, establishing hardness, and the philosophy are given. The basics from this paper will be amplified by many other articles in this Festschrift. Much fuller accounts can be found in the books Downey-Fellows [DF98,DFta], Niedermeier [Nie06], Flum-Grohe [FG06], the two issues of the *Computer Journal* [DFL08] and the recent survey Downey-Thilikos [DTH11].

1 Introduction

1.1 The idea

The story of classical complexity, as witnessed by the classic cartoons in the beginning of Garey and Johnson's book [GJ79], begins with some problem we wish to find an efficient algorithm for. Now, what do we mean by efficient? It seems a reasonable idea to idealize the notion of being efficient by being in *polynomial time*. Having done this, we discover that the only algorithm we have for the given problem is to try all possibilities and this takes $\Omega(2^n)$ for instances of size n. What we would like is to *prove* that there is no algorithm running in feasible time. Using our idealization that feasible=polynomial, this equates to showing that there is no algorithm running in polynomial time.

Suppose that we succeed in showing that there is no polynomial time algorithm. This would mean to us is that we would (i) need to try some other method to solve the problem such as some kind of approximate solution because (ii) we could give up on showing that there was a polynomial time algorithm.

The story continues with the following rhetoric. In spite of the efforts of a number of researchers, for many problems whose best solution known was complete search, there was no *proof* that the problem is not in polynomial time

* Research supported by the Marsden Fund of New Zealand. Dedicated to my old friend Mike on the occasion of his 60th Birthday.

found. It was the key realization of Cook, Levin and crucially Karp [Ka72] that many of these problems could be shown to be polynomial-time reducible to each other, and to the problem of acceptance for a polynomial time nondeterministic Turing machine. That is, they are NP-complete. This means that we have a *practical* "proof" of hardness in that if any of the problems were in polynomial time, all would be; and secondly showing them to be in polynomial time would show that acceptance for a polynomial time nondeterministic Turing machine would be also. The philosophical argument is that a nondeterministic Turing machine is such an opaque object, without any obvious algebraic structure, that it seems impossible to see if it has an accepting path without trying all of them. That's the philosophy anyway.

The methodology above seems fine as a *first foray* into *feasible* computation. However, for practical computation, it seems that we ought to refine the analysis to make it more *fine grained*. Firstly, when we show that something is NP-complete or worse, what we are focusing on is the worst case behaviour. Second, the analysis takes the input as being measured by its *size* alone. You can ask yourself the question: when in real life do we know nothing else about a problem than its *size*? The answer is *never*. For instance, the problem is planar, tree-like, has many parameters bounded, etc. The idea behind parameterized complexity is to try to exploit the *structure* of the input to get some practical tractability. That is, we try to understand what aspect of the problem is to blame for the combinatorial explosion which occurs. If this parameter can be controlled then we would have achieved practical tractability.

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 (see, for example Papadimitriou and Yannakakis [PY97], which posits parameterized complexity as the correct complexity for such analyses). Furthermore, in areas like computational biology, the number 4 is typical, and structure of things like DNA is *far* from random. 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 might profitably exploit. The idea is not to replace polynomial time 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 develop some relevant hardness theory to show that the kind of structure is not useful for this approach.

As I remarked in [Do03], "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."

1.2 Some definitions

I will now discuss the standard examples which we use for the theory. As I discuss in the companion paper [Do12], Mike Fellows and my early work had the three problems Vertex Cover, Dominating Set,Independent Set in our hearts.

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. Of course, these are some of the basic NP-complete problems identified by Karp [Ka72].

As in [Do03], and earlier [DF98] and [DFS98], I will motivate the definitions by looking at a problem in computational biology. As discussed in [Do12] in this volume, and as seen by [GGKS95,KST94,St00,DFS98,BDFHW95] computational biology has been interacting with parameterized complexity from the beginning, and this interaction has continued with throughout, with the work Langston and his group (who have contracts throughout the world to analyse biological data, and use Vertex Cover and other FPT techiques routinely), of Niedermeier and his group, and others. This volume describes many of the applications to computational biology in Stege [St12]. Suppose we had a conflict graph of some data from this area. 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^{12} 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.

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 $(10^{12})^{50}$ many possibilities. Of course this is completely impossible.

Vertex Cover There is now an algorithm running in time $O(1.2738^k + kn)$ ([CKX10]) for determining if an G has a vertex cover of size k. This and and structurally similar algorithms has been implemented and is practical for n of unlimited practical size and k large. The relevant k has been increasing all the time, evolving from about 400 in [CDRST03], to Langston's team [LPSSV08,ELRW11] who now routinely solve instances on graphs with millions of nodes and vertex covers in the thousands. Moreover, this last work is on actual biological data.

As well as using bounded branching (and parallelization [ALSS06]), the method used for this algorithm for VERTEX COVER is called *kernelization* and is based on reduction rules[1], which tend to be easy to implement and perform often much better than anticipated in practice. We will discuss this method in detail soon. The following table from Downey-Fellows [DF98] exhibits the difference between the parameter k being part of the exponent like DOMINATING SET or as part of the constant like VERTEX COVER. This table compares of a running time of $\Omega(n^k)$ vs $2^k n$.

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

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

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 $\leq k$?

We could, for instance, parameterize the problem in other ways. For example, we could parameterize by some width metric, some other shape of the graph, planarity etc. Any of these would enable us to seek hidden tractability in the problem at hand.

For a formal definition, for simplicity I will stick to the *strongly uniform* definition of being fixed-parameter tractable. There are other definitions of less importance in practice, and I refer the reader to [DF98] or [FG06] for more details.

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 $L \subseteq \Sigma^* \times \mathbb{N}$. Flum and

[1] There are other FPT methods based around reduction rules such as Leizhen Cai [LeC96] and Khot and Raman [KR02], which work on certain hereditary properties.

Grohe have an alternative formulation where the second coordinate is a function $\kappa : \Sigma^* \to \Sigma^*$, but I prefer to keep the second parameter as a string or number.

Definition 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 $\leq f(k)|x|^c$.

It is not difficult to show that the multiplicative constant in the definition can be replaced by an additive one, so that $L \in FPT$ iff L can be accepted by a machine in time $O(|x|^c) + f(k)$ for some computable f. In the case of Vertex Cover we have $f(k) = 1.2738^k$, and the O is 2. One nice notation useful here is the O^* notation which ignores the polynomial part be it additive or multiplicative and is only concerned with the exponential part. The algorithm would be said to be $O^*(2^k)$. The table on the web site

`http://fpt.wikidot.com/fpt-races`

lists 35 (at the time of writing) basic problems which are fixed parameter tractable with (mostly) practical algorithms, and for which there are current "races" for algorithms with the best run times.

Now you might (in some cases validly) complain about the presence of an arbitrarily bad computable function f. Could this not be like, for example Ackermann's function? This is a true enough complaint, *but* the argument also applies to *polynomial time*. Could not polynomial time allow for running times like $n^{30,000,000}$? As noted by Edmonds [Ed65], the practical algorithm builder's answer tends to be that "in real life situations, polynomial time algorithms tend to have small exponents and small constants." That certainly was true in 1965, but as we will see this is no longer true. The same heuristic applies here. By and large, for most practical problems, at least until recently, the $f(k)$'s tended to be manageable and the exponents reasonable.

In fact, an important offshoot of parameterized complexity theory is that it does (sometimes) provide tools to show that *bad constants* or *bad exponents* for problems with algorithms *running in polynomial time cannot* be eliminated, modulo some reasonable complexity assumption. As articulated by Alehknovich and Razborov [AR01] who were considering lower bounds for the automizability[2] of resolution and tree-like resolution, what they needed was a complexity theory *sensitive to the structure of polynomial time.* We emphasize that exploring

[2] Alehknovich and Razborov studied proof systems P called in [BPR01] *automatizable* 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. They proved neither resolution nor tree-like resolution is automatizable unless $W[P]$ is randomized FPT by a randomized algorithm with one-sided error, where $W[P]$ is a class we will meet in Section 2

feasible computation requires something like parameterized complexity as it is a theory giving hardness *within polynomial time*. More on this in Section 2.

One of the key features of the theory is a wide variety of associated techniques for proving parametric tractability. We will discuss them in Section 3, but before we do so, let's examine the associated hardness theory.

2 Parametric Intractability

Since we are woefully bad at *proving* problems to be not in polynomial time, we have invented a hardness theory, as mentioned in Section 1, based on the assumption that certain canonical problems are not in polynomial time. The two key ingredients of a hardness theory are (i) a notion of hardness and (ii) a notion of "problem A could be solved efficiently if we could solve problem B"; that is a notion of reducibility.

In the classic theory of NP completeness (i) is achieved by 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?

The Cook-Levin 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 $O(|M|^c)$ 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?

So here is a notion of hardness. Personally I would find it difficult to believe that NP is not P, but that Short Nondeterministic Turing Machine Acceptance could be in FPT, for example, solved in $O(|M|^3)$ for any path length k. In fact, as we will soon see, Short Nondeterministic Turing Machine Acceptance not in FPT is closely related to the statement n-variable 3Sat not being solvable in subexponential time.

Thus to show Dominating Set is likely not FPT could be achieved by showing that *if we could solve it in time $O(n^c)$ by for each fixed k, then we could have a $O(n^c)$ for* Short Nondeterministic Turing Machine Acceptance. Our principal working definition for parameterized reductions is the following.

Definition 2. *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 $\langle G', k'\rangle \in L'$.

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]; as I discuss in [Do12], later in this volume.

Theorem 1. *The following are hard for* Short Nondeterministic Turing Machine Acceptance*:* Independent Set*,* Dominating Set*.*

Following Karp [Ka72], and then four decades of work, we know that thousands of problems are all NP-complete. They are all reducible to one another and hence seem to have the same classical complexity. On the other hand, with parameterized complexity, we have theory which separates Vertex Cover from Dominating Set and Independent Set. With such refined reducibilties, it seems highly unlikely that the hardness classes would coalesce into a single class like NP-complete. And indeed we think that this is the case. We have seen in the theorem above that Short Nondeterministic Turing Machine Acceptance$\equiv_{fpt}$ Independent Set. However, we *do not* think that Dominating Set$\leq_{fpt}$ Independent Set.

