From the W-hierarchy to XNLP

Classes of Fixed Parameter Intractability

Hans L. Bodlaender(✉)

Department of Information and Computing Sciences,
Utrecht University, Utrecht, The Netherlands
h.l.bodlaender@uu.nl

Abstract. In this short survey, a number of old and new notions from parameterized complexity are discussed. We start with looking at the W-hierarchy, including the classes $W[1]$, $W[2]$, $W[P]$. Then, a recent development where problems are shown to be complete for simultaneously non-deterministic time of the form $f(k)n^c$ and space of the form $f(k)\log n$, is discussed. Some consequences and other notions are briefly explored.

Keywords: Parameterized complexity · W-hierarchy · XP · XNLP

1 Introduction

The study of parameterized algorithms and complexity starts at the insight that many computationally hard problems become easier when a parameter of the input can be assumed small. Suppose we are to solve a facility location problem, e.g., we have to place as few as possible fire stations in a city, such that each house in the city is at most a 15 min drive away from a fire station. It is not hard to observe that this is an NP-hard problem. However, if we know that we have only funds available for three fire stations, then an exhaustive search for all possible combinations of at most three locations gives a tractable (polynomial time) algorithm to solve the problem.

In the theory of parameterized algorithms and complexity, we look at parameterized problems; i.e., we identify some aspect of the input as the parameter. Then we ask: when this parameter is a constant, is there a polynomial time algorithm. And if so, does the degree of the polynomial depend on the parameter. The theory started with work by Fellows and Langston at the late 1980s (e.g., [21,22], with some central notions first identified by Abrahamson et al. in 1989 [2], and much foundational work done in the 1990s by Downey and Fellows (e.g., [14–16] and [17].)

Throughout this paper, we view a parameterized problem as a subset of $\Sigma^* \times \mathbb{N}$, with Σ some finite alphabet. We are interested in the algorithmic complexity of parameterized problems for which its 'classic' variant (i.e., where the parameter is just part of the input) is intractable, e.g., NP-hard. Many parameterized problems fall in one of the following three categories (in order of increasing desirability):

P. Mutzel et al. (Eds.): WALCOM 2022, LNCS 13174, pp. 15–25, 2022.
https://doi.org/10.1007/978-3-030-96731-4_2

- There is a value of the parameter for which the problem is NP-hard. E.g., if we consider Graph Colouring where the number of colours is the parameter, then this problem is NP-hard with the parameter (number of colours) equal to 3 [27]. Parameterized problems which are NP-hard for some fixed value of the parameter are called *para-NP-hard*.
- There is an algorithm that solves the problem for inputs of the form (x, k) in $O(n^{f(k)})$ time, where $n = |x|$ is the size of the input, k the parameter, and f a (computable) function. The class of problems with such an algorithm is called XP.
- There is an algorithm that solves the problem for inputs of the form (x, k) in $O(f(k)n^c)$ time, with again $n = |x|$ the size of the input, k the parameter, f a computable function, and c a constant. Problems of this type are called *fixed parameter tractable*, and the class of such problems is called FPT.

One can distinguish different flavours of FPT (and XP), namely *non-uniform* (for each value of k, there is an algorithm of the stated running time), *uniform* (there is one algorithm working for all values of k, but we do not require that f is computable), and *strongly uniform* (as above: we have one algorithm for all values of k, and f is computable). Examples of non-uniform fixed parameter tractability can be obtained with help of well quasi orderings: if we have a graph parameter h which cannot increase by taking a minor of a graph, then from Robertson-Seymour graph theory, we obtain a non-constructive proof tells us that for each k, there is an $O(n^2)$ algorithm that decides for a given graph if $h(G) \leq k$. (See e.g., [13, Section 6.3] with [29].) But, we may not be able to construct the algorithms and thus only know that for each k there exists a (separate) algorithm. See the discussion in [17, Chapter 19]. In the remainder, we only look at strongly uniform cases.

