The Complexity Ecology of Parameters: An Illustration Using Bounded Max Leaf Number

Michael Fellows[1], Daniel Lokshtanov[2], Neeldhara Misra[3], Matthias Mnich[4], Frances Rosamond[1] and Saket Saurabh[2]

[1] University of Newcastle, Callaghan, Australia
{michael.fellows,frances.rosamond}@newcastle.edu.au
[2] University of Bergen, Norway
{daniel.lokshtanov,saket.saurabh}@ii.uib.no
[3] The Institute of Mathematical Sciences, Chennai 600113, India
neeldhara@imsc.res.in
[4] Technical University of Eindhoven, The Netherlands
m.mnich@tue.nl

Abstract. In the framework of parameterized complexity, exploring how one parameter affects the complexity of a different parameterized (or unparameterized problem) is of general interest. A well-developed example is the investigation of how the parameter *treewidth* influences the complexity of (other) graph problems. The reason why such investigations are of general interest is that real-world input distributions for computational problems often inherit structure from the natural computational processes that produce the problem instances (not necessarily in obvious, or well-understood ways). The *max leaf number* $ml(G)$ of a connected graph G is the maximum number of leaves in a spanning tree for G. Exploring questions analogous to the well-studied case of treewidth, we can ask: how hard is it to solve 3-Coloring, Hamilton Path, Minimum Dominating Set, Minimum Bandwidth or many other problems, for graphs of bounded *max leaf number*? What optimization problems are $W[1]$-hard under this parameterization? We do two things:
(1) We describe much improved FPT algorithms for a large number of graph problems, for input graphs G for which $ml(G) \leq k$, based on the *polynomial-time extremal structure theory* canonically associated to this parameter. We consider improved algorithms both from the point of view of kernelization bounds, and in terms of improved fixed-parameter tractable (FPT) runtimes $O^*(f(k))$.
(2) The way that we obtain these concrete algorithmic results is general and systematic. We describe the approach, and raise programmatic questions.

1 Introduction

The analysis of the complexity of problems, for graphs of bounded treewidth, is well-developed and supports many systematic approaches that have developed over a number of years [Cou90,ALS91,Bod96,DF99,Nie06,BK07]. For example, determining whether a graph is 3-colorable can be solved in time $O(n)$

for graphs of treewidth at most k. In the terminology of parameterized complexity [DF99,FG06,Nie06], Graph 3-Coloring is *fixed parameter tractable* for the parameter *treewidth*. In this small example, the asymptotic notation conceals serious costs associated to the treewidth bound k, from two sources:
(1) The complexity of computing a tree-decomposition of width k is $O(2^{35k^3}n)$ for an n-vertex graph.
(2) Once the tree-decomposition is obtained, one would then solve the problem by dynamic programming, in time $O(3^k n)$.

Suppose that we wish to solve Graph 3-Coloring for graphs having a different structural restriction — how should this be done? Here we consider the structural parameter of *bounded max leaf number*, $ml(G)$, where this is defined for a connected graph G as the maximum number of leaves of a spanning tree for G. (We choose this parameter mainly to illustrate the key issues, and because enough is known of the associated polynomial-time extremal structure theory to provide a good example of the general approach. We are not aware of any strong direct applications of bounded max leaf number for natural input distributions.)

One way to approach the problem of determining 3-colorability, parameterizing by max leaf number, is to note that graphs of bounded max leaf number exclude a tree minor and therefore have bounded pathwidth, so that the above-sketched bounded treewidth approach can be used. This classifies Graph 3-Coloring, parameterized by *max leaf number*, as FPT, but this is not an efficient algorithm.

We have two main objectives in this paper:
(1) We describe *efficient* FPT algorithms for Graph 3-Coloring and many other graphs problems, for input G parameterized by a bound $ml(G) \leq k$. We consider such FPT algorithms from both the exponential complexity $O^*(f(k))$ and polynomial-time kernelization points of view.
(2) We do so in a way that is generally systematic, and that "fits" the study of how parameterized structure affects computational complexity in what we term the "ecology" of parameterized complexity.[5]

In the next section, we discuss the basics of parameterized complexity and motivate the general setting for this investigation.

2 Background on Parameterized Complexity and the Complexity Ecology of Parameters

Parameterized complexity is a special case of what one might call a "multivariate" approach to complexity analysis and algorithm design. Here, in addition to the overall input size n, a secondary measurement k (the parameter) is also considered, where one expects the parameter k to be significantly smaller than n and to capture information about the structure of typical inputs or other aspects of the problem situation that affect computational complexity. In the

[5] One can view this issue as a kind of *generalized bidimensionality theory* in the sense of Demaine and Hajiaghayi [DFHT05,DH05,DH07].

familiar "classical" one-dimensional approach, the central concept is *polynomial time* (P) ("the good class"). In the parameterized complexity framework the central notion is *fixed-parameter tractability* (FPT), defined to be solvability in time $f(k)n^c$, where f is some function (unrestricted), and c is a constant. In the classical framework, an algorithm with running time in P is the desirable outcome, as contrasted with the possibility that only running times of the form $O(2^{poly(n)})$ (the "bad class") might be achievable. Classical complexity analysis unfolds in the contrast between these two univariate function classes.

Parameterized complexity analysis unfolds analogously in the contrast between the "good class" of bivariate functions FPT, and the "bad class" of runtimes[6] of the form $O(n^{g(k)})$. To emphasize the contrast, one could also consider defining FPT *additively* as solvability in time $f(k) + n^c$. It turns out that this makes no difference qualitatively[7]: a parameterized problem is *additively FPT* if and only if it is FPT by the usual definition [DFS99,DF99]. The basic contrast in parameterized complexity is thus concerned with whether any exponential costs of the problem can be confined to the parameter.

In the classical framework, evidence that a problem is unlikely to have an algorithm with a runtime in the good class is given by determining that it is NP-hard, PSPACE-hard, EXP-hard, etc. In parameterized complexity analysis there are analogous means to show likely parameterized intractability. The current tower of the main parameterized complexity classes is:

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

Parameterized by the size k of a solution, the familiar Independent Set problem is complete for $W[1]$ [DF95a], and Dominating Set is complete for $W[2]$ [DF95b]. The naturally parameterized Bandwidth problem is hard for $W[t]$ for all t [BFH94]. The best known algorithms for the parameterized Independent Set and Dominating Set problems are slight improvements on the brute-force approach of trying all k-subsets, and run in time $O(n^{O(k)})$ [NP85]. The parameterized class $W[1]$ is strongly analogous to NP, because the k-Step Halting Problem for Turing machines of unlimited nondeterminism is complete for $W[1]$ [DFHKW94,CCDF97]. FPT is equal to $M[1]$ if and only if the so-called *Exponential Time Hypothesis* fails [IP01,DEF+03]. There is an algorithm for the k-Independent Set problem that runs in time $O(n^{o(k)})$ if and only if FPT is equal to $M[1]$, and there is an algorithm for the k-Dominating Set problem that runs in time $O(n^{o(k)})$ if and only if FPT is equal to $M[2]$ [CCF+06].

There are numerous useful recent surveys about parameterized complexity and algorithm design [Ra97,DFS99,Fe02,Fe03,Nie04,GN07,CJ08]; one can also turn to the books and monographs [DF99,FG06,Nie06] for further background.

Major motivation for the subject of parameterized complexity and algorithmics has come from the graph minors project of Robertson and Seymour [RS85].

[6] Solvability in such time defines the parameterized complexity class XP.

[7] However, quantitatively, in classifying a parameterized problem as additively FPT, it might be necessary to use a "larger" function $f(k)$.

The parameterized problem Graph Minor takes as input graphs G and H and asks whether H is a minor of G (that is, whether a graph isomorphic to H can be obtained from G by contracting edges of a subgraph of G). This is a fundamental problem, naturally parameterized by H. To show that this problem is FPT (according to present knowledge) *requires* the entire panoply of the graph minors structure theory.

The following well-known lemma codifies how every FPT parameterized problem has a canonically associated structure theory project, *via* the quest for efficient FPT kernelization bounds.

Lemma 1. *A parameterized problem Π is in FPT if and only if there is a transformation from Π to itself, and a function g, that reduces an instance (x,k) to (x',k') such that:*
(1) the transformation runs in time polynomial in $|(x,k)|$,
(2) (x,k) is a yes-instance of Π if and only if (x',k') is a yes-instance of Π,
(3) $k' \leq k$, and
(4) $|x'| \leq g(k)$.

We say that we have a *kernelization bound* of $g(k)$ in the situation described. The proof is completely trivial, yielding a kernelization bound of only $g(k) = f(k)$ for a parameterized problem solvable in FPT time $f(k)n^c$. However, the shift of perspective that the Lemma codifies is useful and important.

With regards to improved kernelization bounds ("better $g(k)$") we can often achieve strikingly non-exponential bounds, and the polynomial-time "preprocessing" routines that produce small kernels have proven practical value [ACFLSS04,Nie04,Nie06,Wei98]. The Vertex Cover problem can be kernelized in polynomial time to a graph on at most $2k$ vertices [NT75,ACFLSS04,CFJ04]. Planar Dominating Set also has a problem kernel of linear size [AFN04]. The Undirected Feedback Vertex Set problem was recently shown to have a polynomial-sized P-time kernelization [BEF+06], subsequently improved to a kernelization bound of $O(k^3)$ [Bod07]. For a useful recent survey of FPT kernelization see [GN07].

In our results in this paper we address the question of efficient algorithms for solving various problems on graphs of bounded max leaf number, from both the "better $f(k)$" and "better kernelization" points of view, a dual perspective that is now standard in parameterized algorithmics [Nie06].