A standard parameterized version of the satisfiability problem of Cook-Levin is the following. (Other parameterized versions are discussed in the article by Chen and Flum [CF12] in this volume.)
Weighted Cnf Sat
Input: A CNF formula X.
Parameter: A number k.
Question: Does X have a true assignment of weight k (here the weight is the number of variables set to true)?

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^P_m$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 instance of 3 Cnf Sat.

Now, early on, as I mention in [Do12], Fellows and I came to the belief that there is *no parametric reduction at all* from Weighted Cnf Sat to Weighted 3 Cnf Sat. Fellows and I proved 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. Fellows and I then defined 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

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

There are many, many problems hard for $W[1]$ and complete at many levels of this hierarchy. I won't list them here, but examples can be found in this volume, and in the papers and books listed above. The basic papers are [DF92a,DF92b,DF93,ADF95], and there we define the basic W-classes and essay the completeness programme.

The reader might ask about parameterizing *space*. This issue was addressed by Abrahamson, Downey and Fellows [ADF93,ADF95], where the complexity of k-move games was addressed. (It could be argued that a proper treatment of space is yet to be done.) The complexity of k-move games was addressed by extending the hierarchy above using ideas of *alternation* of parameterized quantifiers, giving a hierarchy called the AW-hierarchy. Again we refer to the books and survey articles.

There are also other hierarchies based on other ideas of *logical depth*. One important hierarchy of this kind was found by Flum and Grohe is the A-hierarchy which is also based on alternation like the AW-hierarchy but works differently. For a class Φ of formulae, we can define the following parameterized problem.

p-MC(Φ)
Instance: A structure $\mathcal{A}$ and a formula $\varphi \in \Phi$.
Parameter: $|\varphi|$.
Question: Decide if $\phi(\mathcal{A}) \neq \emptyset$, where this denotes the evaluation of ϕ in $\mathcal{A}$.

Flum and Grohe define

$$\mathrm{A}[t] = [p\text{-MC}(\Sigma_t)]^{\mathrm{FPT}}.$$

For instance, for $k \geq 1$, k-Clique can be defined by

$$\text{clique}_k = \exists x_1, \ldots x_k (\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j \wedge \bigwedge_{1 \leq i < j \leq k} E x_i x_j)$$

in the language of graphs[3], and the interpretation of the formula in a graph G would be that G has a clique of size k. Thus the mapping $(G, k) \mapsto (G, \text{clique}_k)$ is a fixed parameter reduction showing that parameterized Clique is in A[1]. Flum and Grohe populate various levels of the A-hierarchy and show the following.

Theorem 2 (Flum and Grohe [FG02a,FG04]). *The following hold:*

(i) A[1] =W[1].

[3] For narrative flow, I will assume that the reader is familiar with logic, but a more precise discussion will be given in Section 4.5.

(ii) $\mathsf{A}[t] \subseteq \mathsf{W}[t]$.

Clearly $\mathsf{A}[t] \subseteq \mathsf{XP}$, but no other containment with respect to other classes of the W-hierarchy is known. It is conjectured by Flum and Grohe that no other containments than those given exist. This conjecture is not apparently related to any other conjecture. One other important hierarchy, called the M-hierarchy will be discussed later, but in any case it is completely evident that the fine-grained nature of the notion of parametric complexity will lead to significant structure within polynomial time. There is still a great deal to do here.

2.1 Connection with PTAS's

The reader may note that parameterized complexity is addressing intractability *within polynomial time*. In this vein, the parameterized framework can be used to demonstrate that many classical problems that admit a PTAS do not, in fact, admit any PTAS with a practical running time, unless W[1] =FPT. The idea here is that if a PTAS has a running time such as $O(n^{\frac{1}{\epsilon}})$, where ϵ is the error ratio, then the PTAS is unlikely to be useful. For example if $\epsilon = 0.1$ then the running time is already n to the 10th power for an error of 10%. Here is a table from Downey [Do03]

- Arora [Ar96] gave a $O(n^{\frac{3000}{\epsilon}})$ PTAS for Euclidean Tsp
- Chekuri and Khanna [CK00] gave a $O(n^{12(\log(1/\epsilon)/\epsilon^8)})$ PTAS for Multiple Knapsack
- Shamir and Tsur [ST98] gave a $O(n^{2^{2^{\frac{1}{\epsilon}}}-1)})$ PTAS for Maximum Subforest
- Chen and Miranda [CM99] gave a $O(n^{(3mm!)^{\frac{m}{\epsilon}+1}})$ PTAS for General Multiprocessor Job Scheduling
- Erlebach *et al.* [EJS01] gave a $O(n^{\frac{4}{\pi}(\frac{1}{\epsilon^2}+1)^2(\frac{1}{\epsilon^2}+2)^2})$ PTAS for Maximum Independent Set for geometric graphs.

Table 2 below calculates some running times for these PTAS's with a 20% error.

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

Table 2. The Running Times for Some Recent PTAS's with 20% Error.

In Downey [Do03], I argue as follows.

"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 just the 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[4] *PTAS (EPTAS), or a proof that no such PTAS exists*[5]*. 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."*

To use the hardness theory to eliminate the possibility of feasible PTAS's, what we could do is regard $\frac{1}{\epsilon}$ as a parameter and show that the problem is W[1]-hard with respect to that parameterization. In that case *there would likely be no method of removing the* $\frac{1}{\epsilon}$ *from the exponent in the running time and hence no* efficient *PTAS*, a method first used by Bazgan [Baz95]. For many more details of the method we refer the reader to the surveys [Do03,DTH11].

It was an insight of Cai and Juedes that tight lower bounds for approximation and parameterized complexity are intimately related; and indeed, are also related to classical questions about NP and subexponential time. In particular, Cai et. al. [CFJR07] who showed that the method of using planar formulae tends to give PTAS's that are never practical. The exact calibration of PTAS's and parameterized complexity comes through yet another hierarchy called the M-hierarchy.

The base level of the hierarchy is the problem M[1] defined by the core problem below.

Instance: A CNF circuit C (or, equivalently, a CNF formula) of size $k \log n$, with n in unary.
Parameter: A positive integer k.
Question: Is C satisfiable?

That is, we are parameterizing the *size* of the problem rather than some aspect of the problem. The idea naturally extends to higher levels for that, for example, $M[2]$ would be a product of sums of product formula of size $k \log n$ and we are asking whether it is satisfiable. The basic result is thatFPT$\subseteq$ M[1]$\subseteq$W[1].

[4] An *Efficient Polynomial-Time Approximation Scheme (EPTAS)* is an $(1+\epsilon)$-approximation algorithm that runs in $f(1/\epsilon) \cdot n^{O(1)}$ steps. If, additionally, f is a polynomial function then we say that we have a *Fully Polynomial-Time Approximation Scheme (FPTAS).*

[5] The same issue can also be raised if we consider FPTAS's instead of EPTA S's.

The hypothesis FPT≠M[1] is *equivalent to* a classical conjecture called the *exponential time hypothesis*, ETH. This hypothesis is due to Impagliazzo, Paturi and Zane [IPZ01] and asserts that n-variable 3Sat cannot be solved in "subexponential time", DTime $(2^{o(n)})$. This conjecture accords with the intuition that not only does $P \neq NP$ but actually NP is really at exponential level.

One example of a lower bound was the original paper of Cai and Juedes [CJ01,CJ03] who proved the following definitive result.

Theorem 3 (Cai and Juedes [CJ01,CJ03]). k-Planar Vertex Cover, k-Planar Independent Set, k-Planar Dominating Set, *and* k-Planar Red/Blue Dominating Set *cannot be in* $O^*(2^{o(\sqrt{k})})$*-FPT unless* FPT=M[1] *(or, equivalently, unless ETH fails).*

We remark that Theorem 3 is optimal as all the problems above have been classified as $O^*(2^{O(\sqrt{k})})$ (see e.g. Downey and Thilikos [DTH11])

The obvious connection between subexponential complexity and parameterized complexity classes as formalized by Chen and Grohe [CG07] by constructing an isomorphism, the so-called *miniaturization*, between exponential time complexity (endowed with a suitable notion of reductions) and XP (endowed with FPT reductions) such that the respective notions of tractability correspond, that is, subexponential time on the one and FPT on the other side. Other connections between classical complexity and "canonical" miniaturizations can be found in Downey, Flum, Grohe and Weyer [DFGW07].

2.2 XP-optimality

There is a new programme akin to the above establishing tight lower bounds on parameterized problems, assuming various non-collapses of the parameterized hierarchies. A powerful example of this is what is called XP optimality. This new programme regards the classes like W[1] as artifacts of the basic problem of proving hardness under reasonable assumptions, and strikes at membership ofXP. We illustrate this via Independent Set and Dominating Set which certainly are in XP. But what's the best exponent we can hope for for slice k?

Theorem 4 (Chen et. al [CCFHJKX05]). *The following hold:*

(i) Independent Set *cannot be solved in time* $n^{o(k)}$ *unless* FPT=M[1].
(ii) Dominating Set *cannot be solved in time* $n^{o(k)}$ *unless* FPT=M[2].

More on parameterized intractability and its development over time can be found in the article by Jianer Chen [Ch12] in this volme.

3 Positive Techniques

3.1 Bounded Search Trees

A fundamental source of high running times is *branching* in algorithms. A very crude idea to limit the running time is to keep this branching small and a function

of the parameter. For instance, for VERTEX COVER, we can do this as follows. Take any edge $e = vw$, and begin a search tree, by having leaves labeled by v and w, and for each leaf recursively do this for the graphs gotten by deleting any edge covered by v and w respectively. The depth of this process for a k-vertex cover is k and then we can decide of G has a vertex cover in time $O(2^k|G|)$ using this method.

