A Parameterized Complexity Tutorial

Rod Downey*

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

Abstract. The article was prepared for the LATA 2012 conference where I will be presenting two one and half hour lecctures for a short tutorial on parameterized complexity. 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 Preamble

Following early ideas of Fellows and Langston (such as [FL87,FL88,FL89]) in the early 1990's Mike Fellows and the author initiated a new direction in complexity theory with a series of papers [DF92a,DF92b,DF93], and then a number of other papers with talented co-authors, the state of the art (at the time) being given in the monograph [DF98].

The idea was that to try to devise a complexity theory more attuned to the considerations of practical computation than other kinds of complexity such as the classical theory of NP-completeness.

Now over 20 years have passed. Moreover, the 14 years since the publication of [DF98] has seen remarkable development in the area. There have appeared new books, many surveys, and more importantly, a whole host of new techniques, applications, and structural complexity. A remarkable number of young talented people have entered the field.

It is also remains true that in many computer science departments, and departments applying complexity the theory only penetrates in a kind of random fashion. Whilst I am sure that in the audience there will be people who now know more about parameterized complexity than I do, my plan is to give a basic tutorial for the others. I hope that they will see some applications for themselves, and perceive the two sided dialog that the theory allows between the hardness theory and the methods of establishing parameterized tractability.

* Research supported by the Marsden Fund of New Zealand.

1.2 The idea

The pioneers of complexity theory (such as Edmonds [Ed65]) identified *polynomial time* as a mathematical idealization of what might be considered feasible for the running times of algorithms. As the familiar story of complexity theory runs, we soon discover that for many problems, the only algorithm we can find is to try all possibilities. To do so takes $\Omega(2^n)$ for instances of size n. In spite of the combined efforts of many algorithm designers, we are unable to show that there is no polynomial time algorithm for a wide class of such problems.

The celebrated work 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. 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 belief 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.

Parameterized complexity views the story above 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.

Anyone working where they design actual algorithms for real-life problems knows that you should fine tune the algorithm for the situation at hand. Moreover, in applications such as computational biology, 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.

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.

There will be two parts to this theory. First, there have evolved distinct techniques which seem generally applicable to parameterized problems. In this tutorial, I will look at the main players which include *bounded search trees, kernelization, other reduction techniques such a crown reductions [AFLS07] and Cai's technique [LeC96], iterative compression,* and somewhat less practical but still useful *bounded width metrics*, as well as things like *colour coding methods from logic*, and impractical things like WQO theory.

Hand in hand with this is a hardness theory. I will look at the basic hardness classes such as the W-, A-, M- and other hierarchies. I will also look at how these hierarchies can be exploited to establish practical limitations of the techniques.

1.3 Some definitions

I will now discuss the standard examples which we use for the theory. Mike Fellows and my early work had the three problems VERTEX COVER, DOMINATING SET,INDEPENDENT SET motivating the theory.

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], I will motivate the definitions by looking at a problem in computational biology. 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 analyze biological data, and use VERTEX COVER and other FPT techniques routinely), of Niedermeier and his group, and others. The paper [LPSSV08] describes a classic application to computational biology.

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.

[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.

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 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 $f(k)$ be it additive or multiplicative and is only concerned with the exponential part. The algorithm would be said to be $O^*(n^1)$. 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.

In the following I will briefly mention some of the techniques I will discuss in the tutorial.

2 Positive Techniques

2.1 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. Thus, using this idea, we can decide of G has a vertex cover in time $O(2^k|G|)$ using this method.

The running time of this method can often be exploited by by making the tree smaller, or by using some kind of asymptotic combinatorics to constrain the search. For example, 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.

More involved rules 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 Closest 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].

2.2 Kernelization

The idea is that if we 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 2 (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 *re-kernelize* 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.

The best kernelization for Vertex Cover is due to Nemhauser and Trotter [NT75] and has size $2k$. It is based on matching and we will look at this technique in the tutorial. The technique is quite powerful.

There are many other reduction techniques based on locan graph structure like *crown reductions* ([AFLS07]), and then the generalization to the less practical *prortusions*. See Downey and Thilikos [DTH11].