2.1 A Complexity Ecology of Parameters

A striking fact about the structure theory of the graph minors project is its practical relevance. For one example, many naturally occuring databases have bounded treewidth (or bounded hypertreewidth, a related notion). This provides significant inroads for hard database problems [GM99,FFG01,Gr01]. Bounded treewidth seems to be an almost universally relevant parameter.

An example to illustrate the main idea of this paper is afforded by the problem of Type Checking of programs written in high-level logic-based programming languages such as ML. This problem has been shown to be complete for

EXP [HM91,KTU94], and thus is highly intractable from the classical point of view. Nevertheless, the ML compilers (that include type-checking subroutines) work efficiently. The explanation is that human-composed programs typically have a maximum type-declaration nesting depth of $k \leq 5$. The FPT type-checking subroutine that runs in time $O(2^k n)$ is thus entirely adequate in practice. One can reasonably speculate that naturally occuring programs have small nesting depth because the programs would otherwise risk becoming incomprehensible to the programmer creating them.

What this example points to (we think) is that often the "inputs" to one computational problem of interest to real-world algorithmics are not at all arbitrary, but rather are produced by other natural computational processes (e.g., the thinking processes and abilities of the programmer) that are themselves subject to computational complexity constraints. In this way, the natural input distributions encountered by abstractly defined computational problems often have inherited structural regularities and restrictions (relevant parameters, in the sense of parameterized complexity) due to the natural complexity constraints on the generative processes. This seems reasonable, although what the resulting relevant "parameters" are may not be obvious. This connection is what we refer to as the *ecology* of computation.

Our thesis is that it is useful to know how all the various parameterized structural notions interact with all the other computational objectives one might have. The main objective of this paper is to illustrate how such a quest can be systematically engaged. The familiar paradigm of efficiently solving various problems for graphs of bounded treewidth just represents one row of a matrix of algorithmic questions that arise from the relevant parameterized structure theories. For the Max Leaf row, we investigate how to solve various problems optimally on graphs G having bounded *max leaf number*, $ml(G) \leq k$, exploiting the structure that bounding this parameter yields.

Table 1 illustrates the idea of such a matrix of algorithmic questions. We use here the shorthand: TW is Treewidth, BW is Bandwidth, VC is Vertex Cover, DS is Dominating Set, G is Genus and ML is Max Leaf. The entry in the 2nd row and 4th column indicates that there is an *FPT* algorithm to optimally solve the Dominating Set problem for a graph G of bandwidth at most k. The entry in the 4th row and second column indicates that it is unknown whether Bandwidth can be solved optimally by an *FPT* algorithm when the parameter is a bound on the domination number of the input. An entry in the table describes the current state of knowledge about the complexity of the problem where the input graph is assumed to have a structural bound described by the *row*, and the problem described by the *column* is to be solved to optimality. The table just gives a few examples of the unbounded conceptual matrix that we are concerned with.

Most of the research in algorithms so far that pertains to this matrix is concerned with the first row and the diagonal. In this paper, we systematically explore the Max Leaf row. In a sequel paper we explore the Vertex Cover row [FLMRS08].

	TW	BW	VC	DS	G	ML
TW	*FPT*	$W[1]$-hard	*FPT*	*FPT*	?	*FPT*
BW	*FPT*	$W[1]$-hard	*FPT*	*FPT*	?	*FPT*
VC	*FPT*	*FPT*	*FPT*	*FPT*	?	*FPT*
DS	$W[1]$-hard	?	$W[1]$-hard	$W[1]$-hard	?	?
G	$W[1]$-hard	$W[1]$-hard	$W[1]$-hard	$W[1]$-hard	*FPT*	$W[1]$-hard
ML	*FPT*	*FPT*	*FPT*	*FPT*	*FPT*	*FPT*

Table 1. The Complexity Ecology of Parameters

One might ask whether these rows are really interesting, since a graph of bounded *max leaf number* is severely restricted in its structure. To be fair, however, a graph of bounded treewidth is also severely restricted, in contrast to an arbitrary graph. How to determine whether a graph of bounded *max leaf number* is 3-colorable in the "best possible" FPT runtime is an easily stated problem for which the answer is not obvious. Another observation that points to the interest in these rows is that there are now known to be many examples of problems that are $W[1]$-hard parameterized by a bound on treewidth, including Bandwidth [BFH94]; List Coloring, Pre-Coloring Extension and Equitable Coloring [FFL+07]; General Factor [Sz08a]; and Minimum Maximum Outdegree [Sz08b]. One must therefore look "below treewidth" for FPT structural parameterizations for these problems. Note that while both bounded *vertex cover number* and bounded *max leaf number* imply bounded treewidth, neither of these structural bounds implies a bound on the other. List Coloring even remains $W[1]$-hard for graphs of bounded *vertex cover number* [FLMRS08]. Lastly, it seems that for severe structural parameterizations such as bounded *vertex cover number* and bounded *max leaf number*, different FPT techniques are brought forward to importance, such as well-quasi-ordering and bounded variable integer linear programming.

3 Systematically Attacking a Row: Kernelization

We first develop the polynomial-time extremal structure theory for graphs of bounded *max leaf number*, using the approach developed in [Pr05,EFLR05]. We prove a polynomial-time kernelization bound for the Max Leaf problem, based on a collection of P-time reduction rules, and then describe kernelizations for various problems in the Max Leaf row of the ecology matrix by: (1) deploying *similar* reduction rules, and (2) adjusting the kernelization bound. We try to describe what we are doing from a general point of view that suggests how the approach might be adapted to other problems.

3.1 Kernelization for the Max Leaf Problem

In order to articulate the structure that a bound on the *max leaf number* imposes, we seek to prove (for the best possible constant c) the following lemma regarding FPT kernelization for the problem:

Lemma 2. *Suppose (G, k) is a reduced instance of* Max Leaf*, with (G, k) a yes-instance of the problem and $(G, k + 1)$ a no-instance. Then $|G| \leq ck$. (Here c is a small constant that we will clarify below.)*

Proving such a "Boundary Lemma" involves two crucial strategic choices:
(1) A choice of *witness structure* for the hypothesis that (G, k) is a yes-instance.
(2) A choice of inductive priorities.
Below, we prove two versions of such a result, to illustrate the methodology.

The overall structure of the argument is "by minimum counterexample" according to the priorities established by (2), which generally make reference to (1). Given these choices, our proof proceeds by a series of small steps consisting of structural claims that lead to a detailed structural picture at the "boundary" — and thereby to the bound on the size of G that is the conclusion of the lemma.

3.2 Boundary Lemma I

Lemma 3. ***Boundary Lemma I.*** *Suppose (G, k) is a reduced instance of* Max Leaf*, with (G, k) a yes-instance of the problem and $(G, k + 1)$ a no-instance. Then $|G| \leq 7.75k$.*

Proof. The proof is by minimum counterexample. If (G, k) is a yes-instance, $G = (V, E)$, then we can assume we are given as a witness structure a tree subgraph $T = (V', E')$ of G that has k leaves, and we can also assume that G is connected.

We do not assume that T is a spanning subgraph. (If T is not spanning, then it clearly extends to a spanning tree T' for G that has at least k leaves.)

A counterexample to the lemma would be a graph $G = (V, E)$ such that: (1) (G, k) is a reduced instance of Max Leaf, (2) (G, k) is a yes-instance of Max Leaf, (3) $(G, k + 1)$ is a no-instance, and (4) $|G| > 7.75k$.

Among all such counterexamples, consider one where the witness subgraph tree T is as small as possible.

Let $O = V - V'$ be the set of vertices not in the witness subtree T, which we will refer to as *outsiders*. Let L denote the leaves of T, I the internal (non-leaf) vertices of T, $B \subseteq I$ the *branch vertices* of T (the non-leaf, internal vertices of T that have degree at least 3 with respect to T), and let J denote the *subdivider vertices* of T (the non-branch internal vertices of T that have degree 2 with respect to T). See Figure 1.

We will also need to discuss the structure of T in more detail, so we introduce the following further terminology. A path $\langle b, j_1, ..., j_r, b' \rangle$ in T where $b, b' \in B$ are branch vertices of T, and the vertices j_i for $i = 1, ..., r$ are subdivider vertices of

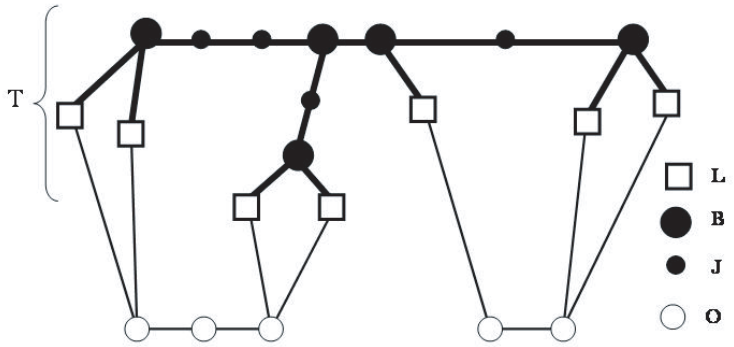

Fig. 1. The witness tree and various sets of vertices.

T is termed a *topological edge* or *topo-edge* of the tree T. In this situation we will say that b and b' are *topologically adjacent* in T, and in order to be able to refer to the length of the path $\langle b, j_1, ..., j_r, b' \rangle$, we may say that b and b' are joined by an *r-topo-edge* in T. We will eventually be interested in structures that arise by considering the subtrees of T induced by 0-topo-edges and 1-topo-edges of T. (Note that a 0-topo-edge is just an ordinary edge of T.)