NP-completeness theory tells us when a problem is para-NP-hard. Assuming P≠NP, para-NP-hard problems do not belong to XP (or FPT). Thus an NP-hardness result for a specific value of a parameter gives evidence that the problem at hand is not likely to belong to XP. To give similar evidence to tell for studied problems that they are not fixed parameter tractable, a number of complexity classes have been introduced, all which are assumed to be not a subset of or equal to FPT. Thus, hardness of a problem for such a class tells that it is unlikely that the problem belongs to FPT.

For a few problems, an unconditional proof that they do not belong to XP is known. Diagonalisation gives that FPT is a proper subset of XP [17, Proposition 27.1.1]. A few problems (formulated in terms of games) are known to be XP-complete [3, 4], and thus, these cannot belong to FPT. (See also [18, Chapter 27].)

This short (and incomplete) survey reviews a number of classes of problems assumed not to be fixed parameter tractable, with some classic results from the field, and some recent developments. The theory in this field is rich (much richer than this survey can show); the focus in this short survey is on classes that contain complete problems that are studied in combinatorial optimisation algorithms. Much information can also be found in a number of excellent text books that on

parameterized algorithms and complexity [13,17,18,24,31] and on the topic of kernelization (a subtopic in the field, not discussed in this survey) [26].

2 Reductions

Hardness and completeness for classes is as usual defined with help of reductions between problems.

A *parameterized reduction* from parameterized problem Q to parameterized problem R is an algorithm A that maps inputs for Q to inputs for R, such that $(x,k) \in Q \Leftrightarrow A((x,k)) \in R$, A uses time $f(k)n^c$ for a computable function f and constant c, and if $A(x,k) = (x',k')$, then $k' \leq g(k)$ for a computable function g.

We also look at *parameterized logspace reductions* (or pl-reductions), where we additionally require that the space used by the reduction algorithm is $O(h(k) + \log n)$. Different types of reductions also are used in the field of parameterized complexity, but these will not be discussed here.

3 The *W*-hierarchy

Downey and Fellows (see e.g. [17]) have introduced the W-hierarchy: a hierarchy of complexity classes of parameterized problems. The hierarchy contains a class $W[i]$ for each positive integer i, the class $W[SAT]$, and the class $W[P]$. Together with FPT and XP, we have the following inclusions.

$$FPT \subseteq W[1] \subseteq W[2] \subseteq W[3] \cdots \subseteq W[SAT] \subseteq W[P] \subseteq XP$$

It is conjectured [17, Chapter 12] that each inclusion is proper. In particular, when $FPT = W[1]$, then the Exponential Time Hypothesis would not hold [10].

$W[i]$ is defined with help of combinatorial circuits. Consider a circuit, with n Boolean input gates, and one output node. Take some fixed constant c. (The choice of c does not matter for the results, e.g., we can set $c = 2$.) The *weft* of the circuit is the maximum number of internal nodes with indegree more than c on a path from an input gate to the output node. Now, $W[i]$ is defined as the parameterized problems with a parameterized reduction to the problem to decide for a given circuit, if we can set k input gates to true and all other input gates to false, such that the circuit outputs true. (k is the parameter of the problem.)

An alternative definition, that can be of help to prove $W[i]$-hardness, is in terms of Boolean formulas. Consider a formula on n Boolean variables. We say the formula is i-normalised, if its is the conjunction of the disjunction of the conjunction of ... of literals, with i alternations between conjunction and disjunction. $W[i]$ can also be defined as the problems with a parameterized reduction to the problem to decide if a given i-normalised formula can be satisfied by setting exactly k variables to true, and all others to false. Again, k is the parameter. (Equivalently, we can ask to set at most k variables to true.)

For $i = 1$, we obtain the $W[1]$-complete problem, for each fixed integer q, Weighted q-CNF Satisfiability. Given is a Boolean formula in Conjunctive Normal Form, with each clause having at most q literals, and we ask if we can satisfy it by setting exactly k (the parameter) variables to true.

Important examples of complete problems are Clique and Independent Set, who are $W[1]$-complete, and Dominating Set, which is $W[2]$-complete.

As we assume that the hierarchy is proper, this implies that it is unlikely that Clique, Independent Set, and Dominating Set are in FPT.

