THE ALGORITHMICS COLUMN

BY

JOSEP DÍAZ

Department of Languages and Computer Systems
Polytechnical University of Catalunya
c/ Jodi Girona 1-3, 080304 Barcelona, Spain
diaz@lsi.upc.es

Layout problems form a family of problems which seem to be difficult to solve efficiently. In the present column, D. Thilikos and M. Serna address the parameterized complexity of graph layout problems. They survey the unifying methodologies yielding fixed parameter tractability when the layout measure is taken as parameter and collect a series of open questions related to its parameterized complexity.

PARAMETERIZED COMPLEXITY FOR GRAPH LAYOUT PROBLEMS*

Maria Serna Dimitrios M. Thilikos

1 Introduction

A relevant class of combinatorial optimization problems is defined by means of a measure over a liner layout (or ordering) of the elements of a graph. A linear

*Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Campus Nord – Edifici Ω, c/Jordi Girona Salgado 1-3, E-08034, Barcelona, Spain. This work was supported by the spanish CICYT project TIC2002-04498-C05-03. The work of the second author was also supported by the spanish CICYT project TIN-2004-07925 (GRAMMARS). Emails: {mjserna,sedthilk}@lsi.upc.edu

layout is a labeling of the vertices or edges of a graph with distinct integers. The goal of the problem is to find a layout for which a certain function is optimized. For most of the functions the derived graph layout problem is NP-hard. On the other hand, a large number of problems in different domains can be formulated as graph layout problems (see for example [21]). This fact makes clear the need to overcome the NP-hardness using other algorithmic approaches like approximation and parameterization.

There are many surveys that deal with different aspects of graph layout problems [63, 4, 54, 21]. We refer the interested reader to them for information on the family of problems and their history. We want to emphasize the richness of the use of linear layouts in the definition of problems, including in this family those problems that can be defined from a linear layout of the edges of a graph. Notice that usually only linear layouts of the vertices are considered [21]. In this paper we address the parameterized complexity of graph layout problems when the value of the function to be optimized is taken as the parameter. Our aim is to present the techniques that are common to several graph layout problems and motivate the research in such problems presenting a series of interesting open problems.

The paper is organized as follows: First of all, in Section 2 we review informally the main concepts of parameterized complexity used in the paper. In Section 3 we formally define the layout measures and graph layout problems and introduce the basic parameterized problem. Section 4 surveys the existing results on the design of FPT-algorithms focusing on the techniques that are common to a big sub-family of problems. Section 5 states the known parameterized hardness result. We finish with several open questions in Section 7.

2 Parameterized complexity

The theory of NP-completeness seems to be one of the greatest achievements in Theoretical Computer Science during the last 35 years. In particular, it offered a solid background for characterizing and investigating the "hardness" of combinatorial problems [43]. However, for practical purposes, such a theory seems to introduce a rather pessimistic viewpoint as the majority of natural non-trivial combinatorial problems seems to be NP-hard and thus one cannot expect that they admit an efficient (i.e. polynomial time) algorithmic solution. However, a more optimistic point of view can be adopted if we take in mind that the NP-completeness concerns only the *worst case complexity* of a combinatorial problem. In many real applications, the inputs of a generally intractable problem may be structured or restricted in a way that makes them tractable in practice. This motivated the idea of splitting the input of a combinatorial problem into two parts: the *main part* and the *parameterized* part. The splitting should be done in a way that the size of

the parameterized part is "small" in the majority of the "real word" applications while the main part is the one that includes elements of the problem that can be really big. The hope in this splitting is that we may be able to design algorithms with complexities whose super-polynomial part is *exclusively* depending on the "small" parameterized part. In other words, when we fix the parameterized part to be of constant size then the problem becomes tractable. If this is possible, then we may consider such a problem "tractable in practice" as it is easy to solve it in most of the cases where we may require a solution. This idea motivated what is now called *Parameterized Complexity*, a theory that during the last 16 years offered a solid and attractive alternative for investigating the hardness of combinatorial problems for from both algorithmic and complexity point of view.

Formally, a parameterized problem has as instances pairs (I, K) where I is the main part and K is the parameterized part. We use the notation $n = |I|$ and $k = |K|$ for the sizes of I and K respectively. The Parameterized Complexity settles the question of whether the problem is solvable by an algorithm (we call it FPT-*algorithm*) of time complexity

$$f(k) \cdot n^{O(1)}$$

where $f(k)$ is a (super-polynomial) function that does not depend on n. If such an algorithm exists, we say that the parameterized problem belongs in the class FPT. In a series of fundamental papers (see [24, 25, 1, 22, 23]), Downey and Fellows invented a series of complexity classes, namely the classes

$$\mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[SAT] \subseteq \mathsf{W}[P]$$

and proposed special types of reductions such that hardness for some of the above classes makes it rather impossible that a problem belongs in FPT (we stress that $\mathsf{FPT} \subseteq \mathsf{W}[1]$). The above theoretical framework initiated the classification of several parameterized problems according to their parameterized complexity. As it is expected in such a project, special attention has been given to parameterizations of problems that emerge from practical applications where *fast* (or "as fast as possible") solutions where really wanted. Parameterized complexity offered insightful results in a great variety of research areas like VLSI-design [37, 34, 35], Robot Motion Planning [17], Data Bases [45, 29, 67], Logical Programming, [56] and others (see also [33, 28, 27, 26]). So far, the most complete list of parameterized problems along with their classification according to their parameterized complexity is the compendium of Marco Cesati [16] including more than 200 problems reflecting the huge amount of work that has been devoted on this theory during the last two decades.