At a certain point in any of these algorithms, you need to appeal to some kind of combinatorics to improve performance. A simple illustration of this idea is that if we can make the search tree smaller than the complete binary tree of length k, then the performance will improve. Notice that, if G has no vertex of degree three or more, then G consists of a collection of cycles, and this is pretty trivial to check. Thus we can assume we have vertices of higher degree than 2. For vertex cover of G we must have either v or *all of its neighbours*, so we create children of the root node corresponding to these two possibilities. The first child is labeled with $\{v\}$ and $G - v$, the second with $\{w_1, w_2, \ldots, w_p\}$, the neighbours of v, and $G - \{w_1, w_2, \ldots, w_p\}$. In the case of the first child, we are still looking for a size $k - 1$ vertex cover, but in the case of the second child we need only look for a vertex cover of size $k - p$, where p is at least 3. Thus, the bound on the size of the search tree is now somewhat smaller than 2^k. It can be shown that this algorithm runs in time $O(5^{k\backslash 4} \cdot n)$, and in typical graphs, there are lots of vertices of higher degree than 3, and hence this works even faster.

The best algorithms along these lines use more complex branching rules. For example, Niedermeier [Nie02] uses the following branching rules.

BRANCHING RULE VC1:
If there is a degree one vertex v in G, with single neighbour u, then there is a minimum size cover that contains u. Thus, we create a single child node labeled with $\{u\}$ and $G - u$.

BRANCHING RULE VC2:
If there is a degree two vertex v in G, with neighbours w_1 and w_2, then either both w_1 and w_2 are in a minimum size cover, or v together with *all other neighbours* of w_1 and w_2 are in a minimum size cover.

BRANCHING RULE VC3:
If there is a degree three vertex v in G, then either v or all of its neighbours are in.

We remark that using these three rules, (using a recurrence relation) it can be shown that if there is a solution of size at most k then the size of the corresponding search tree has size bounded above by $O(1.47^k)$. More involved rules of similar ilk exploring the *local structure* of neighbourhoods in graphs, result in the algorithm of Chen et. al. [CKX10]) with running time $O(1.2738^k)$ for the branching.

There are a number of problems for which this technique is the only method, or at least the best method, for parameterized algorithms. The method has been particularly successful in computational biology with problems like the CLOS-

EST STRING problem [GNR01] and MAXIMUM AGREEMENT FOREST problem [HM07].

In passing I remark that this method is inherently parallelizable and as we see is often used in conjunction with other techniques. The method for VERTEX COVER can be found discussed in [ALSS06].

The paper in this volume by Marx explores the applicability of this technique against other methods like kernelizability discussed in the next section.

3.2 Kernelization

This is again a pretty simply basic idea. If we can make the problem *smaller* then the search will be quicker. This is a *data reduction* or *pre-processing* idea, and is the heart of many heuristics.

Whilst there are variations of the idea below, the simplest version of kernelization is the following.

Definition 3 (Kernelization).
Let $L \subseteq \Sigma^* \times \Sigma^*$ be a parameterized language. A reduction to a problem kernel, or kernelization, comprises replacing an instance (I, k) by a reduced instance (I', k'), called a problem kernel, such that

(i) $k' \leq k$,
(ii) $|I'| \leq g(k)$, for some function g depending only on k, and
(iii) $(I, k) \in L$ if and only if $(I', k') \in L$.

The reduction from (I, k) to (I', k') must be computable in time polynomial in $|I| + |k|$.

There are other notions, where the kernel may be another problem (often "annotated") or the parameter might increase, but, crucially, the size of the kernel depends *only on* k.

Here are some natural reduction rules for a kernel for VERTEX COVER.
REDUCTION RULE VC1:
Remove all isolated vertices.

REDUCTION RULE VC2:
For any degree one vertex v, add its single neighbour u to the solution set and remove u and all of its incident edges from the graph.

These rules are obvious. Sam Buss (see [DF98]) originally observed that, for a simple graph G, any vertex of degree greater than k must belong to every k-element vertex cover of G (otherwise all the neighbours of the vertex must be included, and there are more than k of these).
This leads to our last reduction rule.
REDUCTION RULE VC3:
If there is a vertex v of degree at least $k+1$, add v to the solution set and remove v and all of its neighbours.

After exhaustively applying these rules, we get to a graph (G', k'), where no vertex in the reduced graph has degree greater than $k' \leq k$, or less than two. Then simple combinatorics shows that if such a reduced graph has a size k vertex cover, its must have size $\leq k^2$. This is the size k^2 kernelization.

Now we can apply the bounded depth search tree rule to this reduced graph, and get an algorithm for vertex cover running in time $O(1.2738^k)k^2$. As observed by Langston and his team in problems in sequence analysis, and articulated by Niedermeier and Rossmanith [NR00] better running times can be obtained by *interleaving* depth-bounded search trees and kernelization. That is, first kernelize, begin a bounded search tree, and the *rekernelize* the children, and repeat. This really does make a difference. In [Nie02] the 3-Hitting Set problem is given as an example. An instance (I, k) of this problem can be reduced to a kernel of size k^3 in time $O(|I|)$, and the problem can be solved by employing a search tree of size 2.27^k. Compare a running time of $O(2.27^k \cdot k^3 + |I|)$ (without interleaving) with a running time of $O(2.27^k + |I|)$ (with interleaving).

In actual implementations there are other considerations such as load sharing amongst processors and the like. We refer to the articles in the Computer Journal special issue concerning practical FPT.

We also remark that there are many strategies of reduction rules to *shrink* the kernel. these include things like *crown reductions* (Abu-Khzam et. al. [ACFLSS04]), and other crown structures (such as N. Abu-Khzam et. al.[AFLS07]) which generalize the notion of a degree 1 vertex having its neighbours in the vertex cover, to more complicated structures which resemble "crowns" attached to the graph[6]. In fact generalizing this to even more complex structures called *protrusions* which have a well-behaved structure of small "treewidth" (we will soon meet in Section 5.1) is an excellent source of theoretically efficient algorithms as evidenced by the "metakernelization" paper Bodlaender et. al. [BFLPST09], though the practicality of such is not at all explored.

Clearly, another game is to seek the smallest kernel. For instance, we know by Nemhauser and Trotter [NT75] a size $2k$ kernel is possible for Vertex Cover. A natural question is "can we do better?". As we later see, modulo some complexity considerations, sometimes we can show lower bounds on kernels. (Clearly, if $P = NP$ then all have constant size kernels, so some assumption is needed.) We refer to the site

[6] Specifically, a *crown* in a graph $G = (V, E)$ consists of an independent set $I \subseteq V$ (no two vertices in I are connected by an edge) and a set H containing all vertices in V adjacent to I. A crown in G is formed by $I \cup H$ iff there exists a size $|H|$ maximum matching in the bipartite graph induced by the edges between I and H, that is, every vertex of H is matched. It is clear that degree-1 vertices in V, coupled with their sole neighbours, can be viewed as the most simple crowns in G. If we find a crown $I \cup H$ in G, then we need at least $|H|$ vertices to cover all edges in the crown. Since all edges in the crown can be covered by admitting at most $|H|$ vertices into the vertex cover, there is a minimum size vertex cover that contains all vertices in H and no vertices in I. These observations lead to the reduction rules based on deleting crowns.

http://fpt.wikidot.com/fpt-races

for lots of kernel races.

The state of the art in the theory of kernelization can be found in the article by Saket Saurabh and Neeldhara Misra [SM12] in this volume.

It is not hard to show that a problem is FPT iff it is kernelizable. However, it is not true that FPT=*polynomial size* kernelizable, and this is discussed in the article by Marx [Ma12] in this volume, along with a future agenda for parameterized complexity.

Another practical technique for establishing parameterized tractability is the following.

3.3 Iterative Compression

This technique was first introduced in a paper by Reed, Smith and Vetta in 2004 [RSV04] and more or less re-discovered by Karp [Ka11]. Although currently only a small number of results are known, it seems to be applicable to a range of parameterized minimization problems, where the parameter is the size of the solution set. Most of the currently known iterative compression algorithms solve *feedback set problems* in graphs, problems where the task is to destroy certain cycles in the graph by deleting at most k vertices or edges. In particular, the k-Graph Bipartisation problem, where the task is to find a set of at most k vertices whose deletion destroys all odd-length cycles, has been shown to be FPT by means of iterative compression [RSV04]. This had been a long-standing open problem in parameterized complexity theory.

Definition 4 (Compression Routine).
A compression routine *is an algorithm that, given a problem instance I and a solution of size k, either calculates a smaller solution or proves that the given solution is of minimum size.*

Here is a compression routine for Vertex Cover. Begin with $(G = (V, E), k)$, we build the graph G vertex by vertex. We start with an initial set of vertices $V' = \emptyset$ and an initial solution $C = \emptyset$. At each step, we add a new vertex v to both V' and C, $V' \leftarrow V' \cup \{v\},\ C \leftarrow C \cup \{v\}$. We then call the compression routine on the pair $(G[V'], C)$, where $G[V']$ is the subgraph induced by V' in G, to obtain a new solution C'. If $|C'| > k$ then we output NO, otherwise we set $C \leftarrow C'$.

If we successfully complete the nth step where $V' = V$, we output C with $|C| \leq k$. Note that C will be an optimal solution for G.

The compression routine takes a graph G and a vertex cover C for G and returns a smaller vertex cover for G if there is one, otherwise, it returns C unchanged. Each time the compression routine is used it is provided with an intermediate solution of size at most $k + 1$.

The implementation of the compression routine proceeds as follows. We consider a smaller vertex cover C' as a *modification* of the larger vertex cover C. This modification retains some vertices $Y \subseteq C$ while the other vertices $S = C \setminus Y$

are replaced with $|S| - 1$ new vertices from $V \setminus C$. The idea is to try by brute force all $2^{|C|}$ partitions of C into such sets Y and S. For each such partition, the vertices from Y along with all of their adjacent edges are deleted. In the resulting instance $G' = G[V \setminus Y]$, it remains to find an optimal vertex cover that is disjoint from S. Since we have decided to take no vertex from S into the vertex cover, we have to take that endpoint of each edge that is not in S. At least one endpoint of each edge in G' is in S, since S is a vertex cover for G'. If both endpoints of some edge in G' are in S, then this choice of S cannot lead to a vertex cover C' with $S \cap C' = \emptyset$. We can quickly find an optimal vertex cover for G' that is disjoint from S by taking every vertex that is not in S and has degree greater than zero. Together with Y, this gives a new vertex cover C' for G. For each choice of Y and S, this can be done in time $O(m)$, leading to $O(2^{|C|}m) = O(2^k m)$ time overall for one call of the compression routine. With at most n iterations of the compression algorithm, we get an algorithm for k-Vertex Cover running in time $O(2^k mn)$.