Claim 1. No internal vertex of T is adjacent in G to a vertex of O.

Proof of Claim 1. Otherwise, we could augment T to a subgraph tree with $k+1$ leaves, contradicting that $(G, k+1)$ is a no-instance. (See Figure 2.)

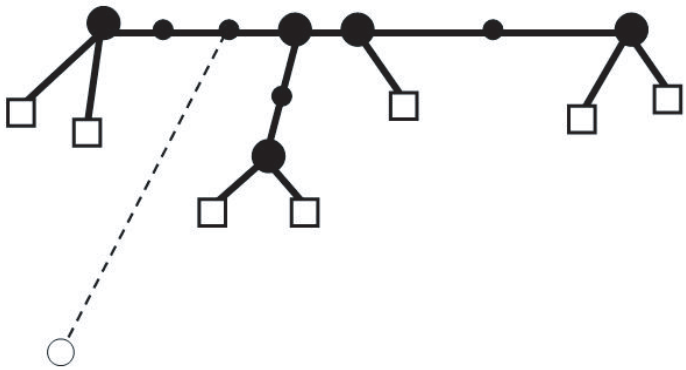

Fig. 2. No internal vertex of T is adjacent to an outsider, else $k+1$ leaves.

Claim 2. A leaf vertex of T is adjacent to at most one outsider.

Proof of Claim 2. Otherwise we contradict that $(G, k+1)$ is a no-instance.

Claim 3. The subgraph $\langle O \rangle$ induced by the outsiders is acyclic.

Proof of Claim 3. Otherwise, since G is connected, we contradict that $(G, k+1)$ is a no-instance. See Figure 3.

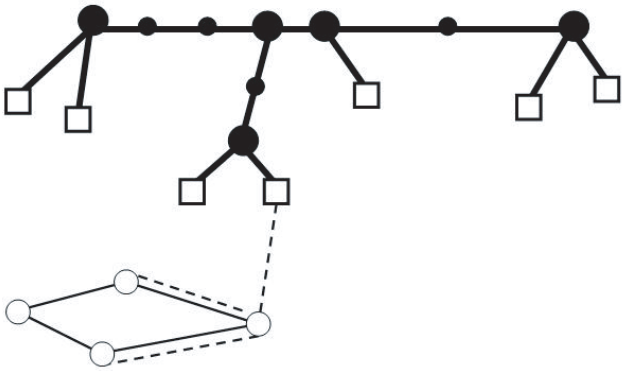

Fig. 3. $\langle O \rangle$ is acyclic, else $k+1$ leaves.

Claim 4. The subgraph $\langle O \rangle$ induced by the outsiders has maximum degree at most 2.

Proof of Claim 4. Otherwise we contradict that $(G, k+1)$ is a no-instance. See Figure 4.

By Claims 3 and 4, the subgraph $\langle O \rangle$ induced by the outsiders consists of a union of paths.

Claim 5. A leaf of G cannot be adjacent to an interior vertex of a path of $\langle O \rangle$.

Proof of Claim 5. Otherwise we contradict that $(G, k+1)$ is a no-instance.

Claims 1-5 show that $\langle O \rangle$ consists of a disjoint union of paths, and that the interior vertices of these paths have degree 2 in G. This motivates us to look for a reduction rule that addresses this structure. The *Two Adjacent Degree 2 Rule* says that for $k' = k$, if u and v are two adjacent vertices of degree 2, then the edge uv can either be deleted or contracted, depending on whether or not it is a bridge.

Claim 6. $\langle O \rangle$ consists of a union of paths, where each path has at most 3 vertices.

Proof of Claim 6. Otherwise the "Two Adjacent Degree 2" reduction rule applies.

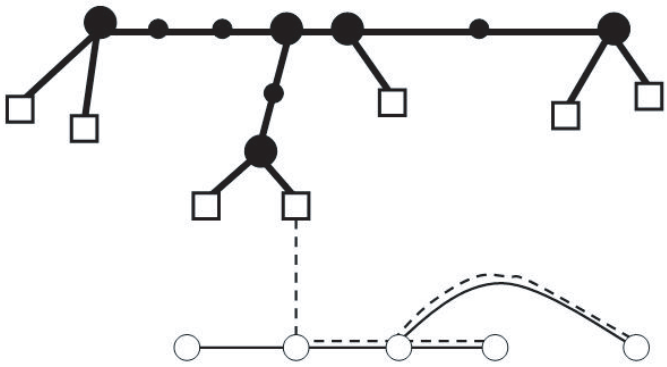

Fig. 4. $\langle O \rangle$ has maximum degree at most 2, else $k+1$ leaves.

The evident structure of $\langle O \rangle$ suggests looking for a reduction that applies to vertices of degree 1 in G. In fact, there is a simple but not at all obvious rule to eliminate vertices of degree 1. (The reader might want to take a moment to try to discover it, in order to appreciate some of the depth of the subject of reduction rules. A degree one rule: how hard can that be?)

Claim 7. $|O| \leq .75k$

Proof of Claim 7. This follows from the fact that T has k leaves, and from Claims 2,5 and 6, inducting on the number of path components in $\langle O \rangle$. The "worst case" for the induction is where a path component of $\langle O \rangle$ has 3 vertices (a path of length 2). In this case each endpoint of the path must be adjacent to at least two leaves of T, else G is reducible either by the "Degree 1" or the "Two Adjacent Degree 2" reduction rules. Thus the number of leaves is $k \geq 4|O|/3$.

The picture that has emerged through Claims 1-7 is starting to give us a handle on how big G can be. Next we must bound the size of T.

Claim 8. $|B| \leq k-2$

Proof of Claim 8. A straightforward induction on the number of leaves.

Claim 9. The subgraph induced by the vertices of a topological edge of T contains no further edges.

Proof of Claim 9. This is trivially true for an r-topo-edge where $r = 0$, so suppose $r \geq 1$. But then we can re-engineer T to have $k+1$ leaves, as shown in Figure 5.

Claim 10. There are no r-topological edges in T for $r \geq 6$.

Proof of Claim 10. Suppose we have a path $\langle b, j_1, ..., j_r, b' \rangle$ in T where $b, b' \in B$ are branch vertices of T, and the vertices j_i for $i = 1, ..., r$ are subdivider vertices of T, where $r \geq 6$. let T_b denote the subtree of T "to the left" of b, and let $T_{b'}$ denote the subtree of T "to the right" of b'. The vertex j_3 cannot be adjacent

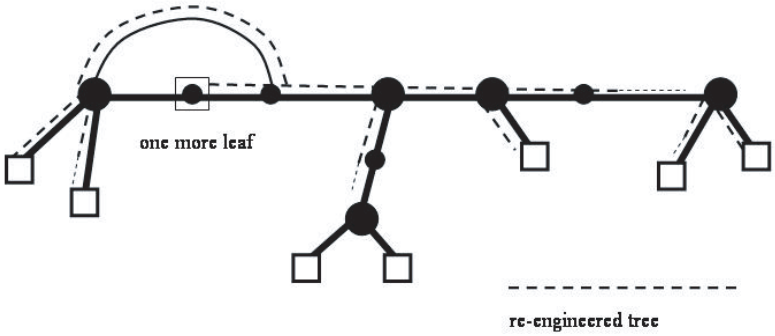

Fig. 5. Topo-edges are induced subgraphs.

in G to a vertex of T_b, otherwise we can re-engineer T to have $k+1$ leaves as shown in Figure 6.

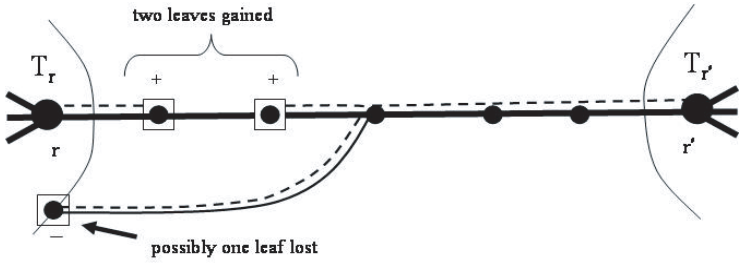

Fig. 6. Long topo-edges of T have middle vertices of degree 2 in G.

The vertex j_3 cannot be adjacent to a vertex of $T_{b'}$ for similar reasons. By Claim 9, and by symmetry, the vertices j_3 and j_4 must have degree 2 in G. But then G is reducible, either by contracting a bridge or by the "Two Adjacent Degree 2 Rule".

Claim 11. Each leaf in T is adjacent in T to a branch vertex.

Proof of Claim 11. Otherwise, we would contradict that T is as small as possible. See Figure 7.

Claim 12. $|J| \leq 5(k-3)$

Proof of Claim 12. This follows from Claims 9 and 10.

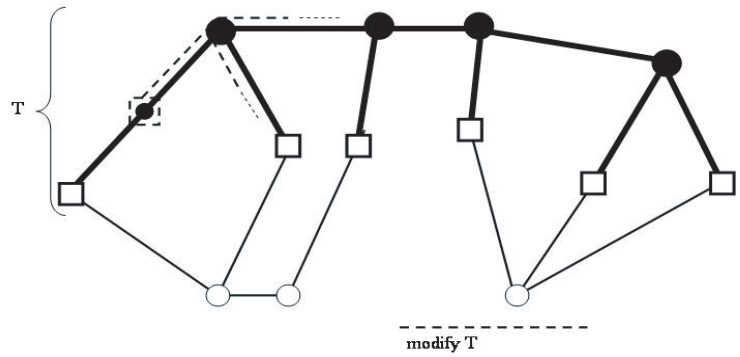

Fig. 7. A leaf is adjacent to a branch, else smaller T.