3 A generic definition of linear layout parameters

Given a set S, a *linear layout* (or *ordering*) of S is a bijection $L : S \to \{1, \ldots, |S|\}$. We denote as $\mathcal{L}_S$ the set of all the linear orderings of set S.

We call *graph parameter* any function mapping graphs to non-negative integers. We provide a generic definition for most of the graph parameters that we will meet in this paper. All of them are defined from a cost function defined over a linear layout of the set of vertices $V(G)$ or edges $E(G)$ of a graph G.

Formally, a graph parameter **par** is determined by a quadruple

$$(U, R, \mathbf{cost}, \lambda)$$

where $U, R \in \{V(G), E(G)\}$, $\mathbf{cost}_{G,L} : R \to \mathbb{N}$ is a function that depends on G and the selected layout $L \in \mathcal{L}_U$, and $\lambda \in \{\mathsf{max}, \mathsf{sum}\}$ is a label. The value associated to graph G is the following

$$\mathbf{par}(G) = \begin{cases} \min_{L \in \mathcal{L}_U} \max_{r \in R} & \mathbf{cost}_{G,L}(r) \quad \text{if } \lambda = \mathsf{min} \\ \min_{L \in \mathcal{L}_U} \sum_{r \in R} & \mathbf{cost}_{G,L}(r) \quad \text{if } \lambda = \mathsf{sum} \end{cases}$$

When $\lambda = \mathsf{max}$, we say that **par** is a *min-max* parameter and when $\lambda = \mathsf{sum}$, we say that **par** is a *min-sum* parameter. When $U = V(G)$ we say that **par** is a *vertex layout parameter* and when $U = E(G)$ we say that **par** is an *edge layout parameter*.

Let us present the definitions of the main graph parameters:

- **Bandwidth** (in short **bw**): $(V(G), E(G), \mathbf{cbw}, \mathsf{max})$ where for any $L \in \mathcal{L}_{V(G)}$ and any $e = \{v, u\} \in E(G)$,

$$\mathbf{cbw}_{G,L}(e) = |L(v) - L(u)|.$$

- **Cutwidth** (in short **cw**): $(V(G), V(G), \mathbf{ccw}, \mathsf{max})$ where for any $L \in \mathcal{L}_{V(G)}$ and any $v \in V(G)$,

$$\mathbf{ccw}_{G,L}(v) = |\{\{u, w\} \in E(G) \mid L(u) \leq L(v) < L(w)\}|.$$

- **Modified Cutwidth** (in short **mcw**): $(V(G), V(G), \mathbf{cmcw}, \mathsf{max})$ where for any $L \in \mathcal{L}_{V(G)}$ and any $v \in V(G)$,

$$\mathbf{cmcw}_{G,L}(v) = |\{\{u, w\} \in E(G) \mid L(u) < L(v) < L(w)\}|.$$

- **Vertex Separation** (in short **vs**): $(V(G), V(G), \mathbf{cvs}, \mathsf{max})$ where for any $L \in \mathcal{L}_{V(G)}$ and any $v \in V(G)$,

$$\mathbf{cvs}_{G,L}(v) = |\{u \in V(G) \mid L(u) \leq L(v) \text{ and } N_G(u) \cap \{w \in V(G) \mid L(w) \geq L(v)\} \neq \emptyset\}|.$$

(By $N_G(u)$ we denote the set of vertices in G that are adjacent to the vertex $u \in V(G)$.)

The vertex separation number of a graph is equivalent (see [50, 51, 64]) to the parameter of **pathwidth**, defined in [69] as follows:

A *path decomposition* of a graph G is defined as a sequence $X = [X_1, \ldots, X_r]$ of subsets of $V(G)$ satisfying the following properties.

1. $\cup_{i,1\leq i\leq r} X_i \subseteq V(G)$.
2. $\forall_{\{v,u\}\in E(G)}\ \exists_{i,1\leq i\leq r}\ \{v,u\} \subseteq X_i$.
3. $\forall_{v\in V(G)}\ \exists_{i,j,1\leq i\leq j\leq r}\ \forall_{h,1\leq h\leq r}\ v \in X_h \Leftrightarrow i \leq h \leq j$.

We call the sets $X_1, \ldots, X_r$, the *nodes* of the path decomposition X. The *width* of X is equal to $\max_{1\leq i\leq r}\{|X_i| - 1\}$ and the *pathwidth* of a graph G (we denote it as $\mathbf{pw}(G)$) is the minimum width over all path decompositions of G.

- **Treewidth** (in short **tw**): $(V(G), V(G), \mathbf{ctw}, \mathsf{max})$ where for any $L \in \mathcal{L}_{V(G)}$ and $v \in V(G)$,

  $$\mathbf{ctw}_{G,L}(v) = |\{u \in V(G) \mid \exists \text{ a } v\text{-}u \text{ path in } G_L(u, \geq)\}|,$$

  where $G_L(u, \geq) = G[\{u\} \cup \{w \in V(G) \mid L(w) \geq L(v)\}]$. We stress that treewidth dates back to [44] and has been defined in [71] generalizing path decompositions to tree decompositions. For a proof of the equivalence of the decomposition definition and the layout definition, see [19].