The parametric tractability of the method stems from the fact that each intermediate solution considered has size bounded by some $k' = f(k)$, where k is the parameter value for the original problem. It works very with with *monotone* problems, where if we get an intermediate no then the answer is definitely no. Note that many minimization problems are not monotone in this sense. For example, a NO instance $(G = (V, E), k)$ for k-Dominating Set can be changed to a YES instance by means of the addition of a single vertex that is adjacent to all vertices in V.

Niedermeier [Nie06] has an excellent discussion of this technique, which would seem to have a lot of applications.

4 Not-Quite-Practical FPT Algorithms

There are a number of distinctive techniques used in parameterized complexity which are "not-quite-practical" FPT algorithms, in the sense that the running times are not feasible in general, but can be in certain circumstances. Additionally, some can be randomized and ones using logical metatheorems can later admit considerable refinement in practice for a *specific* problem. These techniques include color-coding and dynamic programming on bounded width graph decompositions. Since this survey is meant to be brief, I will only allude to these techniques.

4.1 Colour-coding

This technique is useful for problems that involve finding small subgraphs in a graph, such as paths and cycles. Introduced by Alon et al. [AYZ94], it can be used to derive seemingly efficient randomized FPT algorithms for several subgraph isomorphism problems.

It remains in the "not quite practical" basket due to the large numbers needed to implement it. Here is a 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 $O(k2^k|E|)$.

Theorem 5 (Alon, Yuster and Zwick [AYZ94]). *k-PATH can be solved in expected time $2^{O(k)}|E|$.*

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?" The desired FPT algorithm can now be obtained by a process of derandomization. A *k-perfect family of hash functions* is a family $\mathcal{F}$ of functions (colorings) taking $[n] = \{1, \ldots n\}$ onto $[k]$, such that for all $S \subseteq [n]$ of size k there is a $f \in \mathcal{F}$ whose restriction to is bijective (colourful). It is known that k-perfect families of $2^{O(k)} \log n$ linear time hash functions. This gives a deterministic $2^{O(k)}|E| \log |V|$ algorithm for k-PATH. More such applications can be found in Downey and Fellows [DF98], and Niedermeier [Nie02,Nie06]. The $O(k)$ in the exponent hides evil, and the derandomization method at present seems far from practical.

Note that the method does not work when applied to 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.

We remark that recent work ([BDFH08,BDFH09]) has shown, assuming a reasonable complexity assumption (namely that the polynomial time hierarchy does not collapse to two or fewer levels), there is no polynomial size kernel for k-PATH. We meet this result in Section 6.

4.2 Bounded Integer Programming

One technique, not discussed in [DF98,FG06], is the use of INTEGER PROGRAMMING in the design of FPT algorithms. This is discussed in Niedermeier [Nie02,Nie06].

Theorem 6 (Lenstra [Le83]). *The integer programming feasibility problem can be solved with $O(p^{\frac{9p}{2}}L)$ arithmetical operations in $\mathbb{Z}$ of $O(p^{2p}L)$ bits in size, where p is the number of variables, and L the number of bits of the input.*

Niedermeier [Nie02,Nie06] 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 $s_1, \ldots s_k$ over an alphabet Σ each having length L, and a non-negative 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].

4.3 Bounded 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 bounded global parameters are found in the notions of *width metrics*, and in particular through treewidth and local treewidth (defined in the next section). 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. The following definition is now quite mainstream in modern graph theory.

Definition 5 (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 treelike the graph is. One can similarly define *path decomposition* where $\mathcal{T}$ must be a path. A tree decomposition is a road map as to how to build the graph. Knowing a tree or path decomposition of a graph allows for dynamic programming since what is important is the "information flow" across a relatively narrow cut in the graph.

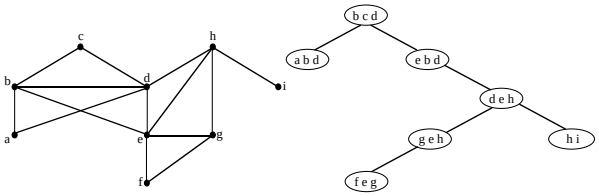

Fig. 1. Example of Tree Decomposition of Width 2

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.

4.4 Algorithms for graphs of bounded treewidth

We sketch how to run algorithms on graphs of bounded treewidth. This can be viewed as *dynamic programming* a very important algorithmic technique. Many classical problems are known to be algorithmically infeasible on general graphs (in that they take exponential time) but become polynomial time when restricted to some bounded treewidth class. A good introduction to this technique is Bienstock and Langston [BL95], Bodlaender and Kloks [BK96], Bodlaender and Koster [BoK08], or Babette de Fluiter [deF70]. I am always surprised when I meet people who are unaware of this technique, since it has been around so long. But for completeness it seems worthwhile to describe the method.

We describe the technique for the Independent Set. Whilst this problem is classically $W[1]$-complete, in the case of graphs of bounded treewidth, we can give a linear time algorithm. So suppose that we have a tree decomposition of G. Consider the one below

Now what we will do is to use *tables* to grow the independent set up the tree starting at the leaves of the decomposition. Notice that once a vertex leaves the bags it will never come back, and hence we don't really need to keep track of its effect. Thus we can work with tables corresponding to all the subsets of the current bag and need only consider independent sets I relative to the current bag. That is, we would consider all the subsets of the bag and see how the size of independent sets relative to in the information flow across the boundary.

Thus at the leaf corresponding to the triangle abc we could have the table below. Here, for instance ab denotes the set $\{a, b\}$, meaning that the independent

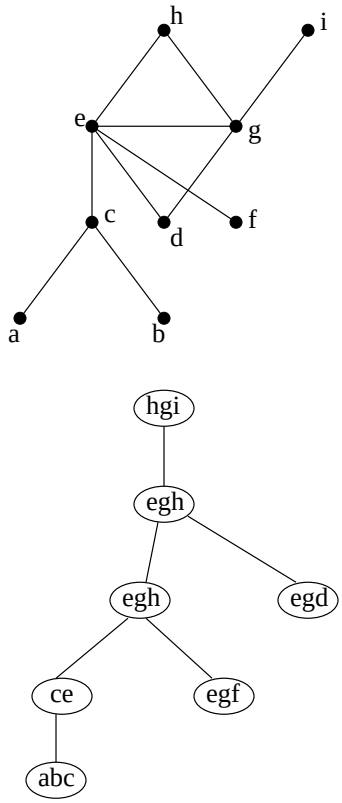

Fig. 2. A tree decomposition

set should contain both a and b, and would have cardinality 2, and bc corresponds to $\{b, c\}$ and this entry has a line since $\{b, c\}$ is *not* an independent set.‘

$\emptyset$	a	b	c	ab	ac	bc	abc
0	1	1	1	2	-	-	-

The table for the next box would have only 4 columns since there are only 4 subsets of $\{c, d\}$ and we consider maximal independent sets I containing the specified subset

$\emptyset$	c	e	ce
2	1	3	-

The first entry has value 2 since this means neither of c or d in included in this I and hence we take the maximal one below, namely 2.

The second entry says that I must contain c, and this corresponds to the entry from below containing c, which is 1. The third one corresponds to the independent set containing e *and not* c, and hence we get 3, by choosing the largest one with this property from below (namely the entry for ab contains no c, and this together with e gives $1 + 2 = 3$.)

Finally, the last entry is 0 since I would need to contain both c and d and no independent set has this property.

The rest of the table if filled in similarly. With pointers you can also keep track of the relevant independent set.

The next table is a join node

$\emptyset$	e	g	h	eg	eh	gh	egh
0	1	1	1	2	-	-	-

This table corresponds to $I \cap \{e, g, h\}$ being the specified set. The first entry corresponds to I having nothing from this set. It has value 3 since we get that from the 2 on the left and 1 on the right. (We must add here.)

The second entry corresponds to the intersection being $\{e\}$, meaning that it must have e and cannot have h or g. Note e is present in both branches and hence we get 3. Next is g with e and h not present. There are 2 from the left (corresponding to the $\emptyset$), and g is present on the right so we take its entry giving 3. the next is h and neither e or g, giving 2 from the left branch, 2 from the right (using f since h is not present) and h itself; giving the total of 4. etc

The final table looks like:

We remark that a sly feature we have not mentioned here is that to run this method, *we need a tree decomposition for the graph* G. It turns out that for a fixed t there is a *linear time algorithm* determining if G has width t, and then finds the tree decomposition of G should the graph actually have one. However, the algorithm has really terrible constants and there is no actually feasible one for this problem. For more on this we refer the reader to Bodlaender [Bod93,Bod96], Bodlaender-Kloks [BK96] and Bodlaender's web site. John Fouhy in his MSc Thesis [Fou03] looked at computational experiments for treewidth heuristics.

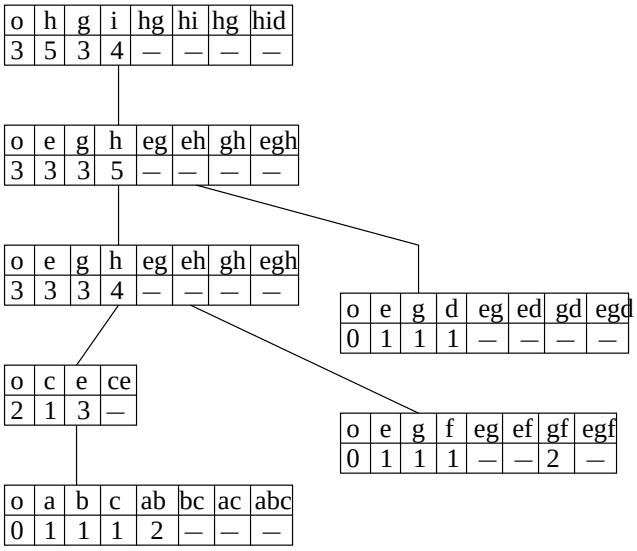

Fig. 3. Dynamic programming

(See www.mcs.vuw.ac.nz/∼downey/students.html.) There is a lot of work to be done here.