We can now conclude the proof of Boundary Lemma I on the basis of Claims 7,8 and 12.

3.3 Boundary Lemma II

Lemma 4. ***Boundary Lemma II.*** *Suppose* (G, k) *is a reduced instance of* Max Leaf*, with* (G, k) *a yes-instance of the problem and* $(G, k+1)$ *a no-instance. Then* $|G| \leq 5.75k$*.*

Proof. The proof is by minimum counterexample. Witnessing that (G, k) is a yes-instance we have a tree T with k leaves, as in the proof of Boundary Lemma I. Here we will have a little more structure and another inductive priority. We consider that the tree T is equipped with a root vertex $r \in V'$. The possible counterexample that we entertain in our argument is one where:
(1) T is as small as possible, and among all counterexamples satisfying this requirement, one where
(2) the sum over all leaves $l \in L$ of the distance in T from r to l is minimized.

All of the structural claims from the proof of Boundary Lemma I hold here as well, since essentially all we have done is add one further inductive priority. This additional priority allows us to establish a strengthening of Claim 10.

Claim 13. T does not have r-topological edges for $r \geq 4$.

Proof of Claim 13. Suppose we have a path $\langle b, j_1, ..., j_r, b' \rangle$ in T where $b, b' \in B$ are branch vertices of T, and the vertices j_i for $i = 1, ..., r$ are subdivider vertices of T, where $r \geq 4$. Let T_b denote the subtree of T "to the left" of b, and let $T_{b'}$ denote the subtree of T "to the right" of b'. We can suppose that the root of T lies in $T_{b'}$, without loss of generality. The vertices j_1 and j_2 cannot be adjacent to vertices in $T_{b'}$, else we can re-engineer T to have $k+1$ leaves, as in the proof of Claim 10. By Claim 9, and since G is irreducible (in particular, j_1 and j_2 do

not both have degree 2 in G), at least one of j_1 or j_2 is adjacent by an edge e to a vertex x of T_b. But then by adding e to T and removing the edge xu, where xu is the first edge on the path in T from x to b, we can obtain a modified tree with k leaves where the priority (2) has been improved.

That concludes the proof of BL II.

We next turn to the issue of how this understanding can be used to efficiently solve problems in our chosen row of the parameterized complexity ecology matrix.

Theorem 1. *For graphs of max leaf number bounded by k, the minimum domination number can be computed in time $O^*(143^k)$ based on a polynomial-time reduction to a kernel of size at most $7.5k$.*

Proof. Since this is an FPT result, we are necessarily (by Lemma 1) interested in effective kernelization for *this* problem. We must therefore develop a polynomial-time extremal account of the boundary case for the induction.

We take the following hypotheses:
(1) (G, k) is a yes-instance of Max Leaf.
(2) $(G, k+1)$ is a no-instance of Max Leaf.
(3) There is a witness structure for (1) that satisfies the inductive priorities of the proof of Boundary Lemma II.
(4) G is *reduced* according to an admissible set of polynomial time kernelization rules.

Here by *reduced* we mean P-time reduction rules that are compatible with the objective of computing a minimum dominating set. Many of the structural claims proved above can be imported to this new situation, modified in some cases because of changes to the admissible set of reduction rules.

Here, because we are computing a minimum dominating set, we are allowed reduction rules that are compatible with this objective: rules that transform G to G' in such a way that given a minimum dominating set for G', we can easily compute a minimum dominating set for G. To the extent that we can find reduction rules for this new computational objective that "mimic" or approximate the ones that were available for the Max Leaf problem, the structural claims about the kernel still (with some modifications) carry over, and we can conclude similar kernelization bounds for problems in the row of the complexity ecology matrix that are amenable to this approach.

The reduction rules shown in Figure 8 below can be used in this way for the Minimum Dominating Set (MDS) problem, for graphs of bounded max leaf number. To solve the minimum domination problem on graphs of bounded max-leaf number, we attempt to solve the decision problem for MDS (that is: does G have a dominating set of cardinality l) for each l. For each l, we iteratively apply the reduction rules for MDS given in Figure 8 until they can no longer be applied. Since the reduction rules given in Figure 8 can be performed by a series of vertex deletions, edge deletions, and edge contractions, the resulting reduced graph G' is a minor of the original graph G. Since the max-leaf number of a graph is a monotonically non-decreasing property of the graphs in the minor ordering

(that is, if H and G are connected and H is a minor of G, then $ml(H) \leq ml(G)$), we have that $ml(G') \leq ml(G) \leq k$. In particular, the resulting reduced graph G' has no chains of vertices of degree 2 longer than 2.

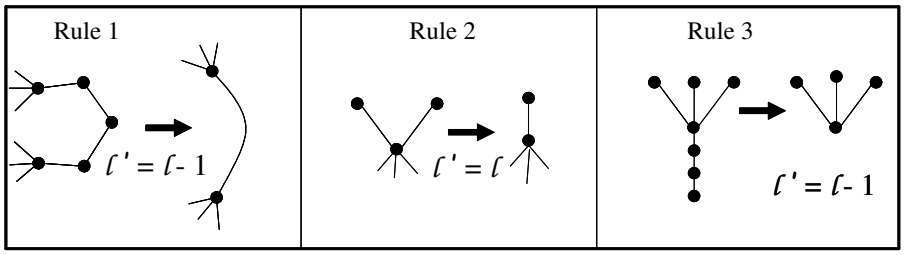

Fig. 8. Reduction rules for minimum domination.

The argument for the bound on the kernel size is by minimum counterexample. One of our hypotheses is that (G, k) is a yes-instance for Max Leaf. We can assume we are given as a witness structure a tree subgraph $T = (V', E')$ of G that has k leaves, and we can also assume that G is connected.

We do not assume that T is a spanning subgraph. (If T is not spanning, then it clearly extends to a spanning tree T' for G that has at least k leaves.)

A counterexample to our theorem would be a graph $G = (V, E)$ such that: (1) (G, k) is a reduced instance of Max Leaf, (2) (G, k) is a yes-instance of Max Leaf, (3) $(G, k + 1)$ is a no-instance, and (4) $|G| > 7.5k$.

Among all such counterexamples, we consider one where a rooted witness subgraph tree T is as small as possible, and meeting the secondary inductive priority of Boundary Lemma II. We now consider how the various Claims of the Max Leaf kernelization bound fare under the modified notion of "reduced instance" that we must consider here in order to solve the MDS problem (that is: we consider the structure of an instance that is reduced with respect to the similar but modified reduction rules depicted in Figure 8).

Claims 1 through 5, as well as 8, 9 and 11, hold by the same arguments as before. We next argue that modified versions of the other structural claims hold, yielding our claimed kernelization bound.

Claim 6′. $\langle O \rangle$ consists of a union of paths, where each path has at most 4 vertices.

Proof of Claim 6′. Otherwise, MDS Reduction Rule 1 could be applied, contradicting our hypothesis that the instance is reduced.

Claim 7′. $|O| \leq 1.5k$

Proof of Claim 7′. The argument for Claim 7 is modified by noting that the "worst case" is where a component of $\langle 0 \rangle$ consists of a three-vertex path, where one endpoint is a vertex of degree 1, and the other endpoint has at least two leaf neighbors.

Claim 13′. The witness tree T does not have r-topological edges for $r \geq 5$.
Proof of Claim 13′. The argument for Claim 13 is modified (where we use the same notation to discuss the situation) by noting that none of the vertices j_1, j_2 or j_3 can be adjacent to vertices in $T_{b'}$, else we can re-engineer T to have $k+1$ leaves. These three vertices cannot all have degree 2, else Reduction Rule 1 applies. By Claim 9, at least one of these must be adjacent to a vertex x in T_b, providing an opportunity to improve the secondary inductive priority.

The kernel can be analyzed by means of the algorithm due to Fomin, Kratsch and Woeginger [FKW04], yielding the running time stated for our algorithm. Knowing the domination number of the problem kernel allows us to compute the domination number of the input graph by retracing this information backwards along the kernelization path in polynomial time. □

What was the best previous result for this problem? A graph of bounded max leaf number has bounded pathwidth, and thus almost all of the entries in the "max leaf row" can be handled by standard bounded pathwidth algorithmics. This is true, but applied in a simple manner, entails a cost of $O^*(2^{35k^3})$ in order to compute the path decomposition. The point of this investigation is to ask for the *best possible* FPT algorithms for FPT entries in this row, that is, how does one best (and hopefully, systematically) exploit bounded max leaf number?

The following theorem is based on essentially the same approach, making use of the reduction rules shown in Figure 9.

Theorem 2. *For graphs of max leaf number bounded by k, the maximum size of an independent set can be computed in time $O^*(2.972^k)$ based on a polynomial-time reduction to a kernel of size at most $7k$.*

Proof. The imported structural claims give a bound of $4.5k$ on the size of a vertex cover for the kernel, which yields the claimed running time by using the algorithm of Chen, Kanj and Xia [CKX05] to analyze the kernel. We must first note that all of the reduction rules shown in Figure 9 are admissible in the sense that they preserve a bound on the max-leaf number. An upper bound of $4.5k$ on the size of a minimum cardinality vertex cover is demonstrated by considering a vertex cover that consists of the leaves of G (at most k), the branch vertices of the tree T (at most k), together with at most $2k$ vertices of the J vertices (based on *Claim* 13′, as in the proof of Theorem 1), and together with at most $k/2$ vertices of $\langle O \rangle$.