- **Linear-width** (in short **lw**): $(E(G), E(G), \mathbf{clw}, \mathsf{max})$ where for any $L \in \mathcal{L}_{E(G)}$ and $e \in E(G)$,

$$\mathbf{clw}_{G,L}(e) = |\{v \in V(G) \mid \exists e', e'' \in E(G) : e' \cap e'' = v \text{ and } L(e') \leq L(e) < L(e'')\}|.$$

- **Edge-Bandwidth** (in short **ebw**): $(E(G), V(G), \mathbf{cebw}, \mathsf{max})$ where for any $L \in \mathcal{L}_{E(G)}$ and $v \in V(G)$,

  $$\mathbf{cebw}_{G,L}(v) = \max\{|L(e) - L(e')| \mid e \cap e' = \{v\}\}.$$

  Also we may comment the following old parameter, also known as *degeneracy*, *linkage*, or *inductivity* (see [61, 73, 57, 62, 40, 52]).

- **Width** (in short **wd**): $(V(G), V(G), \mathbf{cwd}, \mathsf{max})$ where for any $L \in \mathcal{L}_{V(G)}$ and $v \in E(G)$,

  $$\mathbf{cwd}_{G,L}(v) = |N_G(v) \cap \{u \mid L(u) < L(v)\}|.$$

Problem	Introduced in	NP-hardness
Bandwidth	[46]	[66]
Cutwidth	[2]	[44]
Modified Cutwidth	[65]	[65]
Vertex separation	[59]	[55]
Pathwidth	[69]	[3]
Treewidth	[71]	[3]
Linear-width	[80]	[75]
Min-Linear Arrangement	[47]	[42]
Profile	[20, 58, 53]	[20, 58]
Edge-bandwidth	[48]	[81]

Table 1: Basic references for graph layout problems

Although a min-sum version of all the previous graph parameters can be defined, only for few of them the corresponding measure has been considered in the literature.

- **Min-Linear-Arrangement** (in short **mla**): $(V(G), E(G), \mathbf{cbw}, \mathsf{sum})$ or $(V(G), E(G), \mathbf{ccw}, \mathsf{sum})$.
- **Profile** (in short **prf**): $(V(G), V(G), \mathbf{cvs}, \mathsf{sum})$ or $(V(G), V(G), \mathbf{cprf}, \mathsf{sum})$ where for any vertex $v \in V(G)$,

$$\mathbf{cprf}_{G,L}(e) = L(v) - \min\{i \mid 1 \leq i < L(V) \text{ and } \{L^{-1}(i), v\} \in E(G)\}.$$

If **par** is a graph parameter, the corresponding optimization problem is defined as follows:

Optimization Problem for Parameter **par**
Input: A graph G.
Objective: Find a layout L whose associated cost equals **par**(G).

In Table 1 we provide references to the work that introduced the optimization problem and to the work showing its NP-hardness[1]. We refer the reader to [63, 4, 54, 21] for further information.

Notice that the variety of parameters that can be defined in this way is quite extensive, and can be further extended. Also one may suspect that, for most of

[1]The NP-hardness of dual bandwidth follows from Theorem 5.2 of [81], because for caterpillars with max degree 3, dual bandwidth is a constant factor approximation of bandwidth.

them, the problem of deciding whether their values are at most k is NP-hard. However we stress that this is not always the case. For example, $\mathbf{wd}(G)$ can be easily computed in polynomial time. Notice also that the min-sum version of **wd** is a trivial graph parameter.

If **par** is a graph parameter the corresponding parameterized problem is defined as follows:

PARAMETERIZED PROBLEM FOR PARAMETER **par**
Input: A graph G and a non-negative integer k
Parameter: k
Question: $\mathbf{par}(G) \leq k$?

We denote the first 6 parameterized problems that emerge from min-max parameters as k-BANDWIDTH, k-CUTWIDTH, k-MODIFIED CUTWIDTH, k-VERTEX SEPARATION, k-TREEWIDTH, and k-LINEAR-WIDTH respectively. We will postpone the discussion on the parameterized problem for min-sum parameters to Subsection 6.2 where we will propose a different parameterization.

4 Designing FPT-algorithms

In this section we survey generic tools that have been used to show fixed parameter tractability for the problems derived from min-max graph layout parameters.

4.1 Non-constructive tools

A powerful tool for proving the existence of an FPT-algorithm for some graph parameter was given by the theory of Graph Minors developed by Robertson and Seymour for proving the Wagner's Conjecture.

To give a general description of the consequences of this theory on the existence of FPT-algorithms, we consider several types of partial ordering relations on graphs. Usually such a relation is defined by a set of graph operations such as vr: vertex removal (and removal of all edges incident to this vertex), er: edge removal, ec: edge contraction (i.e. the removal of an edge and the identification of its endpoints), et: edge contraction of an edge that has some endpoint of degree ≤ 2, or le: lifting of a pair of adjacent edges (i.e. the removal of two edges $\{v,u\}$, $\{v,w\}$ that share an endpoint and the addition of the edge $\{u,w\}$). Given a set P of graph operations we say that $H \leq_{\mathsf{P}} G$ if H can be obtained from G after applying to it a series of operations in P.