Intuition why Clique and Independent Set are in $W[1]$, while Dominating Set is not, is the following. We can take an input gate (or a Boolean variable) for each vertex of the graph, which is true iff the vertex is in the solution set. To verify that this set forms a independent set, we need to perform a polynomial number of tests (one for each pair of nonadjacent vertices), where each such test looks at two variables (checking that at least one of these is false)—this corresponds to a circuit of weft one. (The same type of argument works for Clique.) To verify that we have a dominating set, we need to perform a polynomial number of tests (one for each vertex), but each of these tests can involve a large number of variables (we check that the vertex is dominated, thus need to look at the variables of the vertex and its neighbours)—this corresponds to a circuit of weft two.

$W[SAT]$ is defined in the same manner as the classes $W[i]$, but now we can use any Boolean formula of polynomial size, and $W[P]$ is defined with combinatorial circuits of polynomial size (without weft restrictions).

In the parameterized algorithms and complexity literature, a large number of problems from various applications have been shown to be hard or complete for classes in the W-hierarchy. In particular, the classes $W[1]$ and $W[2]$ play an important role.

4 Logarithmic Space

4.1 The Story of Bandwidth

There are several problems that are shown to be hard for $W[1]$, for $W[2]$, or for all classes $W[i]$ for all integers $i \in \mathbb{N}$, but which are not known to be member of $W[P]$, i.e., we do not know whether they belong to a class in the W-hierarchy.

As a central example, we look at the Bandwidth problem. Given here is an undirected graph $G = (V, E)$, and the integer parameter k, and we want to decide if there is a bijective function $f : V \to \{1, \ldots, |V|\}$, such that for all edges $\{v, w\} \in E$, $|f(v) - f(w)| \leq k$. Bandwidth is a long studied problem—amongst others, because it is equivalent to asking for a symmetric matrix whether we can permute rows and columns simultaneously, such that all non-zero elements are at a band of width k around the main diagonal.

Already in 1980, Saxe [34] showed that Bandwidth can be solved in $O(n^{k+1})$ time, thus belongs to XP; this was later improved to $O(n^k)$ by Gurari and Sudborough [28]. In 1994, Bodlaender et al. [7] claimed that Bandwidth was

hard for all classes $W[i]$, $i \in \mathbb{N}$, but it took till 2020 till a proof of this fact was written down [5]. In 2014, Dregi and Lokshtanov [19] showed that Bandwidth is W[1]-hard for trees of pathwidth at most two.

Each of these results showed *hardness* for classes in the W-hierarchy, but membership. This gives the question: is Bandwidth member of a class in the W-hierarchy, and can we find a class for which this problem is complete? The same question can be asked for many other problems, that are known to be hard for $W[1]$, but not known to reside in the W-hierarchy.

In the midst of the 1990s, Hallett gave an argument why it is unlikely that Bandwidth belongs to $W[P]$; the argument is discussed in [23]. The argument is as follows: certificates for problems in $W[P]$ have size $O(k \log n)$: we use $\log n$ bits for each of the k input gates that is true to give its index. However, one expects that Bandwidth cannot have such small certificates; for instance, we can have a graph with many connected components; one would expect to need certificates of size at least (but probably much larger than) the number of connected components. The argument resembles the later development of compositionality arguments for showing lower bounds for kernels [6].

So, if Bandwidth is not (likely) in $W[P]$, where is it?

We can go back to the first dynamic programming algorithm by Saxe [34] for Bandwidth. In this algorithm, we build n tables: each table entry of the i table gives 'essential information' of an ordering f of a set S with i vertices. The essential information gives all that is needed to remember of f and S to later determine if there is an ordering of V that starts with f, and then gives the vertices in $V \setminus S$ in some order. A simpler (slower) algorithm is obtained by taking as essential information the last $2k$ vertices of S with their order. One can turn this dynamic programming algorithm into a non-deterministic algorithm, where we do not store all elements of a table, but just guess one entry. We then have the following, non-deterministic algorithm for Bandwidth: repeatedly guess the next vertex in the order, and keep in memory the last $2k$ vertices. (We need to check that we never guess a vertex that is already ordered, but this verification can be done with a dfs search with help of the $2k$ stored vertices; we leave the details as a simple puzzle for the reader.)