Many other NP-hard problems can be addressed for graphs of bounded max-leaf number in much the same way.

Theorem 3. *For graphs of max leaf number bounded by k, it can be determined in $O^*(54^k)$ whether the graph has a Hamiltonian circuit, based on a polynomial-time reduction to a kernel of size at most $5.75k$.*

Proof. Reduction rules admissible for this problem (because they preserve the bound on $ml(G)$) include:

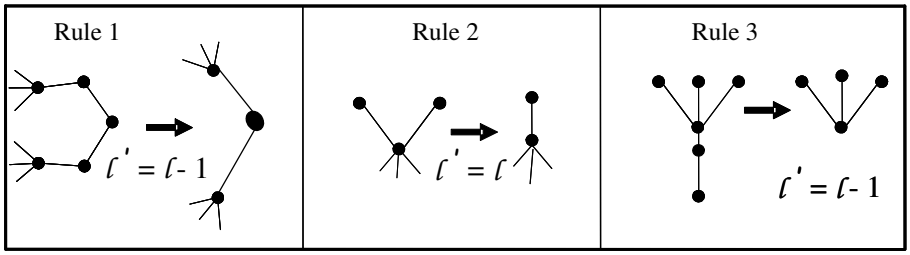

Fig. 9. Reduction rules for Maximum Independent Set.

• If there is a vertex of degree 1, then answer NO.
• Edges between vertices of degree 2 can be contracted.
This leads to a kernelization bound identical to that for Max Leaf (*via* Boundary Lemma II), since all of the structural claims hold in this case by the same arguments. The analysis of the kernel is by means of the dynamic programming algorithm of Held and Karp [HK70]. □

4 Systematically Attacking a Row: A Win/Win Connection to Pathwidth

Several of our results in the previous section based on the combination of systematic kernelization and exponential analysis of the kernel can be improved by what has been called a "Win/Win" algorithm that in polynomial time provides a useful connection between two different parameters [PS03,Fe03]. An early example in the study of parameterized algorithms is the linear time algorithm described by Fellows and Langston that given a graph G outputs either a cycle of length at least k, or a tree decomposition of width at most k [FL89b]. (To emphasize — the algorithm runs in $O(n)$ time for any $k \leq n$.)

Theorem 4. *There is a linear time algorithm that on input a connected graph G, outputs either:*
(1) A spanning tree of G having at least k leaves, or
(2) A path-decomposition of G of width at most $2k$.

Although the proof is easy, we have not been able to find this already in the literature.

Proof. Compute a breadth-first spanning tree of G. If any layer of the tree has population at least k, then the breadth-first spanning tree computed in this way has at least k leaves. Otherwise, suppose the layers of the spanning tree are: $L_0, L_1, ..., L_r$. The series of bags:

$$L_0 \cup L_1,\ L_1 \cup L_2,\ ...,\ L_{r-2} \cup L_{r-1},\ L_{r-1} \cup L_r$$

gives a path decomposition of G of width bounded by $2k$.

This yields more efficient FPT algorithms for some (but not all) of the problems we have considered. For example, combining the above algorithm with the carefully engineered dynamic programming algorithm for DOMINATING SET of Telle and Proskurowsky [TP93] (refined by Alber and Niedermeier [AN02]), we get an $O^*(4^k)$ FPT algorithm for the MINIMUM DOMINATING SET problem, for graphs of *max leaf number* bounded by k. Similarly, determining whether a graph of *max leaf number* bounded by k is 3-colorable can be accomplished in $O^*(9^k)$ time.

5 Systematically Attacking a Row: Well-Quasiordering

One of the most powerful FPT classification tools is *well-quasiordering*. Graphs in general are well-quasiordered by minors (the celebrated Graph Minor Theorem), and also importantly, determining whether a graph H is a minor of a graph G, parameterized by H, is FPT (we say that *the minor order has FPT order tests*) [RS85,RS04].

Bounding a structural parameter, such as the *max leaf number*, can lead to "even more powerful" — but generally easier to prove(!) — well-quasiordering FPT classification tools applicable to a given row of the parameterized complexity ecology matrix. In [Fe03] it is shown that graphs of bounded *vertex cover number* are well-quasiordered by induced subgraphs and admit linear-time FPT order tests.

Here we prove an analogous result, and give an application. For background on well-quasiordering concepts and standard methods of proof in the context of FPT algorithms, we refer the reader to [DF99].

Definition 1. *The* topological order *on graphs (with loops and multiple edges allowed) is defined: $H \leq_{top} G$ if and only if G contains a subgraph G' that is (up to isomorphism) a* subdivision *of H, where a subdivision of H is a graph obtained from H by replacing edges of H by (vertex disjoint) paths. We use $\mathcal{F}_k$ to denote the set of graphs*

$$\mathcal{F}_k = \{G : ml(G) \leq k\}$$

Theorem 5. *For every k, $\mathcal{F}_k$*
(1) is well-quasiordered by the topological ordering, and
(2) admits linear time FPT order tests.

Proof. By a theorem of Kleitman and West [KW91] every graph G with $ml(G) \leq k$ is a subdivision of a graph on at most $4k-2$ vertices. Suppose there is an infinite bad sequence of graphs of bounded *max leaf number* in the topological order. Because the number of distinct graphs on at most $4k-2$ vertices is bounded by a function of k, by the Pigeonhole Principle there is an infinite bad subsequence $(G_1, G_2, ...)$ where for all i, G_i is a subdivision of a fixed graph G on $k' \leq 4k-2$ vertices. Let $(e_1, ..., e_m)$ be a fixed enumeration of the edges of G. Then each G_i is essentially described by the information: (1) the graph G, and (2) a length m

census vector $(s_{i,1}, ..., s_{i,m})$ that records the numbers $s_{i,j}$ of degree 2 vertices on the path that replaces the edge e_j in order to produce G_i from G. Clearly, if for $i < i'$, and for all j, $s_{i,j} \leq s_{i',j}$, then $G_i \leq_{top} G_{i'}$. By Higman's Lemma, this has to happen, contradicting that the sequence is bad. This establishes (1).

By a theorem of Bienstock et al. [BRST91] (see also §4) graphs of bounded *max leaf number* have bounded pathwidth, and since for a fixed graph H, the property of containing H topologically is expressible in MSO logic, we have (2).

We next describe an application to the problem Topological Bandwidth. The *bandwidth* $bw(G)$ of a graph G is the minimum, over all permutations of the vertex set of G, of the maximum distance in the order given by a permutation of adjacent vertices of G. It can happen that a subdivision G' of G has smaller bandwidth than the bandwidth of G. The *topological bandwidth* $tbw(G)$ of a graph G is the minimum, taken over all subdivisions G' of G, of the bandwidth of G'. Determining if $tbw(G) \leq k$ (parameterized by k) is hard for $W[t]$ for all t, even for the restriction to trees [BFH94].

We address the following decision problem in our row of the parameterized complexity ecology matrix:
Topological Bandwidth parameterized by max leaf number (TBW-ml)
Instance: A graph G with $ml(G) \leq k$, and a positive integer r.
Parameter: k
Question: Is $tbw(G) \leq r$?

Theorem 6. TBW-ml *is fixed-parameter tractable.*

Proof. It is sufficient to show that if $ml(G) \leq k$ then $tbw(G) \leq k'$ where $k' = f(k)$ depends only on k, where the function $f(k)$ is computable. Suppose this is shown. Then we compute k' (in time that depends only on k) and if $r \geq k'$, then we answer YES. Otherwise, by the well-quasiordering of $\mathcal{F}_k$ under $\leq_{top}$, there is a finite obstruction set $\mathcal{O}_r$ such that $tbw(G) \leq r$ if and only if for every graph $H \in \mathcal{O}_r$, H is not topologically contained in G. By (2) of the above theorem, this is sufficient to conclude that TBW-ml is solvable in linear-time *nonuniform* FPT.[8] This result can be converted into uniform FPT by the methods of [FL89a] and [FL89b] (both methods are exposited in [DF99]).

To show what we propose, we argue that the bandwidth of G is bounded by $k' = (4k-3) + 2\binom{4k-2}{2}$, so that this is an upper bound on $tbw(G)$. For this we can use the fact that G is a subdivision of a graph H on at most $4k-2$ vertices (by the theorem of Kleitman and West [KW91]). Let $v_1, ..., v_n$ be the vertices of H, and let $e_1, ..., e_m$ be the edges of H. We describe a layout of G that achieves the claimed bound. A *layout* is a 1:1 mapping $l : V(G) \to Z$, and the bandwidth of a layout l is the maximum, over all pairs u, v of adjacent vertices of G of $|l(u) - l(v)|$.
(1) Assign vertex v_i the position $l(v_i) = -i$.
(2) Associate to each edge e_j the set of positive integers

$$P_j = \{r : r \equiv j - 1 \pmod{m}\}$$

[8] See [DF99] for the distinction between uniform and nonuniform FPT.

(3) As G is obtained from H by replacing each edge e_j with a path p_j, complete the description of the layout l of G by assigning the vertices of the path p_j to positions in P_j in the natural way. This means that the path goes "out and back" in an interleaved manner.
It is straightforward to check that the bound is achieved.

6 Bandwidth is FPT parameterized by *max leaf number*

The Bandwidth problem, parameterized in the natural way, is known to be hard for $W[t]$ for all t [BFH94]. Since a graph of bandwidth at most b has pathwidth at most b, it follows that if we parameterize the Bandwidth problem by the pathwidth of the input, then this is also hard for $W[t]$ for all t.