In the following table we show how known partial orderings can be defined using different choices of P. In the same table we also introduce the notation used in this paper.

The meaning of $H \leq_{\mathsf{P}}$ when P=		notation
H is an induced subgraph of G	{vr}	$\leq_{is}$
H is a subgraph of G	{vr, er}	$\leq_{sb}$
H is topologically contained in G	{vr, er, et}	$\leq_{tp}$
H is a minor of G	{vr, er, ec}	$\leq_{mn}$
H can be immersed in G	{vr, er, le}	$\leq_{im}$

Given a partial ordering $\leq_*$ and a graph set $\mathcal{G}$ we denote $\mathsf{ob}_{\leq_*}(\mathcal{G})$ the set of all the minimal (with respect to $\leq_*$) elements of the set containing any graph not in $\mathcal{G}$. We call $\mathsf{ob}_{\leq_*}(\mathcal{G})$ *obstruction set of* $\mathcal{G}$ *with respect to the relation* $\leq_*$. A very deep and general question in graph theory asks whether, given a pair $(\mathcal{G}, \leq_*)$, the set $\mathsf{ob}_{\leq_*}(\mathcal{G})$ is finite or not. We call a partial ordering *simple* if for any set of graphs $\mathcal{G}$, the set $\mathsf{ob}_{\leq_*}(\mathcal{G})$ is finite.

Another interesting question is whether for some partial ordering $\leq_*$ and a graph class $\mathcal{G}$, there is an algorithm that, given a graph G of $\mathcal{G}$ as an input, checks whether $H \leq_{mn} G$ in polynomial (linear) on $|V(G)|$ time. If this holds, we say that $\leq_*$ is *polynomially (linearly) checkable* relation on $\mathcal{G}$. In case of polynomially checkable orderings, if the degree of this polynomial does not depend on H we say that $\leq_*$ is *uniformly polynomially checkable on* $\mathcal{G}$.

We say that a graph class $\mathcal{G}$ is $\leq_*$*-closed* if $G \in \mathcal{G}$ and $H \leq_* G$ implies that $H \in \mathcal{G}$.

It is now easy to prove the following:

Theorem 1. *Let $\leq_*$ be a simple and polynomially checkable relation and let $\mathcal{G}$ be a $\leq_*$-closed graph class. Then the problem of deciding whether a graph G belongs in $\mathcal{G}$ can be solved in polynomial time.*

According to Theorem 1, looking for partial orderings that are simple and polynomially checkable can be very helpful for massively classifying combinatorial problems in the class P of polynomially solvable problems. One of the greatest contributions of the Graph Minors Theory on algorithmic design was to prove that $\leq_{mn}$ and $\leq_{im}$ are simple partial orderings, while there are counterexamples showing that the same does not hold for $\leq_{is}, \leq_{sb}$ and $\leq_{tp}$ (see [70]). Also, according to the same theory, $\leq_{mn}$ is uniformly polynomially checkable (by an $O(n^3)$ algorithm). However, it is open whether the same holds also for $\leq_{mn}$. Indeed, according to [37, 38], $\leq_{im}$ is polynomially checkable for all graphs (by a $O(n^{|V(H)|+3})$ algorithm), however it is still an open question whether $\leq_{mn}$ is uniformly polynomially checkable on the same class.

From the above remarks, it follows that if we prove that the YES-instances (or the NO-instances) of some combinatorial problem on graphs form a $\leq_{mn}$-closed or a $\leq_{im}$-closed graph class, we also classify the problem in P. We will now comment

the consequences of this fact to the design of FPT-algorithms (notice that, so far, we did not demand our relations to be uniformly polynomially checkable).

A graph invariant **par** is $\leq_*$-closed if for any integer $k \geq 0$, the set $\mathcal{G}(\mathbf{par}, k) = \{G \mid \mathbf{par}(G) \leq k\}$ is $\leq_*$-closed. Then Theorem 1 can be rewritten as follows:

Theorem 2. *Let $\leq_*$ be a simple and uniformly polynomially checkable (on all graphs) relation and let* **par** *be a $\leq_*$-closed graph invariant. Then, the problem of deciding whether* $\mathbf{par}(G) \leq k$*, parameterized by k, belongs in* FPT.

Notice that the above result does not follow without the uniformity demand. Indeed, the only we have in such a case is the existence of a a $O(n^{f(k)})$ algorithm that is polynomial for any fixed value of k. Especially for $\leq_{im}$ it is known that is it uniformly linearly checkable when restricted to graphs of bounded treewidth (or pathwidth). In fact, there is a $O(f(k, |V(H)|) \cdot n)$ algorithm that checks whether a fixed graph H is a minor of (can be immersed into) a graph G of treewidth at most k (see [68, 14, 18]).

So, if we want to show that Parameterized Problem for Parameter **par** is in FPT it is enough to prove that **par** is $\leq_{mn}$-closed or that **par** is $\leq_{im}$-closed and $\mathbf{par}(G) \geq \mathbf{pw}(G)$ (or $\mathbf{par}(G) \geq \mathbf{tw}(G)$). Indeed, it is possible to prove that this holds for many of the parameters defined in Section 3. In particular, cutwidth is $\leq_{im}$-closed and $\mathbf{cw}(G) \leq \mathbf{pw}(G)$, while vertex separation, treewidth, and linear width are $\leq_{mn}$-closed.