Actually, this whole process can be implemented by automata acting on trees. We refer the reader to Downey and Fellows [DF98], Fellows and Langston [FL89], or Abrahamson-Fellows [AF93].

Treewidth is the archetype of a number of graph width metrics[7] naturally arise in this context which restrict the inherent complexity of a graph in various senses. The idea here is that a useful width metric should admit efficient algorithms for many (generally) intractable problems on the class of graphs for which the width is small. This leads to consideration of these measures from a parameterized point of view. The corresponding naturally parameterized problem has the following form:

Let $w(G)$ denote any measure of graph width.
Instance: A graph $G = (V, E)$.
Parameter: A positive integer k.
Question: Is $w(G) \leq k$?

The hope is that this problem is FPT, and then, equipped with the relevant decomposition, we can run algorithms. Another important example of a width metric is *cliquewidth* where a clique decomposition works as follows. We inductively define graphs using sets of $k + 1$ colours and *parse* operators as follows. The first operator is c_i which says "create a vertex of colour i." The second one

[7] And have been used in other settings such as matroids where the width corresponds to the dimension of the intersection of "subspaces" in some decomposition, See, for example Hlineny Whittle [HW06].

is $j(i,j)$ which says "join all vertices of colour i to all vertices of colour j." The is dj which says "form the disjoint union of all graphs constructed so far", and finally $r(i,j)$ which says "recolour all vertices of colour i to j." The width of the decomposition is the smallest number of colours necessary minus 1. For example, a clique has with 1 : to make a clique of size n, create a vertex coloured 1, and then one of colour 2, apply $j(1,2)$, then $r(1,2)$, then create a new vertex of colour 1, and repeat enough times. It is not hard to show that if G has bounded treewidth then it also has bounded cliquewith.

It is unknown if CLIQUEWIDTH is FPT or $W[1]$ hard. It is known to be NP-complete for k varying by Fellows, Rosamond, Rotics and Szeider [FRRS06,FRRS09]. There is a lot of evidence that it is parametrically hard such as Fomin et. al. [FGLS10].

However, sometimes we are lucky and are supplied with a clique decomposition. The point is that *given* a clique decomposition we can run linear time algorithms for some problems. Thus the *certificated* problem where we input G and its clique decomposition, and ask questions is often FPT.

For much more on treewidth and other width parameters, we refer the article in this volume by Bodlaender [Bod12].

4.5 Logic

One source of (often impractical) FPT results is the use of metatheorems from logic. The idea is that to feed in some logical description of the problem, and use some algorithmic machine to given an FPT algorithm for it. The generality of the methodology often means that the algorithms obtained have large constants, and implementations need fine tuning. Sometimes you can use methods from (parameterized) complexity to show that the large constants *can't* be removed, but more on this later.

We tend to look at problems defined in *first-order logic* and *monadic second-order logic*. We remind the reader that first order logic uses individual variables and form the logic by the following rules.

1. *Atomic formulas:* $x = y$ and $R(x_1, ..., x_k)$, where R is a k-ary relation symbol and $x, y, x_1, ..., x_k$ are individual variables, are FO-formulas.
2. *Conjunction, Disjunction:* If ϕ and ψ are FO-formulas, then $\phi \wedge \psi$ is an FO-formula and $\phi \vee \psi$ is an FO-formula.
3. *Negation:* If ϕ is an FO-formula, then $\neg\phi$ is an FO-formula.
4. *Quantification:* If ϕ is an FO-formula and x is an individual variable, then $\exists x\ \phi$ is an FO-formula and $\forall x\ \phi$ is an FO-formula.

First order logic allows for the description of local behaviour of structures. For instance to say that a graph has an independent set of size k,

$$\exists x_1 \dots x_k \bigwedge_{1 \le i \le j \le k} \neg E(x_i, x_j),$$

where $E(x,y)$ denoted the edge relation on the graph.

For monadic second order logic we add *set* variables, one for each subset of vertices in the graph. Formulas of monadic second-order logic (MSO) are formed by the rules for FO and the following additional rules:

1. *Additional atomic formulas:* For all set variables X and individual variables y, Xy is an MSO-formula.
2. *Set quantification:* If ϕ is an MSO-formula and X is a set variable, then $\exists X\, \phi$ is an MSO-formula, and $\forall X\ \phi$ is an MSO-formula.

We can state that a graph is k-colorable using an MSO-formula,

$$\exists X_1 \ldots \exists X_k \left(\forall x \bigvee_{i=1}^{k} X_i x \ \wedge\ \forall x \forall y \Big(E(x,y) \rightarrow \bigwedge_{i=1}^{k} \neg (X_i x \wedge X_i y) \Big) \right)$$

Actually I am being sloppy here as there are variations of the meaning of this depending on whether individual and set variables are allowed for edges; and if so this is denoted by MS_2. There is a whole industry here devoted to the use of logic in algorithmc for computer science, and I will concentrate on MS_2 only to give the flavour of the methodology. For more on this see [DF98,FG06].

Now, whilst model checking for even first order formulae is already PSPACE complete, we have the following.

Theorem 7 (Courcelle 1990). *The model-checking problem for MS_2 restricted to graphs of bounded treewidth is linear-time fixed-parameter tractable.*

Thus, the Independent Set problem above can easily be obtained from Courcelle's Theorem by simply writing a formula in monadic second order logic describing that G has a k-independent set.

Actually Courcelle's Theorem is true for a mild extension of MS_2 called MS_2^+ where certain "counting" relations are added. The result gives the flavour of the methods from logic. If we have a suitably restricted class of graphs, then model checking becomes FPT. Examples include first order logic for families of graphs of bounded *local treewidth*. This is the notion of how fast the treewidth grows in a neighbourhood of an vertex of any graph in the family. This is called bounded if there is a function f, such that for all n, the treewidth of the n-neighbourhood of a vertex of any member of the family is bounded by $f(n)$. Examples of classes of graphs that have bounded local treewidth include graphs of bounded treewidth (naturally), graphs of *bounded degree*, *planar* graphs, and graphs of *bounded genus*.

Theorem 8 (Frick and Grohe [FrG01]). *Parameterized problems that can be described as model-checking problems for FO are fixed-parameter tractable on classes of graphs of bounded local treewidth.*

There are other notions of bounded with where this works. In general, things like treewidth, cliquewith, and any other of these metrics indicate the the member of of the graph family will be built by certain inductive methods. Such

methods clearly have reflections in reality when we consider how, for example, computer chips are designed. It is not surprising that such inductive families tend to have better algorithmics than general graphs. For more on this we refer the reader to [GK11].

5 Exotica, WQO theory

In this last section we look at exotic methods for proving problems to be FPT. These tend to only give membership of FPT and no practical algorithms. As discussed in Downey [Do12] and Langston [La12] in this Festschrift, this material is the parent of parameterized complexity, in a sense made clear in those papers, and emanating from material such as [FL87,FL88].

A *quasi-ordering* on a set S is a reflexive transitive relation on S. We will usually represent a quasi-ordering as $\leq_S$ or simply $\leq$ when the underlying set S is clear. Let $\langle S, \leq \rangle$ be a quasi-ordered set. We will write $x < y$ if $x \leq y$ and $y \not\leq x$, $x \equiv y$ if $x \leq y$ and $y \leq x$ and finally $x|y$ if $x \not\leq y$ and $y \not\leq x$. Note that if $\langle S, \leq \rangle$ is a quasi-ordered set then $S/\equiv$ is partially ordered by the quasi-order induced by $\leq$. Recall that a *partial ordering* is a quasi-ordering that is also antisymmetric.

Definition 6 (Ideal and Filter). *Let $\langle S, \leq \rangle$ be a quasi-ordered set. Let S' be a subset of S.*

(i) We say that S' is a filter *if it is closed under $\leq$ upwards. That is, if $x \in S'$ and $y \geq x$ then $y \in S'$. The* filter generated by S' *is the set $F(S') = \{y \in S : \exists x \in S'(x \leq y)\}$.*[8]

(ii) We say that S' is a (lower) ideal *if S' is $\leq$ closed downwards. That is if $x \in S'$ and $y \leq x$ then $y \in S'$. The* ideal generated by S' *is the set $I(S') = \{y \in S : \exists x \in S'(x \geq y)\}$.*

(iii) Finally, if S' is a filter (an ideal) that can be generated by a finite subset of S' then we say that S' is finitely generated.

We will need some distinguished types of sequences of elements.

Definition 7. *Let $\langle S, \leq \rangle$ be a quasi-ordered set. Let $A = \{a_0, a_1, ...\}$ be a sequence of elements of S. Then we say the following.*

(i) A is good *if there is some $i < j$ with $a_i \leq a_j$.*

(ii) A is bad *if it is not good.*

(iii) A is an ascending chain *if for all $i < j$, $a_i \leq a_j$.*

(iv) A is an antichain *if for all $i \neq j$ $a_i|a_j$.*

(v) $\langle S, \leq \rangle$ is Noetherian *if S contains no infinite (strictly) descending sequences. (i.e. there is no sequence $b_0 > b_1 > b_2....$)*

(vi) $\langle S, \leq \rangle$ has the finite basis property *if for all subsets $S' \subseteq S$, $F(S')$ is finitely generated.*

The following result is relevant to our work.

[8] Sometimes, filters are called *upper ideals*.