What we now have is a non-deterministic algorithm for Bandwidth; the algorithm uses polynomial time, n non-deterministic guesses of a vertex, $O(k \log n)$ memory (as we remember $O(k)$ vertices with order).

In 2015, Elberfeld et al. [20] introduced a number of different classes of parameterized problems, including several subclasses of FPT and of XP, characterising the use of time, space, size of kernels, and more. One of these subclasses is the class, which we call here XNLP (and was called $N[fpoly, flog]$ in [20]). XNLP is the class of parameterized problems that can be recognized by a non-deterministic algorithm that simultaneously use $O(f(k)n^c)$ time and $O(f(k) \log n)$ memory, with f a computable function, and c a constant.

The non-deterministic algorithm for Bandwidth sketched above shows that it belongs to this class XNLP. Interestingly, it is possible to show that

Bandwidth is XNLP-complete [8]. For XNLP-completeness, we need to use parameterized logspace reductions.

4.2 XNLP-complete Problems

To show that problems are XNLP-hard, we use parameterized logspace reductions from known XNLP-hard problems. Recently, several problems have been shown to be XNLP-complete [8,20]. Several need a chain of reductions. Useful intermediate XNLP-complete problems are, amongst others:

- Timed Accepting Non-deterministic Linear Cellular Automaton [20]. We have a linear cellular automaton: a row of k cells, each having at each time step a value (state) from an alphabet (which can be of linear size, so we use $O(\log n)$ bits to denote an element from the alphabet). At each time step, each cell receives a value, non-deterministically depending on its value at that of the neighbouring cell(s). One state is said to be accepting, and the question is whether there exists a run where after t (given in unary) time steps, a cell has the accepting state. k is the parameter.
- Chained Weighted CNF-Satisfiability [8]. We have n sets of Boolean variables $X^1, \ldots, X^n$, each of size r, and a Boolean formula in Conjunctive Normal Form F, and a parameter k. The question is to set of each set X^i exactly k variables to true, and all others to false, such that the following formula is satisfied:

  $$\wedge_{1 \leq i < n} F(X_i, X_{i+1})$$

  Several special cases are also shown to be XNLP-complete in [8]. The hardness proof is of a similar vein as the Cooks proof of the NP-hardness of Satisfiability [12]: the logic formula describes the working of the automaton.
- Chained Clique [8]. Given is a graph $G = (V, E)$, where V is partitioned into n subsets $V_1, \ldots, V_n$, and the parameter $k \in \mathbb{N}$. Question is whether we can choose from each set V_i a subset $S_i \subseteq V_i$ of k vertices, such that for each pair of successive sets, $S_i \cup S_{i+1}$ $(1 \leq i < n)$ forms a clique of size $2k$.
- Accepting NNCC Machine [8]. The XNLP-completeness of the problem whether the following non-deterministic machine has an accepting run has been proven to be a very helpful tool to show several problems XNLP-hard. The machine has k integer counters, which start at 0. At each time step, all counters can be increased non-deterministically to an integer that is at most n. There is a series of tests: each test looks at two counters, and has two integers from $[0, n]$. If the first counter equals the first of these integers, and the second counter equals the second integer, then the machine halts and rejects. If all tests succeed, the machine accepts.

From the XNLP-hardness of Accepting NNCC Machine, we can (with one intermediate step) obtain the XNLP-hardness of Bandwidth, but also XNLP-hardness of Scheduling with Precedence Constraints, parameterized by the number of machines and thickness. Other XNLP-complete problems include Longest Common Subsequence [20], List Colouring with the pathwidth

of the graph as parameter [8], and Independent Set and Dominating Set on graphs of pathwidth $k \log n$, where again k is the parameter.

XNLP-completeness has two interesting consequences. First, it implies hardness for all classes $W[i]$ for all $i \in \mathbb{N}$. Interestingly, often the XNLP-hardness proofs are easier than the earlier proofs of $W[i]$-hardness for all i. Second, a conjecture of Pilipczuk and Wrochna [32] for Longest Common Subsequence implies the same conjecture for all XNLP-hard problems.