Therefore, we conclude to the following:

Corollary 1. *The* Parameterized Problem for Parameter **par** *belongs to* FPT *for* **par** $\in$ { cutwidth, vertex separation, treewidth, linear-width }. *Moreover, the corresponding* FPT*-algorithms are linear on the size of the input graph.*

Unfortunately, the result of Robertson and Seymour is *non-constructive* in the sense that it does not give any systematic method of constructing the corresponding obstruction set. Therefore, Theorems 1 and 2 only guarantee the *existence* of the corresponding algorithms but their proofs do not provide a way to construct them (see [82, 36, 41]). However, this provides a strong motivation towards finding the corresponding algorithms for a wide range of graph classes and parameters. It seems that if one knows that an algorithm with certain complexity exists, this makes it easier to search for a way to construct it. We will devote the next section to the construction issue.

4.2 The automaton idea

In this section we resume a general method developed, in [9], for proving that an FPT-algorithm can be constructed for a series of min-max layout parameters.

In particular we present the construction of such an algorithm for the case of cutwidth. Later, we will discuss how and when the presented ideas can be used to obtain FPT-algorithms for other parameters.

The main idea resides in the construction, for any given $k \geq 0$, of a *finite state automaton* that decides whether a graph G has cutwidth $\leq k$. We present the idea gradually. First we give an easy, but non-efficient, solution and then we refine it until it becomes the desired one. Our automaton will have many final states. Given an automaton A, as usual, $\mathsf{L}(A)$ denotes the set of strings recognized by A.

Consider a graph $G = (V, E)$ and a set of integers $K = \{0, \ldots, k\}$. We assume that K and V are disjoint alphabets. For a given $L \in \mathcal{L}_V$, $w(L)$ is the string of V^* whose i-th character is the vertex in the i-th position in L, $i = 1, \ldots, |V|$. In our first automaton $A_{G,k}$, the input is a string representing an arbitrary ordering of the vertices of G. The states of our automaton are strings $q = k_0u_1k_1 \cdots u_rk_r$ consisting of alternating numbers and vertices representing layouts of cost at most k. We define the automaton $A_{G,k} = (Q, V, \delta, q_s, F)$ where $Q = \{w \mid w \in K(VK)^* \text{ and } |w| \leq 2|V| + 1\}$, $q_s = \{0\}$, $F = \{s \in Q \mid |s| = 2n + 1\}$, and for any $q = k_0u_1k_1 \cdots u_rk_r \in Q$ and $v \in V$,

$$\begin{aligned}
\delta(q, v) \quad &= \quad \{q' \mid q' = n'_0u'_1n'_1u'_2 \cdots u'_{r+1}n'_{r+1} \text{ where } q' \in Q \text{ and} \\
\exists_{i,0\leq i\leq r} : \ &\forall_{h=1,\ldots,i} u'_h = u_h \wedge u'_{i+1} = v \wedge \forall_{h=i+1,\ldots,i} u'_{h+1} = u_h \wedge \\
&\forall_{h=1,\ldots,i} n'_h = n_h + |\{\{v, u_j\} \mid \{v, u_j\} \in E \wedge j \leq h\}| \wedge \\
&\forall_{h=j+1,\ldots,r+1} n'_h = n_{h+1} + |\{\{v, u_j\} \mid \{v, u_j\} \in E \wedge j \geq h\}|\}
\end{aligned}$$

The initial state is the sequence containing only the number 0 and the set of final states contains all sequence of length $2|V| + 1$. Any state represents a way to order some prefix of the input. Let $q = k_0u_1k_1 \cdots u_rk_r$ be such a state and let v the new character that $A_{G,k}$ receives. The transition function "guesses" where the new vertex v should be inserted. The insertion duplicates some integer in the sequence representing the current state, then inserts v between the two copies of this integer and then for each edge $\{v, u_i\}$ of G increases by one all integers between the positions of u_i and v. Clearly, if after this operation all integers of the new state are at most k then the ordering guessed so far is an ordering of the graph $G[\{v_1, \ldots, v_r, v\}]$ of cutwidth $\leq k$. It is now easy to verify the following:

Lemma 1. *Let $G = (V, E)$ and $L \in \mathcal{L}_V$, then $\mathbf{cw}(G) \leq k$ iff $w(L) \in \mathsf{L}(A_{G,k})$.*

Clearly, the description of $A_{G,k}$ cannot be implemented efficiently as both V and Q have size that depend on G. To improve this, the first step is to modify $A_{G,k}$ so that its input alphabet has size that depends only on k. To achieve this goal we take into consideration the fact that $\mathbf{pw}(G) \leq l$ and we use the structure of the corresponding path-decomposition.