In this section we show that the following problem is FPT:
Bandwidth parameterized by max leaf number (BW-ml)
Instance: A graph G for which $ml(G) \leq k$, and a positive integer r.
Parameter: k
Question: Is $bw(G) \leq r$?

Theorem 7. *BW-ml is fixed-parameter tractable.*

Proof. As formal details would be arduous, and the algorithm is anyway impractical, we provide only a sketch of a proof.

The overall structure of our algorithm is that we branch on an exhaustive set of *solution plans* that has size bounded by a function of k, and that is sure to capture at least one solution, if any exists. Then, to determine whether a particular solution plan can be realized, we employ a subroutine that consists of solving (in polynomial time) a system of linear equations.

We use the fact that if G has $ml(G) \leq k$ then G is a subdivision of a graph H on at most $4k - 2$ vertices [KW91]. We also use the fact that if $ml(G) \leq k$, then $bw(G) \leq k' = O(k^2)$, established in the proof of Theorem 6. In particular, we have the initial step:

If $r > k'$ then output YES.

A *solution plan* consists of two layers of specification: (A) a topological and local specification of the general shape of a solution, and how it might be carried out in key local areas, and (B) in the remaining *gaps* between these local specifications, a local plan / formula for negotiation, that contributes to a global calculation of whether the solution plan can be achieved. Part A of a solution plan consists of the following items of information:
(A.1) A permutation of the vertices of H representing the intended order of these vertices in a layout of G. (Note that all the other vertices of G are vertices of degree 2 on paths that replace the edges of H in order to create G.) We consider the vertices of H to be *reference points* of the plan.
(A.2) A description of the topological routing of the paths that the edges of H are replaced by in producing G from H. (For an example, see the upper part of

Figure 10. The vertical lines indicate an injective mapping of the vertices into integer positions on the real line. The square vertices represent "points of change of direction" of the topological routing.)

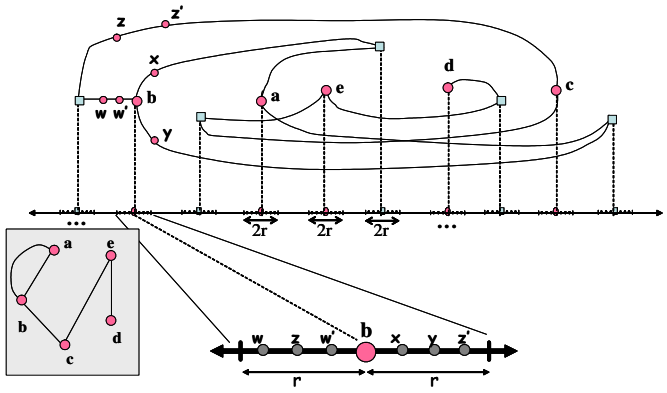

Fig. 10. Part A of a Solution Plan.

In such a plan, we are only interested in the zones that the routes of the paths cross or enter, and we allow that the route of the path between vertices u and v of H may make up to two changes of direction. (We argue below that if there is any solution, then there is a solution in which each path-route makes at most two changes of direction. This issue is illustrated in Figure 11.)

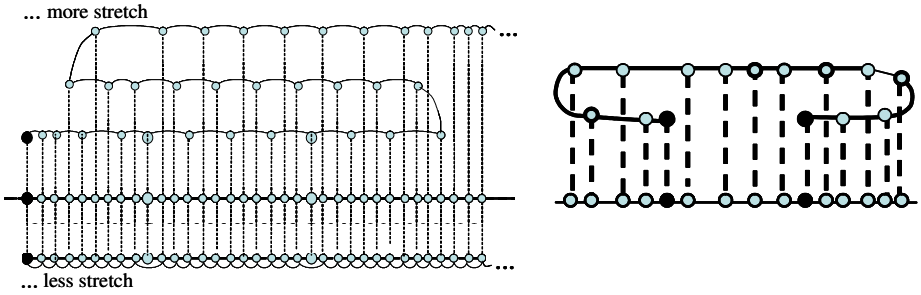

Fig. 11. No wiggle-waggle necessary.

(A.3) If a plan calls for a path to make a change of direction, then this creates a new reference point. The information for a plan includes a permutation of all of the reference points (the vertices of H, together with all of the points that represent changes of direction of the routes of the paths).
(A.4) A plan also completely specifies the layout (we will use the phrase: *reference-point locally*) within a distance of r of any reference point, allowing that these neighborhoods of adjacent reference points may be coalesced, in the case that the reference points are planned to be close together. This reference-point local specification for a particular reference point, details how various degree 2 vertices of the edge-paths present in that region of the topological plan are ordered among the $2r$ layout positions in the (relative) local neighborhood of the reference point. (Note that this reference-point local specification requires that the numbers of degree two vertices on the various paths that build G from H are sufficient to meet the specification, else the plan can be discarded from our search of the possibilities for a solution.) See the lower part of Figure 10.

We claim that the number of solution plans (part A) is bounded by a function of k. Let us start with the number of possible reference-point local specifications that might be given a specific reference point. Since $r \leq k'$ by *Step 1* of the algorithm, there are at most $2k'$ positions to be assigned to the edge-paths routed in the neighborhood of the reference point by the topological part of the solution plan, so we have a bound of $m^{2k'}$, where m is the number of edges in H, which gives a crude bound of $O(k^{O(k^2)})$. Since there are $O(k^2)$ reference points, this gives a bound of $O(k^{O(k^4)})$ for the number of possible reference-point local specifications to be explored for any given permutation of the reference points. The number of such permutations (which implicitly bounds the number of topological plans) is bounded by $O(O(k^2)!) = O(k^{O(k^2)})$. So the total number of solution plans (part A) is bounded by $O(k^{O(k^4)})$. We will use k'' to designate this bound.

Crucial to the correctness of our algorithm is the following claim.

Lemma 5. *If G has any layout of bandwidth at most r, then it has one where each edge-path makes at most two turns.*

Proof. Let $f : V \to N$ denote a layout function, mapping the vertices of G to integer positions on the real line, witnessing that the bandwidth of G is at most r. As discussed above, we view G as a subdivision of a graph H on at most $4k-2$ vertices. Suppose x and y are adjacent vertices of H, and denote the chain of degree 2 vertices in G between x and y by v_i. Thus is G we have the "edge-path" $P(x, y) : x, v_1, v_2, ..., v_m, y$. We will use S to denote the set of vertices v_i.

We describe how to modify f to obtain a different solution f' that lays out the edge-path in a way that involves at most two turns. Let P denote the set of positions $f(S)$. Since f is injective, $|P| = m$. Our modification of f consists in mapping S to P in a different way that involves at most two turns. Making a similar modification for each edge-path of G, since these modifications can be made independently, yields the lemma.

For convenience, suppose $f(x) < f(y)$. We can consider that P is partitioned into three sets

$$P = P_0 \cup P_1 \cup P_2$$

where P_0 is the set of positions in P that are less than $f(x)$, P_1 is the set of positions in P that are between $f(x)$ and $f(y)$, and P_2 is the set of positions in P that are greater than $f(y)$. Let $m_i = |P_i|$, for $i = 0, 1, 2$.

Let $p(0, i)$ for $i = 1, ..., m_0$ denote the positions in P_0, sorted according to increasing distance ("to the left") from $f(x)$. Similarly, let $p(1, i)$ for $i = 1, ..., m_1$ denote the positions in P_1 sorted by increasing distance ("to the right") from $f(x)$, and let $p(2, i)$ for $i = 1, ..., m_2$ denote the positions in P_2 sorted by increasing distance ("to the right") from $f(y)$.

For simplicity, we will assume that all of the sets of positions P_i are nonempty and have size at least 2. (It is easy to modify the argument to handle the other cases.) Our description of f' is summarized: we assign the vertices of S "out and back" in an interleaved manner to the positions in P_0, then progressively to the positions in P_1, and then "out and back" in an interleaved manner to the positions in P_2. In particular, we make $f'(v_1) = p(0, 2)$, $f'(v_2) = p(0, 4)$,...,$f'(v_i) = p(0, 2i)$,... until we reach the leftmost position of P_0, and then progressively back through the alternately skipped positions, and similarly with regards the positions in P_2. We say that positions $p(0, i)$ and $p(0, j)$ are *nearly consecutive* if $|i - j| = 2$. We argue that the distance between nearly consecutive positions in P_0 is at most r (and similarly, that the distance between nearly consecutive positions in P_2 is at most r). This is true, because otherwise, f would fail to be a bandwidth r layout function.

Before we discuss what constitutes Part B of a solution plan, we need to take account of what remains to be determined if a solution plan (Part A) is to be realized. What Part A specifies is an ordering of the reference vertices together with a description of what a solution layout should look like in the vicinity of the reference vertices. This implicitly involves a commitment to some numbers of subdivisions on the edges of G, and the commitment is feasible only if the number of subdivisions of each edge of H (in the description of G) is greater than or equal to the commitment implicit in Part A of a solution plan. Fix attention on solution plan $\mathcal{P}$, partially specified by the Part A information. For each edge e of H, let $\mathcal{P}(e)$ denote the number of "further" subdivisions (i.e., beyond those already implied in Part A) needed on edge e in order to reach the number in the description of G as a subdivision of H. What remains to be determined is schematically illustrated in Figure 12.