Theorem 9 (Folklore, after Higman [Hi52]). *Let $\langle S, \leq \rangle$ be a quasi-ordered set. The following are equivalent.*

(i) $\langle S, \leq \rangle$ *has no bad sequences.*
(ii) Every infinite sequence in S *contains an infinite chain.*
(iii) $\langle S, \leq \rangle$ *is Noetherian and* S *contains no infinite antichain.*
(iv) $\langle S, \leq \rangle$ *has the finite basis property.*

Definition 8 (Well Quasi-Ordering). *Let $\langle S, \leq \rangle$ be a quasi-ordering. If $\langle S, \leq \rangle$ satisfies any of the characterizations of Theorem 9, then we say that $\langle S, \leq \rangle$ is a* well quasi-ordering *(WQO).*

The reader might well wonder what any of this abstract pure mathematics has to do with algorithmic considerations. The key is provided by the finite basis characterizations of a WQO. Suppose that $\leq$ is the relevant quasi-ordering and for a fixed x the question "Q_y: Is $x \leq y$?" is FPT (resp. polynomial time). Then for a WQO, if F is a filter, then F has a finite basis $\{b_1, ..., b_n\}$. Then to decide if $y \in F$ we need only ask "$\exists i \leq n(b_i \leq_S y)$?" That is, membership of each filter is FPT (resp. polynomial time) (even though we don't know the finite basis!).

Often the argument is phrased in terms of *obstruction sets*. Let $\langle S, \leq \rangle$ be a quasi-ordering. Let I be an ideal of $\langle S, \leq \rangle$. We say that a set $O \subseteq S$ forms an *obstruction set* for I if

$$x \in I \text{ iff } \forall y \in O(y \not\leq x).$$

That is O is an obstruction set for I if I is the complement of $F(O)$. The WQO principle says all ideals have finite obstruction sets.

So here is our new engine for demonstrating that problems are in P. Prove that the problem is characterized by a WQO with a finite obstruction set in a quasi-ordering with Q_y in P.

The best known example of an obstruction set is provided by topological ordering.

Definition 9 (Topological Ordering). *A* homeomorphic or topological embedding *of a graph $G_1 = (V_1, E_1)$ in a graph $G_2 = (V_2, E_2)$ is an injection from vertices V_1 to V_2 with the property that the edges E_1 are mapped to disjoint paths of G_2. (These disjoint paths in G_2 represent possible* subdivisions *of the edges of G_1.) The set of homeomorphic embeddings between graphs gives a partial order, called the* topological order. *We write $G_1 \leq_{top} G_2$.*

While topological ordering is not a WQO, there are a number of important finite basis results. The most famous is the following (which was independently discovered by Pontryagin).

Theorem 10 (Kuratowski's Theorem, Kuratowski [Ku30]). *The graphs $K_{3,3}$ and K_5 of Figure 4 form an obstruction set for the ideal of planar graphs in the topological ordering.*

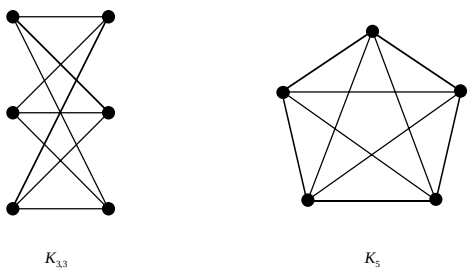

Fig. 4. Obstructions for planarity

Very recently, Grohe, Marx, Kawarbayashi, and Wollan,[GMKW11] proved that $\leq_{top}$ is FPT, with $G \leq_{top} H$ $O(|H|^3)$ for a fixed G. A consequence of that result is that any filter with a finite basis in the topological quasi-ordering has an FPT membership algorithm. The [GMKW11] argument is very difficult, and the algorithm has horrendous constants. We remark that graphs of pathwidth 2 are well-quasi-ordered by $\leq_{top}$, and hence those graphs have FPT membership for any ideal.

A generalization of topological ordering is much more amenable to being a WQO. An equivalent formulation of $\leq_{top}$ is the following. $G \leq_{top} H$ iff G can be obtained from H by a sequence of the following two operations.
(i) (deletions) Deleting vertices or edges.
(ii) (degree 2 contractions) The *contraction* of an edge xy in a graph W is obtained by identifying x with y. A contraction has degree 2 iff one of x or y has degree 2.

The *minor ordering* is a generalization of $\leq_{top}$ is obtained by relaxing the degree 2 requirement in (ii).

Definition 10 (Minor Ordering). *We say that G is a minor of H if G can be obtained from H by a sequence of deletions and contractions. We write $G \leq_{minor} H$.*

An example of the minor ordering is given in Figure 5

The proof of Kuratowski's Theorem also gives the following.

Theorem 11 (Kuratowski's Theorem (II)). *$K_{3,3}$ and K_5 are an obstruction set for the ideal of planar graphs under $\leq_{minor}$.*

The way to think of $\leq_{minor}$ is to think of $G \leq_{minor} H$ as taking $|G|$ many collections C_i of connected vertices of H, and coalescing each collection C_i to a single vertex, then H being topologically embeddable into the new coalesced graph, so that the edges of G become disjoint paths from coalesced vertices. Sometimes this is called the "folio" definition of the minor ordering. Figure 6 below demonstrates this idea.

Notice that H has no vertices of degree 4 and hence there can be no topological embedding of G into H.

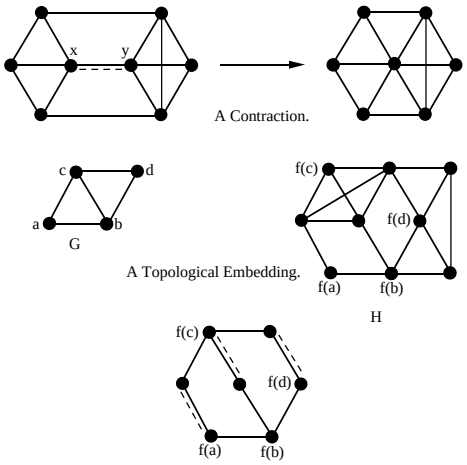

Fig. 5. A contraction and a Topological Embedding

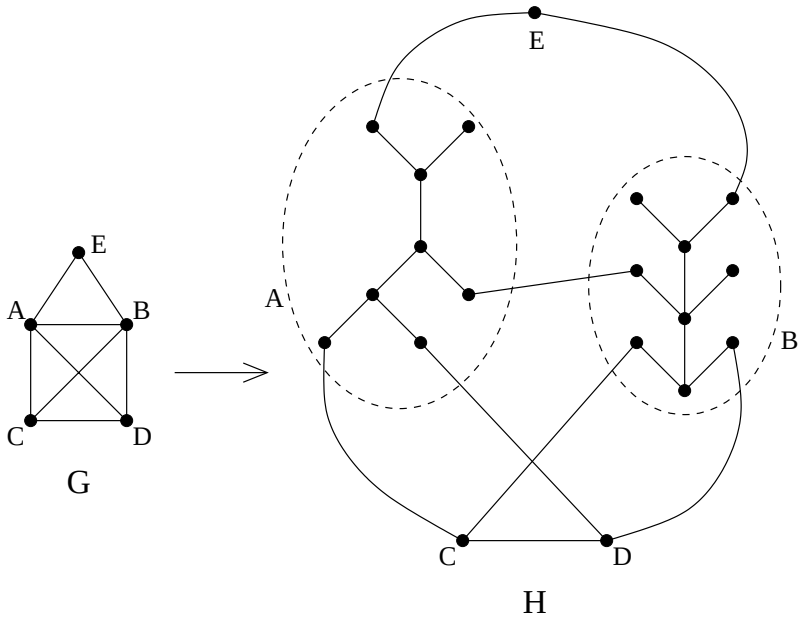

Fig. 6. The folio definition

Wagner [Wa37] conjectured the following.

Wagner's Conjecture: *Finite graphs are well quasi-ordered by the minor ordering.*

Notice that graphs of genus $\leq g$ for a fixed g form an ideal in the minor ordering. Hence a consequence of Wagner's conjecture is a Kuratowski Theorem for surfaces. *Graphs of genus $\leq g$ have a finite obstruction set.* One of the triumphs of 20th century mathematics is the following great theorem of Neil Robertson and Paul Seymour.

Theorem 12 (Graph Minor Theorem, Robertson and Seymour). *Wagner's Conjecture holds: Finite graphs are well quasi-ordered by the minor ordering.*

Remarkably, Robertson and Seymour showed that $x \leq_{minor} y$ is $O(|y|^3)$ for a fixed x. Thus *every minor ideal has a cubic time recognition algorithm!.*

For example, the problem of determining if a graph has genus k for a fixed k becomes FPT immediately. This is, of course, not the best algorithm, and a constructive linear time FPT one was given by Mohar [Mo99].

Multiple applications of this metatheorem can be obtained by the following. We call a parameter p of graphs *treewidth bounded* if the treewidth of G is bounded by $f(p(G))$ for some computable f. We say that p is MSO definable if $\{G : p(G) \leq k\}$ is monadic second order definable by a formula ϕ_k for each fixed k.

Theorem 13 (Adler, Grohe and Kreutzer [AGK08]). *If p is a treewidth bounded MSO definable parameter, then the obstruction set for $\{G : p(G) \leq k\}$ is finite and bounded by $g(k)$ for some computable g. Thus if p is a treewidth bounded minor closed graph parameter, then checking $p(G) \leq k$ is* constructively[9] *in FPT with time bound $g(k).n$*

5.1 Protrusions

Using this notion we can prove a structural lemma about the structure of graphs involving what are called *protrusions.* In my mind this is an extension of the concept of a crown.

Definition 11 (Bodlaender et. al. [BFLPST09]). *Given a graph G, $X \subset V(G)$ is called an* r-protrusion *if the treewidth of $G[X]$ is $\leq r$ and the boundary of X intersection G is $\leq r$.*

Protrusions are good because they allow for efficient kernelizations in the same way that crowns do. Here is an archetypical theorem.

[9] Given the definition of FPT we have used this seems a strange statement. However, simply applying the graph minor machinery establishes that some problem has a cubic time algorithm and we don't know what it is. To know it requires knowledge of the relevant obstruction set. Hence we usually get FPT results for a class known as nonuniform FPT.