Suppose now that $\mathcal{X} = [X_1, \ldots, X_r]$ is a path decomposition of G of width $\leq l$. We say that $\mathcal{X} = [X_1, \ldots, X_r]$ is a *nice path decomposition* if $|X_1| = 1$ and $\forall_{i,2\leq i\leq|\mathcal{X}|}\ |(X_i - X_{i-1}) \cup (X_{i-1} - X_i)| = 1$. It is easy to see (see e.g. [10, 78]) that, for some constant l and given a path decomposition of a n-vertex graph G that has width at most l and $O(n)$ nodes, one can find a nice path decomposition of G that has width at most l and has at most $2n$ nodes in $O(n)$ time. We distinguish two types of nodes in a nice path decomposition $\mathcal{X} = [X_1, \ldots, X_r]$. We say that X_i is an *introduce* node if $i = 1$ or $|X_i - X_{i-1}| = 1$, while X_i is a *forget* node if $|X_{i-1} - X_i| = 1$. Clearly, any node X_i of a nice path decomposition is either an *introduce* or a *forget* node.

Let G be a graph and let $\mathcal{X} = [X_1, \ldots, X_r]$ be a nice path decomposition of G of width at most l. It is easy to construct an $(l+1)$-coloring $\chi : V \to \{\mathbf{1}, \ldots, \mathbf{l+1}\}$ of G such that vertices in the same X_i have always distinct colors (from now on, whenever we refer to these colors we will use bold characters).

If X_i is an introduce node then set $p(i) = \texttt{ins}(\mathbf{t}, \mathbf{S})$ where $\{\mathbf{t}\} = \chi(X_i - X_{i-1})$ and $\mathbf{S} = \chi(N_{G[X_{i-1}]}(v))$ (if $i = 1$ then $\mathbf{S} = \emptyset$). If X_i is a forget node then set $p(i) = \texttt{del}(\mathbf{t})$ where $\{\mathbf{t}\} = \chi(X_{i-1} - X_i)$.

Let $w(\mathcal{X})$ be a string whose i-th letter is $p(i)$, $i = 1, \ldots, r$. We define now an automaton that receives $w(\mathcal{X})$ as input (provided that a path decomposition $\mathcal{X}$ of G is given).

We set up a set of labels $\mathbf{T} = \{\mathbf{-}, \mathbf{1}, \ldots, \mathbf{l+1}\}$ where bold numbers represent colors and "$\mathbf{-}$" represents the absence of a vertex. Let also Σ be the set of all possible $\texttt{ins}(v, S)$ and $\texttt{del}(v)$ (notice that $|\Sigma| = O(l \cdot 2^{l+1})$, i.e. its size depends only on l).

We define the automaton $A_{G,l,k} = (Q', \Sigma, \delta, q_s, F)$ where $Q' = \{w \mid w \in K(\mathbf{T}K)^*, |w| \leq 2n+1$ and each label of $\mathbf{T}$ appears at most once in $w\}$, $q_s = [0]$, $F = \{w \in Q' \mid w = 2n+1\}$, and for any $q = n_0\mathbf{t}_1 n_1 \mathbf{t}_2 \cdots \mathbf{t}_r n_r \in Q'$ and $\texttt{ins}(\mathbf{t}, \mathbf{S}) \in \Sigma$ or $\texttt{del}(\mathbf{t}) \in \Sigma$,

$$\begin{aligned}
\delta(q, \texttt{ins}(\mathbf{t}, \mathbf{S})) \;=\; & \{q' \mid q' = n'_0\mathbf{t}'_1 n'_1 \mathbf{t}'_2 \cdots \mathbf{t}'_{r+1} n'_{r+1} \text{ where } q' \in Q' \\
\text{and } \exists_{i,0\leq i\leq r} :\; & \forall_{h=1,\ldots,i}\mathbf{t}'_h = \mathbf{t}_h \wedge \mathbf{t}'_{i+1} = \mathbf{t} \wedge \forall_{h=i+1,\ldots,i}\mathbf{t}'_{h+1} = \mathbf{t}_h \wedge \\
& \forall_{h=1,\ldots,i} n'_h = n_h + |\{\mathbf{t}_j \mid \mathbf{t}_j \in \mathbf{S} \wedge j \leq h\}| \wedge \\
& \forall_{h=j+1,\ldots,r+1} n'_h = n_{h+1} + |\{\mathbf{t}_j \mid \mathbf{t}_j \in \mathbf{S} \wedge j \geq h\}|\}
\end{aligned}$$

and

$$\begin{aligned}
\delta(C, \texttt{del}(\mathbf{t})) \;=\; & \{C' \mid C' = n_0\mathbf{t}_1 n_1 \mathbf{t}_2 \cdots n_{i-1}\mathbf{-}n_i \cdots \mathbf{t}_{r+1} n_{r+1} \\
& \text{where } \mathbf{t} = \mathbf{t}_i\}
\end{aligned}$$

Notice that the previous definition can be directly extracted from the definition of $A_{G,k}$, replacing the vertices by labels. The good news are that the number of labels

depends on the pathwidth of G and not on its size. Finally, what is really new in $A_{G,l,k}$ is the "delete transition" where the deleted color is just removed from the corresponding sequence. The following holds (see [9]).

Lemma 2. *Let $G = (V, E)$ be a graph and $\mathcal{X}$ be a path decomposition of G of width $\leq l$. Then* $\mathbf{cw}(G) \leq k$ *iff* $w(\mathcal{X}) \in \mathsf{L}(A_{G,l,k})$.

Notice that in the above automaton the definition of δ does not require any knowledge of G. This is because the adjacency information codified in the string $w(\mathcal{X})$ is now enough for the definition of δ. However, the number of states Q' depends still on the size of G. To explain how this problem can be overcome we give an example.