In the figure, the boxes represent the local solution information specified in Part A (that is, local to the reference vertices). Between these boxes are tracks representing the edge-paths that go between these areas. Each track is part of the routing of a edge-path in the topological specification of Part A. Note that the situation between two consecutive boxes has a simple structure: there is just a set of edge-paths between the two boxes. We refer to the situation between two *boxes* as a *gap*. Let $\mathcal{G}_i$, $i = 1, ..., t$ denote the set of gaps of the Part A,

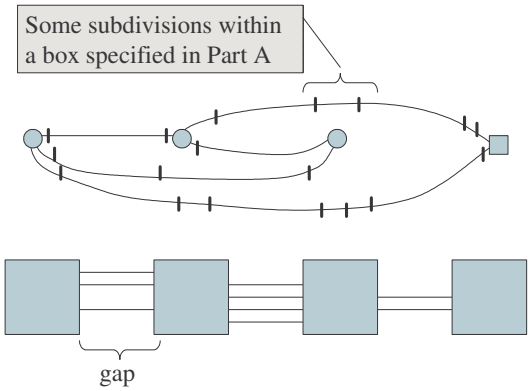

Fig. 12. The situation at a gap.

partial solution specification. Abusing notation in harmless way, we will treat $\mathcal{G}_i$ as denoting the set of edge-paths in the gap.

What remains to be determined is whether subdivisions can be introduced to the edge-paths in the gaps, in a way that is locally consistent with a bandwidth r layout, and so the the total number of subdivisions introduced for each edge e of H, sums to $\mathcal{P}(e)$, summing over the gaps that include e. Let $m(i, e)$ denote the number of subdivisions introduced to the track representing e in the gap $\mathcal{G}_i$.

What we require, then, is that

$$\forall e \quad \sum_{i:e\in\mathcal{G}_i} m(i, e) = \mathcal{P}(e) \tag{1}$$

subject to local consistency with bandwidth at most r.

We make this determination, algorithmically, as follows. First of all, for each edge e of H, $\mathcal{P}(e)$ is a constant that we calculate for G according to Part A of the solution plan specification. The $m(i, e)$ are treated as integer variables, and we assemble a system of linear equations that has a feasible solution if and only if the solution plan partially specified by Part A, can be carried out. Clearly, for each gap $\mathcal{G}_i$, local consistency with a bandwidth at most r layout depends on a suitable relationship between the values of the variables

$$M(i) = \{m(i, e) : e \in \mathcal{G}_i\}$$

The local consistency constraints are expressed as a set of linear equations, using variables special to Part B of a solution plan specification. We next study the situation for a specific gap $\mathcal{G}_i$. Refer to Figure 13. We may consider that we "build" the part of the solution in the gap from left to right. Each step consists of either terminating the process, or introducing a subdivision on one of the tracks. Our local consistency concern is that no edge of G is stretched more than a distance r. Because of the Part A specification, we begin with a

state based on how much each track is "already stretched" (to the left) due to the specification by Part A of the box on the left of the gap. Each step changes the state. There are at most $r^{|\mathcal{G}_i|}$ states that are consistent with a bandwidth (at most) r layout. When we terminate, we have to consider whether the state information concerning "stretch to the left" is compatible with the stretch imposed by the specification of Part A concerning the box on the right of the gap.

We can model the situation with a finite-state machine $\mathcal{M}_i$. The *alphabet* corresponds to the tracks of the layout. *Processing a letter* corresponds to introducing a subdivision (the "next vertex to be laid out in the gap") to one of the tracks, the *accept states* correspond to the states (based on left-stretch) that are compatible with the box on the right of the gap (and hence are acceptable for terminating the local construction of a partial solution). Local consistency of the values of the integer variables of $M(i)$ corresponds to $\mathcal{M}_i$ accepting a word whose *letter content* is the same as the values of the variables of $M(i)$. Figure 14 shows an example of a gap automaton.

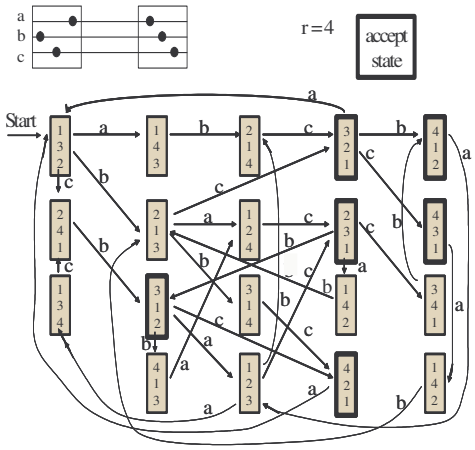

Fig. 13. Example of a Gap Automaton.

In a digraph D, define two directed paths Δ and Δ' from a vertex s to a vertex t to be *arc-equivalent*, if for every arc a of D, Δ and Δ' pass through the arc a the same number of times. We need the following lemma.

Lemma 6. *Any directed path Δ through a finite digraph D on n vertices from a vertex s to a vertex t of D is arc-equivalent to a directed path Δ' from s to t, where Δ' has the form:*
(1) Δ' consists of an underlying directed path ρ from s to t of length at most

$O(n^2)$, *together with,*
(2) Some numbers of short loops*, where each such* short loop l *begins and ends at a vertex* v *of* ρ*, and has length at most* n*.*

Proof. We give an algorithmic proof. Consider the sequence σ of vertices visited by Δ from s to t in D. If σ has length greater than n, then σ contains a short loop relative to a vertex u (i.e., u is repeated in σ). Let σ' denote σ reduced by this short loop. Repeating this, one obtains some numbers of short loops rooted at various vertices, together with a final reduced sequence σ^* of length at most n. Loops are either rooted at a vertex x of σ^* (in which case we are done) or not. If not, then an augmentation of σ^* by a short loop that includes the root x (for each such x) is sufficient to conclude the lemma. □

We are now in a position to describe Part B of a solution plan. Part B specifies, for each gap $\mathcal{G}_i$, a transition path ρ_i from the start state, to an accept state of the gap automaton $\mathcal{M}_i$, of length at most the square of the number of states of $\mathcal{M}_i$.

The number of solution plans (Part A + Part B) is bounded by $O(k^{O(k^{O(k^2)})})$.

For each fully-articulated solution plan, the only thing yet undetermined is the number of times any possible (short) loop might be executed, in each gap. We express our feasibility constraints in terms of one variable for each such possible loop, for each gap, with a linear equation summing appropriately, the variables of $M(i)$, and we adjust the global linear constraints of the equations of (1) appropriately, according to the subdivisions of the various edge-paths implicit in Part B of a solution plan specification (not already accounted for by the calculations relative to Part A). The number of variables involved in the system of linear equations for a fully articulated solution plan is crudely bounded by the number of short loops that are possible; for an n state gap automaton this is bounded by n^n. The number of variables is thus also crudely bounded by $O(k^{O(k^{O(k^2)})})$.

We must argue that if there is any solution, then one of our branches will succeed. This follows from Lemma 5, and the fact that what a solution f "does" in a gap corresponds directly to a path ρ from the start state to an accept state in the automaton for that gap, where the numbers of subdivisions introduced add up to $\mathcal{P}(e)$ for each edge e of G. Lemma 6 shows that ρ can be replaced by a different path ρ' that has the same letter-content as ρ, where ρ' has the form specified by Part B. Corresponding to ρ' is a different solution f' that is described by a specification of the form (Part A + Part B), together with the numbers of times various short loops are traversed (which implicitly describes the solution f' in the various gaps).

In this way we are able to solve the problem in FPT time by branching on at most $O(k^{O(k^{O(k^2)})})$ solution plans, and in each case checking feasibility by the solvability of a system of linear equations.

7 Summary

What we show in this paper is an example of how to systematically explore a "row" of the parameterized complexity ecology matrix, through the example of bounded *max leaf number*.

This is mostly an idea paper, that raises many more questions than it answers. There are clearly many more rows of the matrix to be explored. One would like to see further development of systematic approaches as well as improved concrete results.

Acknowledgements. We thank Fedor Fomin and Anil Nerode for useful conversations about this project. The research of M. Fellows and F. Rosamond has been supported by the Australian Research Council through the Australian Centre for Bioinformatics, by the University of Newcastle Parameterized Complexity Research Unit under the auspices of the Deputy Vice-Chancellor for Research, and by a Fellowship to the Durham University Institute for Advanced Studies. Fellows and Rosamond gratefully acknowledge the hospitality provided by a William Best Fellowship at Grey College, Durham, while the paper was in preparation, and thank the Alexander von Humboldt Foundation for support while the paper was revised and completed. The research of S. Saurabh was supported in part by the Norwegian Research Council. Preliminary versions of some of the results in this paper have previously appeared in the conference publications: "FPT is P-Time Extremal Structure I," *Proc. ACiD 2005* by V. Estivill-Castro, M. Fellows, M. Langston and F. Rosamond, and "The Complexity Ecology of Parameters: An Illustration Using Bounded Max Leaf Number," *Proc. CiE 2007* by M. Fellows, F. Rosamond and S. Saurabh. We thank an anonymous referee for substantially improving our presentation of these ideas and results.

References

[ACFLSS04] F. N. 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 Workshop on Algorithm Engineering and Experiments (ALENEX)*, New Orleans, January, 2004, ACM/SIAM, *Proc. Applied Mathematics 115*, L. Arge, G. Italiano and R. Sedgewick, eds.

[AD08] Y. Aumann and Y. Dombb. Fixed structure complexity. *Proc. IWPEC 2008*, Springer-Verlag, *Lecture Notes in Computer Science* 5018 (2008), 30–42.

[AFN04] J. Alber, M. Fellows and R. Niedermeier. Polynomial time data reduction for dominating set. *Journal of the ACM* 51 (2004), 363–384.