Theorem 14 (Bodlaender et. al. [BFLPST09]). *If P is a problem of finite integer index, there is a computable function f and an algorithm which, given an instance (G, k) and an r-protrusion X of size at least $f(r)$ produces an instance (G^*, k^*) such that $|V(G^*)| < |V(G)|$, $k^* \leq k$ and $(G, k) \in P$ iff $(G^*, k^*) \in P$. Furthermore this runs in time $O(|X|)$.*

The idea here is that we can apply this iteratively to show that methods from the graph minor project can be used efficiently, at least insofar as the side of kernels is concerned.

Algorithms based on the structural idea of finding protrusions and kernelizing has seen a lot of development here. In the next section we will give one illustration, and be content with that for this basic introduction.

5.2 Bidimensionality

This is an area that grew from results about *excluded minor* theorems. Excluded minor theorems are of the kind that say if we exclude a certain graph or graphs from being a minor of a members of that family, then that family is well-behaved. It is one of the underlying intuitions for the Robertson Seymour theory. The archetype result is the following

Theorem 15 (Robertson and Seymour). *For every $n > 0$ there is a c_n such that every graph of treewidth $\geq c_n$ has an n-*grid *as a minor.*

This result was extended as follows.

Theorem 16 (Demaine and Hajiaghayi [DeH08]). *For every fixed H there is a c_H such that every H-minor-free graph or treewidth $\geq c_H \cdot n$ has an n-grid as a minor.*

This result can be seen as the first part of a theory called *bidimensionality* theory. This is again a large undertaking and we refer the reader to Demaine and Hajiaghayi [DeH08] for details of the theory.

Here is one example. We say a graph parameter p is *minor bidimensional* if p is closed under minors and p evaluated on the k-grid is $\Omega(k^2)$. A problem Π is called *subgraph separable* if its solutions can be described in terms of vertex subsets of the input graph, and there is some constant d such that for each G and $S \subseteq V(G)$, every optimal solution Z for G, every union H of some subsets of connected components of $G \setminus S$, and every optimal solution Z' for H, we have

$$|Z'| - d|S| \leq |Z \cap H| \leq |Z'| + d|S|.$$

The conditions above tend to be relatively easy to apply. The relevant structural lemma is the following.

Lemma 1 (Separation Lemma). *Suppose Π is a problem that is subgraph separable and minor-bidimensional. Fix a graph H. Then there is a constant $c_{H,\Pi}$ such that for every H-minor-free graph $(G, k) \in \Pi$, there is a subset $S \subset V(G)$, with $|S| = O(k)$ and the treewidth of $G[V \setminus S] \leq c_{H,\Pi}$.*

The lemma allows us to kernelize by using protrusion like actions on the non-core part of the graph. Skipping details, this allows us to show the following.

Theorem 17 (Demaine, Fomin, Hajiaghayi and Thilikos). *Every subgraph separable minor-bidimensional problem Π with finite index has a linear kernel on graphs excluding some fixed graph as a minor.*

Further details and applications of these ideas are beyond the scope of this basic survey and we refer the reader to [DeH08,GK11] for further details of bidimensionality theory and other algorithmic metatheorems.

For much more on the methods for constructivising results on graph minors, we refer to the article by Daniel Lokshtanov and Dimitrios Thilikos [LT12] in this volume.

6 Limitations and lower bounds

Many of the results above gave FPT algorithms but an analysis of the algorithms reveal very bad running times. For many of these if the unparameterized problem is NP complete then they would have a polynomial running time. However, assuming that NP≠P, or something akin to that, we can ask if these running times can be improved. Similarly, we have seen that kernelization to a small kernel is a valuable way to generate practical algorithms (although the kernels and algorithms are far from practical in the case of the algorithms from bidimensionality theory). Again we can ask the same question.

The combinatorics of FPT are sensitive to the issues of polynomial time, and can often be used in this way.

For example, we have seen that Frick and Grohe proved that deciding first-order statements is FPT for every fixed class of graphs of bounded local treewidth. Courcelle shows a similar result for MS_2 for graphs of bounded treewidth. In each case the algorithmic metatheorem gives algorithms where each alternation of quantifier (roughly) gives another power of two in towers of powers of two for the constants. In the case of, for example, local treewidth Frick and Grohe [FrG02] prove such towers of two's cannot be removed unless $\mathsf{W}[\mathsf{P}]$=FPT. Similar results were established by Flum and Grohe for treewidth based around P≠NP.

In terms bounds for kernelizations, there has been a lots of progress in the last few years. Bodlaender, Downey, Fellows and Hermelin [BDFH08,BDFH09] gave general methods for establishing that classes of problems (distillable) did not have polynomial kernels assuming certain unlikely things don't happen to complexity classes. For example, using a lemma of Fortnow and Santhanam [FS11] they showed that no Or-compositional parameterized problem can have a polynomial sized kernel assuming co-$NP \not\subseteq NP/Poly$. Here we refer to [BDFH09,CFM11,FS11,BTY08,HKSWWta] for more details.

7 Left out

I have left out many things, likely close to various researcher's hearts. For example, there is work on parameterized counting (McCartin [McC06] and Flum

and Grohe [FG02b]) where we count the number of paths of length k to define, for instance, $\#W[1]$. One nice theorem here is the following.

Theorem 18 (Flum and Grohe [FG02b]). *Counting the number of cycles of size k in a bipartite graph is $\#\mathsf{W}[1]$-complete.*

This result can be viewed as a parameterized analog of Valiant's theorem on the permanent. Another area is parameterized randomization, such as Downey, Fellows and Regan [DFR98], and Müller [Mu06,Mu08], but here problems remain. Parameterized approximation looks at questions like: Is it possible to have an FPT algorithm which, on parameter k, either outputs a size $2k$ dominating set for G, or says no dominating set of size k? Such algorithms famously exist for Bin Packing and don't exist for most natural $W[P]$ complete problems. Here we refer to Downey, Fellows, McCartin and Rosamond [DFMR08] and Eickmeyer, Grohe and Grüber [EGG08] for more details. We have left out discussions of parameterizing above guaranteed values such as Mahajan and Raman [MR99], plus discussions of the breadth of applications. For the last, we can only point at the relevant books, and the *Computer Journal* issues [DFL08]. In this volume we do look at areas such as artificial intelligence (Szeider [Sz12]), biology (Stege [St12]), cryptography (Koblitz [Ko12]), and social choice (Betzler and Niedermeier [BN12]). But of course, there are many other applications.

There are many other important topics such as implementations, connections with exact algorithms, connections with classical complexity and the like. Space limitations preclude this material being included. Hopefully this brief survey gives the reader the ability to appreciate the articles of this Festschrift.

References

[ACFLSS04] F. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters and C. T. Symons, "Kernelization Algorithms for the Vertex Cover Problem: Theory and Experiments," *Proceedings of the 6th ALENEX 2004*, (2004), 62-69.

[AFLS07] F. Abu-Khzam, M. Fellows, M. Langston, W. Suters, "Crown Structures for Vertex Cover Kernelization," *Theory Comput. Syst.*, Vol. 41(3) (2007), 411-430.

[ALSS06] F. Abu-Khzam, M. Langston, P. Shanbhag and C. Symons, "Scalable Parallel Algorithms for FPT Problems," *Algorithmica* Vol. 45 (2006), 269-284.

[ADF93] K. Abrahamson, R. Downey and M. Fellows, "Fixed parameter intractability II," in *Proceedings Tenth Annual Symposium on Theoretical Aspects of Computer Science* (STACS'93)(Ed. G. Goos and J. Hartmanis), Springer-Verlag Lecture Notes in Computer Science, Vol 665 (1993) 374-385. 73 (1995), 235–276.

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

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

[AGK08] I. Adler, M. Grohe, and S. Kreutzer, "Computing excluded minors," *Proceedings of the of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)*, 641-650.

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

[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.

[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.

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

[BN12] N. Betzler and R. Niedermeier, "Parameterized complexity and social choice," this volume.

[BL95] D. Bienstock and M. Langston, "Algorithmic Implications of the Graph Minor Theorem," in *Handbook of Operations Research and Management Science: Network Models* (M. O. Ball, T. L. Magnanti, C. L. Monma and G. L. Nemhauser, editors), North VHolland, 1995, 481-502.

[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.

[Bod12] H. Bodlaender, "Treewidth and other width parameters," this volume.

[BDFH08] H. Bodlaender, R. Downey, M. Fellows, and D. Hermelin, "On Problems without Polynomial Kernels (Extended Abstract)," in Proceedings of *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008*, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games. Lecture Notes in Computer Science 5125 Springer 2008, 563-574.

[BDFH09] H. Bodlaender, R. Downey, M. Fellows, and D. Hermelin, "On Problems without Polynomial Kernels," *Journal of Computing and System Sciences*. Vol. 75(No. 8): 423-434 (2009)

[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.

[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.

[BFLPST09] H. Bodlaender, F. Fomin, D. Lokshtanov, E. Pennick, S. Saurabh, and D. Thilikos, "(Meta)kernelization", in 50th IEEE FOCS, 2009.

[BK96] H. Bodlaender and T. Kloks, "Efficient and constructive algorithms for the pathwidth and treewdith of graphs," *J. Algorithms*, Vol. 21 (1996), 358-402.

[BoK08] H. Bodlaender and A. Koster, "Combinatorial optimization on graphs of bounded treewidth," *The Computer Journal*, Vol. 51 (2008), 256-269.

[BTY08] H. Bodlaender, S. Thomasse, and A. Yeo, "Analysis of data reduction, transformations give evidence for non-existence of polynomial kernels," Tech. Rept, UU-CS-2008-030, Utrecht Univ. 2008.

[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.

[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.

[CFJR07] L. Cai, M. Fellows, D. Juedes, and F. Rosamond, "The complexity of polynomial time approximation," *Theoretical Computer Science*, Vol. 41 (2007), 459-477.

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

[CJ03] L. Cai and D. Juedes, "On the existence of subexponential parameterized algorithms," *Journal of Computing snd Systems Science,* Vol. 67 (2003), 789-807.