Suppose we have a substring **5**4–6–9**2** in some state of Q'. Then notice the following: If the automaton accepts the string $w(\mathcal{X})$ and during its operation enters this state and proceeds by "spliting" the number 6 then it will also accept the same string if it "splits" instead the number 4. This essentially means that it is not a problem if we "forget" 6 from this state. As a consequence of this observation, we can reduce the length of the strings in Q' by suitably "compressing" them (see [10, 11, 76, 79, 78]). To explain this we first need some definitions.

Let $w \in Q'$. We say that a z is a *portion* of w if its is a maximal substring of w that does not contain symbols from $\mathbf{T}$. We say that a portion of w is *compressed* if it does not contain a sub-sequence n_1–n_2–n_3 such that either $n_1 \leq n_2 \leq n_3$ or $n_1 \geq n_2 \geq n_3$. The operation of replacing in a portion any such a subsequence by n_1–n_3 is called *compression* of the portion. We also call *compression of* $w \in Q'$ the string that appears if we compress all portions of w. We define $\tilde{Q}$ as the set occurring from Q' if we replace each of its strings by their compressions. This replacement naturally defines a class of equivalence in Q' where two strings are equivalent if they have the same compression. That, way $\tilde{Q}$ defines a "*set of characteristics*" of Q' in the sense that it contains all the useful information that an automaton needs to operate. The good news is that the size of $\tilde{Q}$ is now depending *only* on l and k. Indeed, in [10, 78] it was proved that $|\tilde{Q}| \leq (l+1)!\frac{8}{3}2^{2k}$ and this makes it possible to construct an automaton that overcomes the big space drawbacks of the previous one.

In particular define the automaton $\tilde{A}_{l,k} = (\tilde{Q}, \Sigma, \tilde{\delta}, q_s, \tilde{Q})$. Here $\tilde{\delta} = \beta \circ \delta$ where β is the function that receives a set of members of Q' and outputs the set of all their compressions. The following holds:

Lemma 3. *Let $G = (V, E)$ be a graph and $\mathcal{X}$ be a path decomposition of G of width $\leq l$. Then* $\mathbf{cw}(G) \leq k$ *iff* $w(\mathcal{X}) \in \mathsf{L}(\tilde{A}_{l,k})$.

As we mentioned, the construction of the automaton $\tilde{A}_{l,k}$ depends only on l and k. Moreover, as the input is a string of length $O(n)$ the decision can be made

non-deterministically in linear time. As for every non-deterministic finite state automaton an equivalent deterministic one can be constructed (notice that the state explosion will be an exponential function on k and l), we deduce the following:

Theorem 3. *For any k, l, one can construct an algorithm that given an n-vertex graph G and a path decomposition of G of width $\leq l$, decides whether* $\mathbf{cw}(G) \leq k$ *in $O(f(k,l) \cdot n)$ steps where f is a function that depends only on l and k.*

Also, in [9] it is described how to turn the decision algorithm to an algorithm that, in case of a positive answer, also outputs the corresponding layout.

4.3 Extensions

Notice that the kernel of the construction $\tilde{A}_{l,k}$ is the transition function. In fact $\tilde{\delta}$ incorporates the way that the values encoded in its states of $\tilde{A}_{l,k}$ are being updated each time the automaton reads a `ins`(**t**,**S**) or a `del`(**t**) and this follows directly from the cost function of the graph invariant.

By modifying accordingly the construction of $\tilde{A}_{l,k}$ the result of Theorem 3 can be derived for other parameters. For vertex separation the values of each state of the automaton should now count, for each gap, instead of the overlapping edges, the number of their left-side endpoints. In case of modified cutwidth one may consider a slightly more complicated version of states where the integers are replaced by pairs of integers that encode not only the number of edges that cross gaps between vertices but also the number of edges that cross a vertex. The transition function should reflect the way that the values of a state are being altered when the automaton reads an `ins`(**t**,**S**) or a `del`(**t**). Avoiding the details, we resume to the following general result.

Theorem 4. *For any k, l, one can construct an $O(f(k,l) \cdot n)$ step algorithm (f is a function depending only on l and k) that, for an n-vertex graph G and a path decomposition of G of width $\leq l$, decides if the cutwidth/vertex separation/modified cutwidth of G is at most k and, if so, outputs a linear layout of $V(G)$ of cutwidth/vertex separation/modified cutwidth at most k.*

In [78] it is given a different proof of Theorem 4 for the case of cutwidth that does not use the automaton machinery. Instead it gives a more direct way to define and work with the notion of a "set of characteristic of layouts" that is something analogous –however more compact– to the information we keep in the states of our automaton. That way, also provides an estimation for function f that is $O(l^3k^2 \cdot (l+1)! \, (\frac{8}{3} \, 2^{2k})^{l+1})$ (see also [77]).

Recall that according to [50, 51] $\mathbf{pw}(G) = \mathbf{vs}(G)$. Also it is easy to see that, given a path decomposition of width $\leq k$, a layout of vertex separation number

$\leq k$ can be constructed in linear (on $n = |V(G)|$) time. This means that pathwidth can also be added in the list of parameters of Theorem 4.

For the case of vertex-separation the first proof of Theorem 4 was given in [10] in terms of pathwidth. This was the first time where the technique of sequence compression appeared. Combining the results of [10] along with the result of [8] one can prove the following (see also [12]).