Conjecture 1 (Pilipczuk and Wrochna [32]). Suppose parameterized problem Q is XNLP-hard. Then Q has no algorithm that runs in $n^{f(k)}$ time and $f(k)n^c$ space, for a computable function f and constant c, with k the parameter, and n the total input size.

XNLP is a subset of XP (instead of making non-deterministic guesses, we tabulate all reachable states of the memory), but from Conjecture 1, we obtain that it is unlikely that an XNLP-complete problem has an XP algorithm that uses little space ('fpt space').

4.3 Other Classes with Logarithmic Space and Reconfiguration

Well known in classic complexity theory are the classes L and NL: problems solvable in logarithmic spacew with a deterministic, respectively non-deterministic algorithm. An interesting class is SL (with the S an abbreviation of *'symmetric'*), which allows to 'reverse' computations. Reingold [33] showed that L=SL, which is used in a result discussed below.

The parameterized counterparts of L and NL are respectively XL (parameterized problems solvable in $f(k) \log n$ space), and XNL (parameterized problems solvable with a non-deterministic algorithm in $f(k) \log n$ space). See e.g., [11].

Recently, Bodlaender et al. [9] explored the complexity of Independent Set and Dominating Set reconfiguration. Given are two sets S_1 and S_2, which are both independent sets of G (or, respectively, both dominating sets). We want to change S_1 into S_2 in a number of moves, where each move changes one vertex of the set to another one, while each intermediate set still must be an independent (or dominating) set. We look at the problem if such a move sequence exists, or such a move sequence with t moves exists. The sizes $|S_1| = |S_2|$ are the parameter of the problem. The complexities of these questions depend on whether t is not given, a second parameter[1], given in unary, or given in binary. Table 1 summarises the different results. The XL-completeness for the case where there is no bound on the number of moves uses an interesting argument, with the following intuition: when we can use arbitrary many moves, we can always reverse any move. That corresponds (via the reductions) to a computation on a symmetric Turing Machine, which yields XSL-completeness, where XSL is the parameterized counterpart of SL. But, by Reingolds result [33], SL = L, which implies XSL = XL, thus the problems without a bound on the number of moves are XL-complete.

[1] Formally, instead of giving a problem two parameters, we can take the sum of these two values as parameter.

Table 1. Complexity of reconfiguration problems, with set sizes as (one of the) parameter(s)

Nb of steps	Independent Set	Dominating Set	References
parameter	$W[1]$-complete	$W[2]$-complete	[9,30]
unary	XNLP-complete	XNLP-complete	[8]
binary	XNL-complete	XNL-complete	[9]
not bounded	XL-complete	XL-complete	[9]

5 Other Classes of Hard Parameterized Problems

There are a several other important classes of parameterized problems, which are assumed not to be fixed parameter tractable. The following brief overview mentions just a few of these, and is far from complete.

The A-hierarchy. Flum and Grohe [25] introduced the ***A*-hierarchy**: parameterized equivalences of the classes in the polynomial time hierarchy. The hierarchy contains classes $A[1], A[2], \ldots$ While $A[1] = W[1]$, classes higher in the hierarchy contain their W-counterparts as (likely proper) subsets. We will not give the formal definitions here; intuitively, each level in the A-hierarchy adds one alternation between universal and existential quantification.

One such alternation can be seen in the Clique Dominating Set problem. Given is an undirected graph $G = (V, E)$, and integers k and ℓ, which both are parameters of the problem. (Or, more precisely, we take $k + \ell$ as parameter.) The question is if there is a set S of k vertices that dominates all cliques with ℓ vertices. (I.e., for every clique C of size ℓ, C contains a vertex that is in S or has a neighbour in S.) Clique Dominating Set is an example of an $A[2]$-complete problem [25, Theorem 8.20].

The AW-hierarchy. Alternation is also a key element in the classes defined in the AW-hierarchy [1], see also [17, Chapter 14]. The classes can be defined with help of weighted variants of Quantified Boolean Formulas. Several complete problems for these classes are defined in terms of combinatorial games, where the problem is whether there is a winning strategy for the first player in a given position in at most k moves, where this number of moves k is taken as parameter.