[CDRST03] J. Cheetham, F. Dehne, Rau-Chaplin, U. Stege and P. J. Taillon, "Solving Large FPT Problems on Coarse Grained Parallel Machines," *Journal of Computer and Systems Sciences*, vol. 67, no. 4, (2003) 691-706

[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.

[Ch12] J. Chen, "Parameterized intractability and its development over the time," this volume.

[CCFHJKX05] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, A. Kanj, G. Xia, "Tight lower bounds for certain parameterized NP-hard problems," *Information and Computation*, Vol. 201 (2005), 216-231.

[CFM11] Y. Chen, J. Flum, M. Müller, "Lower Bounds for Kernelizations and Other Preprocessing Procedures," *Theory Comput. Syst.*, Vol. 48(4) (2011), 803-839.

[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.

[CKX10] J. Chen, Iyad A. Kanj, Ge Xia, "Improved upper bounds for vertex cover," *Theor. Comput. Sci.*, 411 (2010), 3736-3756.

[CF12] J. Chen and J. Flum, "A parameterized halting problem," this volume.

[CG07] J. Chen and M. Grohe, "An isomorphism between subexponential and parameterized complexity theory," *SIAM. J. Comput.*, Vol. 37 (2007), 1228-1258.

[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

[DeH08] E. Demaine, M. Hajiaghayi, "The Bidimensionality Theory and Its Algorithmic Applications," *Comput. J.*, Vol. 51(3) (2008), 292-302.

[Do03] R. Downey, "Parameterized complexity for the skeptic," in *Computational Complexity, 18th Annual Conference*, IEEE, 2003, 147-169.

[Do12] R. Downey, "The Birth and Early Years of Parameterized Complexity", this volume.

[DF92a] "Fixed parameter tractability and completeness," (with M. R. Fellows), *Congressus Numerantium*, **87** (1992) 161-187.

[DF92b] R. Downey and M. Fellows, "Fixed parameter intractability," *Proceedings Structure in Complexity, Seventh Annual Conference, IEEE Publication*, (1992) 36-50.

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

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

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

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

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

[DFta] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, Springer-Verlag, 2012, in preparation.

[DFL08] R. Downey, M. Fellows and M. Langston, Two special issue of *The Computer Journal* devoted to Parameterized Complexity, Volume 58 Numbers 1 and 3, 2008, Oxford University Press.

[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.

[DFMR08] R. Downey, M. Fellows, C. McCartin, and F. Rosamond, "Parameterized approximation of dominating set problems," *Information Processing Letters* Vol. 109(1) (2008), 68-70.

[DFR98] R. Downey, M. Fellows and K. Regan. "Parameterized circuit complexity and the W-hierarchy," *Theoretical Computer Science*. Vol. 191 (1998), 97-115.

[DFS98] R. Downey, M. Fellows and U. Stege, "Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability," DIMACS series on Combinatorics in the 21st Century, AMS Publ., 1998.

[DFGW07] R. Downey, J. Flum, M. Grohe, and M. Weyer, "Bounded fixed-parameter tractability and reducibility," *Annals of Pure and Applied Logic* Vol. 148 (2007), 1-19.

[DTH11] R. Downey and D. Thilikos, "Confronting intractability via parameters," *Computer Science Review*. Vol. 5 (2011), 279-317

[ELRW11] J. D. Eblen, M. A. Langston, G. L. Rogers and D. P. Weerapurage, "Parallel Vertex Cover: A Case Study in Dynamic Load Balancing," *Proceedings, Australasian Symposium on Parallel and Distributed Computing*, Perth, Australia, January, 2011.

[Ed65] J. Edmonds, "Paths, trees and flowers," *Canadian J. Math.*, Vol. 17 (1965), 449-467.

[EGG08] K. Eickmeyer, M. Grohe, M. Grüber, "Approximation of Natural W[P]-Complete Minimisation Problems Is Hard," *IEEE Conference on Computational Complexity 2008*, 8-18

[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.

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

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

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

[FRRS06] M. Fellows, F. Rosamond, U. Rotics, and S. Szeider, "Cliquewidth minimization is NP-hard," Proceedings STOC 2006, 354-362.

[FRRS09] M. Fellows, F. Rosamond, U. Rotics, and S. Szeider, "Cliquewidth NP-complete," *SIAM J. on Discrete Mathematics*, Vol. 23 (2009), 909-939.

[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.

[FG04] J. Flum and M. Grohe, "Parameterized Complexity and Subexponential Time," *Bulletin of the European Association for Theoretical Computer Science* Vol. 84, October 2004.

[FG06] J. Flum and M. Grohe, *Parameterized Complexity Theory,* Springer-Verlag, 2006.

[deF70] B. de Fluiter, *Algorithms for graphs of small treewidth*, PhD Thesis, Utrecht Univ, 1970.

[FGLS10] F. Fomin, P. Golovach, D. Lokshtanov, S. Saurabh, "Intractability of Clique-Width Parameterizations," *SIAM J. Comput.*, Vol. 39(5) (2010, 1941-1956.

[FS11] L. Fortnow and R. Santhanam, "Infeasible instances of compression and sucinct pcp's" *Journal of Computing and System Sciences,* Vol. 77 (2011), 91-106.

[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.

[Fou03] J. Fouhy, "Computational Experiments on Graph Width Metrics," MSc Thesis, Victoria University, Wellington, 2003.

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

[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.

[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.

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

[GK11] M. Grohe and S. Kreutzer, "Methods for Algorithmic Meta Theorems," in Martin Grohe, Johann Makowsky (Eds), *Model Theoretic Methods in Finite Combinatorics,* Contemporary Mathematics 558, American Mathematical Society, 2011.

[GMKW11] M. Grohe, D. Marx, K. Kawarbayashi, and P. Wollan, "Finding Topological Subgraphs is Fixed-Parameter Tractable," *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC'11),* 479-488.

[Gu12] G. Gutin, "Parameterized complexity and exact computation," this volume.
[HM07] M. Hallett and C. McCartin, "A Faster FPT Algorithm for the Maximum Agreement Forest Problem," *Theory of Computing Systems*, 41 (3), 2007.
[HKSWWta] D. Hermelin, S. Kratsch, K. Soltys, M. Whalstrom, and X. Wu, "Hierarchies of inefficient kernelization," to appear.
[Hi52] G. Higman, "Ordering by divisibility in abstract algebras," *Proc. London Math. Soc.*, Vol. 2 (1952), 326-336.
[HW06] P. Hlineny and G. Whittle, "Matroid tree-width," *Eur. J. Comb.* Vol. 27(7) (2006), 1117-1128.
[IPZ01] R. Impagliazzo, R. Paturi and F. Zane, "Which problems have strongly exponential complexity?", JCSS 63(4),: 512–530, 2001.
[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.
[Ka72] R. Karp, "Reducibility Among Combinatorial Problems," in *Complexity of Computer Computations* (1972) 85-103.
[Ka11] R. Karp, "Heuristic algorithms in computational molecular biology," *J. Comput. Syst. Sci.*, Vol. 77(1) (2011) 122-128.
[KR02] S. Khot and V. Raman, 'Parameterized Complexity of Finding Subgraphs with Hereditary properties', *Theoretical Computer Science*, 289 (2002) 997-1008,
[Ko12] N. Koblitz, "Crypto galore," this volume.
[Ku30] K. Kuratowski, "Sur le probleme des courbes gauches en topologie," *Fund. Math.*, Vol. 15 (1930), 271-283.
[La12] M. Langston, "Fixed parameterized tractability, a prehistory" this volume.
[LPSSV08] M. Langston, A. Perkins, A. Saxton, J. Scharff and B. Voy "Innovative Computational Methods for Transcriptomic Data Analysis: A Case Study in the Use of FPT for Practical Algorithm Design and Implementation," *The Computer Journal*, Vol. 51 (2008), 26-38.
[Le83] H. Lenstra, "Integer Programming with a Fixed Number of Variables," *Mathematics of Operations Research,* 8 (1983), 538-548.
[LT12] D. Lokshtanov and D. Thilikos, "Constructive graph minors," this volume.
[MR99] M. Mahajan and V. Raman, "Parameterizing Above Guaranteed Values: MaxSat and MaxCut," *J. Algorithms* 31 (1999), 335-354.
[Ma12] D. Marx, "Future directions in parameterized complexity," this volume. (tentative title)
[McC06] C. McCartin, "Parameterized counting problems," *Annals of Pure and Applied Logic*, Vol. 138 (2006), 147-182.
[Mo99] B. Mohar, "A Linear Time Algorithm for Embedding Graphs in an Arbitrary Surface," *SIAM J. Discrete Math* 12 (1999) 6-26
[Mu06] M. Müller, "Randomized Approximations of Parameterized Counting Problems," *IWPEC 2006,* 50-59
[Mu08] M. Müller, "Valiant-Vazirani Lemmata for Various Logics," *Electronic Colloquium on Computational Complexity (ECCC)*, Vol. 15(063) (2008).
[Nie02] R. Niedermeier, "Invitation to Fixed-Parameter Algorithms," Habilitationschrift, University of Tübingen, September, 2002.
[Nie06] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms,* Oxford University Press, 2006.

[NR00] R. Niedermeier and P. Rossmanith, "A general method to speed up fixed-parameter tractable algorithms," *Information Processing Letters*, 73, (2000), 125-129.

[NT75] G. Nemhauser and L. Trotter Jr, " Vertex packings: Structural properties and algorithms," *Mathematical Programming*, 8, (1975), 232-248.

[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.

[SM12] S. Saurabh and N. Misra, "Kernelization," this volume.

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

[St12] U. Stege, "Parameterized complexity in biology and other applications," this volume.

[Sz12] S. Szeider, "Parameterized complexity and artificial intelligence," this volume.

[RSV04] B. Reed, K. Smith and A. Vetta, "Finding odd cycle transversals," *Operations Research Letters*, Vol. 32 (2004), 299-301.

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

[Wa37] K. Wagner, "Uber einer eigenshaft der eberner complexe," *Math. Ann.*, Vol. 14 (1937), 570-590.