2.3 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 3 (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.

2.4 Other techniques

Other methods we will discuss will include *colour coding* (Alon, Yuster and Zwick [AYZ94]), *treewidth* (e.g. [BoK08]), and more exotic methods. Given time, we will look at constructivization of the methods via what is called *bidimensionality theory.*

For colour-coding, it is most easily understood with a simple example. The method remains not-quite-practical as the numbers involved are large, but not astronomical. 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* colour the whole graph with k colors, and look for a *colourful* solution, namely one with k vertices of one of each color.

The two keys to this idea are
(i) we can check for colourful 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 1 (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. 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 is where the problem lies, and the derandomization method at present seems far from practical. It should be noted that the method does not show that k-CLIQUE is in randomized FPT because (i) above *fails.* We also remaark that recent work ([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. Thus, this is a classic problem which is in $O^*(2^{O(k)})$-FTP yet has no polynomial kernel under widely held complexity assumptions.

Another use of dynamic programming concernes optimization problem on graphs with inductively defined internal structure. One example of this is treewidth, but there are other width metrics like *cliquewidth, branchwidth,* and others.

Definition 4 (Robertson and Seymour [RS86a]).

(a) A tree-decomposition *of a graph $G = (V, E)$ is a tree $\mathcal{T}$ together with a collection of subsets T_x (called* bags*) of V labeled by the vertices x of $\mathcal{T}$ such that $\cup_{x \in \mathcal{T}} T_x = V$ and (i) and (ii) below hold:*
(i) For every edge uv of G there is some x such that $\{u, v\} \subseteq T_x$.
(ii) (Interpolation Property) If y is a vertex on the unique path in $\mathcal{T}$ from x to z then $T_x \cap T_z \subseteq T_y$.

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

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

The point of the notion is that it is a measure of how 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.

Figure 1 gives an example of a tree decomposition of width 2.

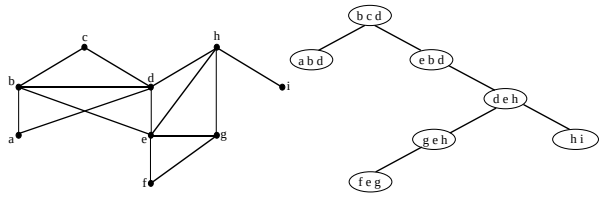

Fig. 1. Example of Tree Decomposition of Width 2

The point is that suppose you were seeking a dominating set in the graph of the example. Starting at the bag $\{f, e, h\}$, there are $2^3 = 8$ possible beginnings of a dominating set: $\emptyset, \{f\}, \{e\}, \{g\}, \{e, f\}, \{f, g\}, \{e, g\}, \{e, f, g\}$, that is, the subsets of $\{e, f, g\}$. These correspond to dominating sets of sizes, respectively 0,1,1,1,2,2,2,3. Now, if we move to the next bag, $\{g, e, h\}$, up the path we lose the vertex f and it will never appear again (because of the interpolation property). So that to figure out the possible dominating sets, we only need to remember to point at what we have done before, and the information flow across the boundary of the cut. Thus if we choose $\emptyset$ from the first bag, we must choose one of $\{e\}$, $\{g\}$, or $\{e, g\}$ from bag $\{g, e, h\}$, but choosing $\{e\}$ from bag $\{f, e, h\}$, would allow for $\emptyset$, for example, from $\{g, e, h\}$. Join bags are handled in a similar manner. Then one has to simply climb up the decomposition to the top and read off the minimum dominating set. The crucial thing is that the exponential part of running time of this process depends on the bag size, and the complexity of the definition of the property of the problem being analysed.

This vague statement can be made more formal using Courcelle's Theorem ([Co87]) which shows that any problem in "monadic second order counting logic" is linear time FPT for graphs on a fixed treewidth, and later work shows a similar statement for graphs of bounded "local treewidth" for first order properties. The key problem with these methods is that it can be hard to actually find the tree decomposition, though the problem is FPT by Bodlaender [Bod96]. Unfortunately, this algorithm is $O(2^{35t^3}|G|)$ for a fixed treewidth t, which is far from practical. It also seems unlikely that the problem of treewidth k has a polynomial kernel as argued in [BDFH09]. Moreover, assuming reasonable assumptions, it has been shown that the big towers of 2's in the constants for the running times obtained from Courcelle's Theorem cannot be gotten rid of. (Frick and Grohe [FrG02], also Flum and Grohe [FG04])

3 Parametric Intractability

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 red 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?