Counting Problems. Let us now consider counting problems, i.e., we want to determine the number of solutions to a problem. In classic complexity theory, many complexity classes have counting variants, with $\#P$ (the class which map the input to the number of accepting paths of a non-deterministic Turing Machine) of central importance. Typical $\#P$-complete problems are: given a Boolean formula, how many satisfying truth assignments does it have; given a graph, how many Hamiltonian circuits does it have? Flum and Grohe [24] introduced parameterized classes for counting problems, including counting variants of the classes in the W-hierarchy.

An interesting example of a $\#W[1]$-complete problem is that of counting the number of paths of length k in a given graph G; k is again the parameter [24]. In contrast, deciding if there is at least one path of length k is fixed parameter tractable. The difference between the complexity of deciding and counting can here be explained by the fact that negative inputs (graphs that do not have a path of length k) have a special structure (e.g., they have treewidth at most k), and such structure can be exploited algorithmically. In contrast, when counting we cannot make assumptions on the graph's structure.

6 Conclusions

In the study of parameterized algorithms, many parameterized problems are known to be hard for a complexity class that is assumed not to be a subset of FPT, and thus, are believed not to be in FPT. For a subset of these problems, completeness for a parameterized class is known. Still, there are many problems where only hardness for some class has been proved, but membership in that class is not known, and sometimes not expected.

Thus, the situation is much less clear than in classic NP-completeness theory. There, for many problems, both NP-hardness and membership in NP is known. The parameterized counterparts of those problems often reside in different classes, and their precise complexity in the hierarchies has often not yet been established. To gain more understanding and give precise characterisations of the parameterized complexity of well known combinatorial problems gives a large number of intriguing open problems. The discussion on XNLP shows that such results can have wider consequences, e.g., give more information on the use of additional resources like memory by the algorithms.

References

1. Abrahamson, K.A., Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness IV: On completeness for $W[P]$ and PSPACE analogues. Ann. Pure Appl. Logic **73**, 235–276 (1995)
2. Abrahamson, K.R., Ellis, J.A., Fellows, M.R., Mata, M.E.: On the complexity of fixed-parameter problems. In: Proceedings of the 30th Annual Symposium on Foundations of Computer Science, FOCS 1989, pp. 210–215 (1989)
3. Adachi, A., Iwata, S., Kasai, T.: Classes of pebble games and complete problems. SIAM J. Comput. **8**(4), 576–586 (1979)
4. Adachi, A., Iwata, S., Kasai, T.: Some combinatorial game problems require $\Omega(n^k)$ time. J. ACM **31**(2), 361–376 (1984)
5. Bodlaender, H.L.: Parameterized complexity of Bandwidth of caterpillars and Weighted Path Emulation. In: Kowalik, Ł, Pilipczuk, M., Rzążewski, P. (eds.) WG 2021. LNCS, vol. 12911, pp. 15–27. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86838-3_2
6. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. J. Comput. Syst. Sci. **75**, 423–434 (2009)