[AN02] J. Alber and R. Niedermeier. "Improved Tree Decomposition Based Algorithms for Domination-Like Problems," *Proceedings of the 5th Latin American Theoretical IN-formatics (LATIN 2002)*, Springer-Verlag LNCS 2286 (2002), 613–627.

[ALS91] S. Arnborg, J. Lagergren and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms* 12 (1991), 308–340.

[BEF+06] K. Burrage, V. Estivill-Castro, M. Fellows, M. Langston, S. Mac and F. Rosamond. The undirected feedback vertex set problem has polynomial kernel size. *Proceedings IWPEC 2006*, Springer-Verlag, *Lecture Notes in Computer Science* 4169 (2006), 192–202.

[BFH94] H. Bodlaender, M. Fellows and M. Hallett. Beyond NP-completeness for problems of bounded width: hardness for the W hierarchy. *Proceedings of the ACM Symposium on the Theory of Computing (STOC)* (1994), 449–458.

[BK07] H. L. Bodlaender and A. M. Koster. Combinatorial optimisation on graphs of bounded treewidth. *The Computer Journal* 51 (2007), 255–269.

[Bod96] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small width. *SIAM J. Computing* 25 (1996), 1305–1317.

[Bod07] H. L. Bodlaender. A cubic kernel for feedback vertex set. *Proceedings STACS 2007*, Springer-Verlag, *Lecture Notes in Computer Science* 4393 (2007), 320–331.

[BRST91] D. Bienstock, N. Robertson, P. Seymour and R. Thomas. Quickly excluding a forest. *J. Combinatorial Theory B* 52 (1991), 274–283.

[CCDF97] L. Cai, J. Chen, R. Downey and M. Fellows. The parameterized complexity of short computation and factorization. Proceedings of the *Sacks Symposium*, in *Archive for Mathematical Logic* 36 (1997), 321–338.

[CCF+06] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation* 201 (2005), 216–231.

[CFJ04] B. Chor, M. Fellows and D. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. *Proceedings WG 2004*, Springer-Verlag, *Lecture Notes in Computer Science* 3353 (2004), 257–269.

[CJ08] The Computer Journal: Two special issues of surveys of various aspects of parameterized complexity and algorithmics. (Guest editors: M. Fellows, R. Downey and M. Langston.) *The Computer Journal* Volume 51: Numbers 1,3 (2008).

[CKX05] J. Chen, I. Kanj and G. Xia. Improved Parameterized Upper Bounds for Vertex Cover. *Proceedings MFCS 2006*, Springer-Verlag, *Lecture Notes in Computer Science* 4162 (2006), 238–249.

[Cou90] B. Courcelle. The monadic second order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation* 85 (1990), 12–75.

[DEF+03] R. Downey, V. Estivill-Castro, M. Fellows, E. Prieto-Rodriguez and F. Rosamond. Cutting up is hard to do: the parameterized complexity of k-cut and related problems. *Electron. Notes Theor. Comp. Sci.* 78 (2003), 205–218.

[DF95a] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness I: basic results. *SIAM J. Computing* 24 (1995), 873–921.

[DF95b] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: on completeness for W[1]. *Theoretical Computer Science* 141 (1995), 109–131.

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[DFHKW94] R. Downey, M. Fellows, M. Hallett, B. Kapron 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.

[DFHT05] E. D. Demaine, F. V. Fomin, M. Hajiaghayi and D. M. Thilikos. Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. *Journal of the ACM* 52 (2005), 866–893.

[DFS99] R. Downey, M. Fellows and U. Stege. Parameterized complexity: a framework for systematically confronting computational intractability. In: *Contemporary Trends in Discrete Mathematics* (R. Graham, J. Kratochvil, J. Nesetril and F. Roberts, eds.), Proceedings of the DIMACS-DIMATIA Workshop on the Future of Discrete Mathematics, Prague, 1997, *AMS-DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 49 (1999), 49–99.

[DH05] E. D. Demaine and M. Hajiaghayi. Bidimensionality: New connections between FPT algorithms and PTASs. *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2005)*, Vancouver, January 2005, pp. 590–601.

[DH07] E. D. Demaine and M. Hajiaghayi. The bidimensionality theory and its algorithmic applications. *The Computer Journal* 51 (2008), 292–302.

[EFLR05] V. Estivill-Castro, M. Fellows, M. Langston and F. Rosamond. Fixed-parameter tractability is P-time extremal structure theory I: The case of max leaf. *Proceedings of ACiD 2005: Algorithms and Complexity in Durham* (2005), 1–41.

[Fe02] M. Fellows. Parameterized complexity: the main ideas and connections to practical computing. In: *Experimental Algorithmics*, Springer-Verlag, *Lecture Notes in Computer Science* 2547 (2002), 51–77.

[Fe03] M.R. Fellows. Blow-ups, win/win's and crown rules: Some new directions in FPT. *Proceedings WG 2003*, Springer-Verlag, *Lecture Notes in Computer Science* 2880 (2003), 1–12.

[FFG01] J. Flum, M. Frick and M. Grohe. Query evaluation via tree-decompositions. *Proc. ICDT*, Springer-Verlag, *Lecture Notes in Computer Science* 1973 (2001), 22–32.

[FFL+07] M. Fellows, F. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider and C. Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Proceedings of COCOA 2007*, Springer-Verlag, *Lecture Notes in Computer Science* 4616 (2007), 366–377.

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

[FKW04] F. Fomin, D. Kratsch and G. Woeginger. Exact (exponential) algorithms for the dominating set problem. *Proceedings of WG 2004*, Springer-Verlag, *Lecture Notes in Computer Science* 3353 (2004), 245–256.

[FL89a] M. Fellows and M. A. Langston. An analogue of the Myhill-Nerode theorem and its use in computing finite-basis characterizations. *Proceedings Thirtieth IEEE Symposium on the Foundations of Computer Science (FOCS)* (1989), 520–525.

[FL89b] M. Fellows and M. A. Langston. On search, decision and the efficiency of polynomial-time algorithms. In: *Proc. Symp. on Theory of Computing (STOC)*, 1989, 501–512.

[FLMRS08] M. Fellows, D. Lokshtanov, N. Misra, F. Rosamond and S. Saurabh. Graph layout problems parameterized by Vertex Cover. To appear, *Proceedings* ISAAC 2008.

[GM99] M. Grohe and J. Marino. Definability and descriptive complexity on databases with bounded treewidth. *Proceedings of the 7th International Conference on Database Theory*, Springer-Verlag, *Lecture Notes in Computer Science* 1540 (1999), 70–82.

[GN07] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, March 2007, 31–45.

[Gr01] M. Grohe. The parameterized complexity of database queries. *Proc. PODS 2001*, ACM Press (2001), 82–92.

[HK70] M. Held and R. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research* 18 (1970), 1138–1162.

[HM91] F. Henglein and H. G. Mairson. The complexity of type inference for higher-order typed lambda calculi. *Proc. Symp. on Principles of Programming Languages (POPL)*, ACM Press (1991), 119–130.

[IP01] R. Impagliazzo and R. Paturi. Which problems have strongly exponential complexity? *J. Computer and Systems Sciences* 63 (2001), 512–530.

[KTU94] A. J. Kfoury, J. Tiuryn and P. Urzyczyn. An analysis of ML typability. *J. ACM* 41 (1994), 368–398.

[KW91] D. J. Kleitman and D. B. West. Spanning trees with many leaves. *SIAM J. Discrete Mathematics* 4 (1991), 99–106.

[Nie04] R. Niedermeier. Ubiquitous parameterization — invitation to fixed-parameter algorithms. In: *Mathematical Foundations of Computer Science MFCS 2004*, Springer-Verlag, *Lecture Notes in Computer Science* 3153 (2004), 84–103.

[Nie06] R. Niedermeier. *Invitation to Fixed Parameter Algorithms.* Oxford University Press, 2006.

[NP85] J. Nesetril and S. Poljak. On the complexity of the subgraph problem. *Commun. Math. Univ. Carol.* 26 (1985), 415–419.

[NT75] G. L. Nemhauser and L. E. Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming* 8 (1975), 232–248.

[Pr05] E. Prieto-Rodriguez. *Systematic kernelization in FPT algorithm design.* Ph.D. Thesis, School of EE&CS, University of Newcastle, Australia, 2005.

[PS03] E. Prieto and C. Sloper. Either/Or: using vertex cover structure in designing FPT algorithms — the case of k-internal spanning tree. *Proc. WADS'03*, Springer-Verlag, *Lecture Notes in Computer Science* 2748 (2003), 474–483.

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

[RS85] N. Robertson and P. Seymour. Graph minors: a survey. In: J. Anderson, ed., *Surveys in Combinatorics*, Cambridge University Press (1985), 153–171.

[RS04] N. Robertson and P. Seymour. Graph minors XX. Wagner's conjecture. *J. Comb. Th. Series B* 92 (2004), 325–357.

[Sz08a] S. Szeider. Monadic second order logic on graphs with local cardinality constraints. *Proc. MFCS 2008*, Springer-Verlag, 601-612.

[Sz08b] S. Szeider. Not so easy problems for tree decomposable graphs. *Proc. ICDM 2008*, to appear.

[TP93] J.A. Telle and A. Proskurowski. "Practical Algorithms on Partial k-Trees with an Application to Domination-Like Problems." *Proceedings WADS'93 – The Third Workshop on Algorithms and Data Structures*, Springer-Verlag LNCS 709 (1993), 610–621.

[Wei98] K. Weihe. Covering trains by stations, or the power of data reduction. *Proc. ALEX'98* (1998), 1–8.