Assuming the rhetoric of the Cook-Levin Theorem, it seems hard to believe that Short Nondeterministic Turing Machine Acceptance could be in FPT, for example, solved in $O(|M|^3)$ for any path length k. Indeed, 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 5. *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].

Theorem 2. *The following are hard for* Short Nondeterministic Turing Machine Acceptance*:* Independent Set*,* Dominating Set*.*

In the tutorial, I plan to discuss the proof of Theorem 2.

A classic non-example of a parameterized reduction is the classical reduction of Sat to 3Sat. Early on Downey and Fellows conjectured that there is *no* parametric reduction of k-Dominating Set to k-Independent Set, and we see that the refined reduction of parameterized complexity theory give rise to several hierarchies based on logical depth. In particular, the W-hierarchy is based on weighted versions of satsifiabillity (i.e., the number of literals set to true) for a formula in $PoSoP...$ form or depth t. I will discuss the W-hierarchy below.

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

Here the 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. There are many, many problems hard for $W[1]$ and complete at many levels of this hierarchy. 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 a logical alternation. 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

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

in the language of graphs, 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 3 (Flum and Grohe [FG02a,FG04]). *The following hold:*

(i) $A[1] = W[1]$.
(ii) $A[t] \subseteq W[t]$.

Clearly $A[t] \subseteq XP$, but no other containment with respect to other classes of the W-hierarchy is known. There are also hierarchies like the M-hierarchy based around parameterizing the problem size.

As discussed in [Do03], this hierarchy is related to PTAS's; the connection of parameterized complexity with PTAS's going back to Bazgan [Baz95]. 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%. The idea is then to use $\frac{1}{\epsilon}$ as a parameter and show $W[1]$-hardness, say, for that parameterization. Then the problem cannot have a PTAS without $\frac{1}{\epsilon}$ (or worse) in the exponent. Thus it will be unlikely that any feasible PTAS can be found. More precise measurements can be obtained using the M-hierarchy, as we now see.

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 $k \log n$.
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]. The hypothesis FPT$\neq$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 4 (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 $\mathsf{FPT}=\mathsf{M}[1]$ (or, equivalently, unless ETH fails).*

We remark that Theorem 4 is optimal as all the problems above have been classified as $O^*(2^{O(\sqrt{k})})$ (see e.g. Downey and Thilikos [DTH11])

3.1 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 5 (Chen et. al [CCFHJKX05]). *The following hold:*

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

4 Other things

The other things I plan to discuss is how to use parameterized complexity to show that various methods such as logical metatheorems (such as Courcelle's Theorem and the Frick Grohe [FrG01] theorem on local treewidth) which make very large constants, cannot be improved assuming the W-hierarchy does nor collapse. Also I will look at recent work on lower bounds for kernelization such as [BDFH09,CFM11,FS11,BTY08,HKSWWta]. This is quite exciting work which allows us to demonstrate that the kernelization technique cannot be used to give polynomial sized kernels.

I won't have time to discuss the use of parameterized complexity to rule out various techniques in other areas such as Aleknovich and Razborov [AR01].

5 Left out

Clearly in 2 lectures I won't have time to deal with things in a lot of depth. I have alose left out a lot. 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 6 (Flum and Grohe [FG02b]). *Counting the number of cycles of size k in a bipartite graph is $\#W[1]$-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].

There are many other important topics such as implementations, connections with exact algorithms, connections with classical complexity and the like. Time and space limitations preclude this material being included.

References

[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.

[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.

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

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

[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.

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

[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.

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

[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

[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.

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

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

[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.

[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.

[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

[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.

[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.

[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.

[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.

[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.

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

[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,

[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.

[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.

[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.

[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.