7. Bodlaender, H.L., Fellows, M.R., Hallett, M.: Beyond NP-completeness for problems of bounded width: Hardness for the W hierarchy. In: Proceedings of the 26th Annual Symposium on Theory of Computing, STOC 1994, pp. 449–458. ACM Press, New York (1994)
8. Bodlaender, H.L., Groenland, C., Nederlof, J., Swennenhuis, C.M.F.: Parameterized problems complete for nondeterministic FPT time and logarithmic space. arXiv abs/2105.14882 (2021). https://arxiv.org/abs/2105.14882. To appear in proceedings FOCS 2021
9. Bodlaender, H.L., Groenland, C., Swennenhuis, C.M.F.: Parameterized complexities of dominating and independent set reconfiguration. In: Golovach, P.A., Zehavi, M. (eds.) 16th International Symposium on Parameterized and Exact Computation, IPEC 2021. LIPIcs, vol. 214, pp. 9:1–9:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.IPEC.2021.9
10. Chen, J., Huang, X., Kanj, I.A., Xia, G.: Strong computational lower bounds via parameterized complexity. J. Comput. Syst. Sci. **72**, 1346–1367 (2006)
11. Chen, Y., Flum, J., Grohe, M.: Bounded nondeterminism and alternation in parameterized complexity theory. In: 18th Annual IEEE Conference on Computational Complexity (Complexity 2003), Aarhus, Denmark, 7–10 July 2003, pp. 13–29. IEEE Computer Society (2003). https://doi.org/10.1109/CCC.2003.1214407
12. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual Symposium on Theory of Computing, STOC 1971, pp. 151–158. ACM, New York (1971)
13. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
14. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness III: Some structural aspects of the W hierarchy. In: Ambos-Spies, K., Homer, S., Schöning, U. (eds.) Complexity Theory, pp. 191–226. Cambridge University Press, Cambridge (1993)
15. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness I: Basic results. SIAM J. Comput. **24**, 873–921 (1995)
16. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness II: On completeness for $W[1]$. Theoret. Comput. Sci. **141**, 109–131 (1995)
17. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, New York (1999). https://doi.org/10.1007/978-1-4612-0515-9
18. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. TCS, Springer, London (2013). https://doi.org/10.1007/978-1-4471-5559-1
19. Dregi, M.S., Lokshtanov, D.: Parameterized complexity of bandwidth on trees. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014. LNCS, vol. 8572, pp. 405–416. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43948-7_34
20. Elberfeld, M., Stockhusen, C., Tantau, T.: On the space and circuit complexity of parameterized problems: classes and completeness. Algorithmica **71**(3), 661–701 (2014). https://doi.org/10.1007/s00453-014-9944-y
21. Fellows, M.R., Langston, M.A.: Nonconstructive advances in polynomial-time complexity. Inf. Process. Lett. **26**, 157–162 (1987)
22. Fellows, M.R., Langston, M.A.: Nonconstructive tools for proving polynomial-time decidability. J. ACM **35**, 727–739 (1988)
23. Fellows, M.R., Rosamond, F.A.: Collaborating with Hans: Some remaining wonderments. In: Fomin, F.V., Kratsch, S., van Leeuwen, E.J. (eds.) Treewidth, Kernels, and Algorithms. LNCS, vol. 12160, pp. 7–17. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-42071-0_2

24. Flum, J., Grohe, M.: The parameterized complexity of counting problems. SIAM J. Comput. **33**(4), 892–922 (2004). https://doi.org/10.1137/S0097539703427203
25. Flum, J., Grohe, M.: Parameterized Complexity Theory. TTCSAES, Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-29953-X
26. Fomin, F., Loksthanov, D., Saurabh, S., Zehavi, M.: Kernelization - Theory of Parameterized Preprocessing. Cambridge University Press, Cambridge (2019)
27. Garey, M.R., Johnson, D.S.: Computers and Intractability, A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
28. Gurari, E.M., Sudborough, I.H.: Improved dynamic programming algorithms for bandwidth minimization and the mincut linear arrangement problem. J. Algorithms **5**, 531–546 (1984)
29. Kawarabayashi, K., Kobayashi, Y., Reed, B.A.: The disjoint paths problem in quadratic time. J. Comb. Theory Ser. B **102**(2), 424–435 (2012). https://doi.org/10.1016/j.jctb.2011.07.004
30. Mouawad, A.E., Nishimura, N., Raman, V., Simjour, N., Suzuki, A.: On the parameterized complexity of reconfiguration problems. Algorithmica **78**(1), 274–297 (2016). https://doi.org/10.1007/s00453-016-0159-2
31. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and Its Applications, Oxford University Press, Oxford (2006)
32. Pilipczuk, M., Wrochna, M.: On space efficiency of algorithms working on structural decompositions of graphs. ACM Trans. Comput. Theory **9**(4), 18:1-18:36 (2018). https://doi.org/10.1145/3154856
33. Reingold, O.: Undirected connectivity in log-space. J. ACM **55**(4), 1–24 (2008)
34. Saxe, J.B.: Dynamic programming algorithms for recognizing small-bandwidth graphs in polynomial time. SIAM J. Algebraic Discrete Methods **1**, 363–369 (1980)