Theorem 5. *For any k, one can construct an $O(f(k) \cdot n)$ step algorithm that, for an n-vertex graph G, decides whether $\mathbf{pw}(G) \leq k$, and, if so, outputs a path decomposition of G with width at most k.*

Notice that from the definitions of cutwidth and vertex separation (that, in turn is equal to pathwidth) it follows that $\mathbf{cw}(G) \geq \mathbf{pw}(G)$. Therefore, if we want to check if $\mathbf{cw}(G) \leq k$ then we can do the following: check first if $\mathbf{pw}(G) \leq k$ and if the answer is no then output that $\mathbf{cw}(G) > k$. Otherwise, we have a path decomposition of G of width $\leq k$ and we can check whether $\mathbf{cw}(G) \leq k$ applying Theorem 4 for cutwidth and for $l = k$. Also the same technique can be useful to check whether $\mathbf{mcw}(G) \leq k$ as $\mathbf{mcw}(G) \geq \mathbf{pw}(G)$ (this follows from [60] where it is proved that the topological bandwidth of a graph is never greater than its modified cutwidth plus one and never smaller than its node search number that is equal to the vertex separation number plus one). We resume with the following.

Theorem 6. *For any k, one can construct an $O(f(k) \cdot n)$ step algorithm that for an n-vertex graph G, decides whether the cutwidth/vertex separation/modified cutwidth of G is at most k and, in such a case, outputs a linear layout of $V(G)$ of cutwidth/vertex separation/modified cutwidth at most k.*

For the case of modified cutwidth, Theorem 6 also follows from Theorem 5 by a parameterized reduction of the k-MODIFIED CUTWIDTH problem to the k-PATHWIDTH problem presented in [9] (see also [39, 64]). Also for similar results (algorithms and reductions) concerning directed versions of vertex separator, cutwidth, and modified cutwidth see [9] and [7].

5 When we do not hope for FPT-algorithms

The only parameter where we have strong evidence that does not belong in FPTis bandwidth. The result appeared in [6] and the proof used a reduction from the following problem:

EMULATON IN A PATH *Input:* A graph G and a non-negative integer k Parameter: k Question: Is there a partition $\{V_1, \ldots, V_r\}$ of $V(G)$ where $\|V_i\| \leq k, i = 1, \ldots, r$ and for any edge $e \in E(G)$, there exists some m, $1 \leq m < r$ such that $e \subseteq V_m \cup V_{m+1}$?

We define the graph invariant emulation in a path as the function that outputs the minimum k for which G is a YES-instance of the EMULATON IN A PATH problem.

In [6] it was proved that EMULATON IN A PATH is W[P]-hard and this implies that PARAMETERIZED PROBLEM FOR PARAMETER **par** is W[P]-hard when **par**=bandwidth.

It seems that no hardness result exists on the parameterized complexity of other layout graph parameters. As several problems remain open, some of them may be candidates for hardness for some level of the parameterized complexity class hierarchy (see Subsections 6.2, 6.3, and 6.4).

6 Open problems

6.1 Atomata idea

One may ask what is the most general framework where the automaton idea can be applied. Actually, the only remaining parameterized problems among those defined in Section 3 that we know that belongs to FPT are treewidth and linear-width. Especially for linear-width we believe that there is an automaton-driven proof of its parameterized tractability. However, here the NFA should "guess" insertion operations concerning edges and the states should encode vertices that are incident to edges in both sides. A specific FPT-algorithm for checking whether $\mathbf{lw}(G) \leq k$ appeared in [13]. This algorithm defines a "set of characteristics" especially for linear-width that takes in mind the particularities in way the value of linear-width is being calculated. This fact makes us believe that an automaton-driven proof of the parameterized tractability of linear-width is possible but not as easy as in the case of the parameters considered in Theorem 6.

For treewidth it seems that the automaton-idea is not friendly because each time a vertex is introduced the update in the information encoded in the states is *global* in the sense that it cannot be restricted to the neighbors of the inserted vertex; it concerns connected components of the "so-far" considered graph. In fact this difficulty follows from the fact that guessing the position of new vertices in a linear layout cannot tame its tree-structure. This indicates that an automaton-driven proof of the parameterized tractability of treewidth requires the use of *tree automata*. For an linear FPT-algorithm for treewidth that uses the idea of a "set of

characteristics" see [10]. We stress that the technique of defining a "set of characteristics" is not exclusive for layout parameters and has been used for several other "tree-fashion" problems such as carving-widh [79] and branchwidth [11, 76].

6.2 Min-sum prameters

There is a long standing question on what is the correct way to parameterize problems on min-sum parameters such as min linear arrangement and profile. Clearly, the classic way should be to ask whether the value of the invariant is at most k. However for min linear arrangement, the answer to this question is directly NO if $|E(G)| > k$ otherwise, we have an equivalent instance of at most $2k$ vertices where the answer can be found using brute force whose cost depends only on k. Thus the "naive" parameterization of the problem is trivially in FPT(a similar argument also holds for profile). Therefore we suggest more rich parameterizations like the following:

VERTEX AVERAGE PARAMETERIZED PROBLEM FOR **par**
Input: A graph G and a non-negative integer k
Parameter: k
Question: $\mathbf{par}(G) \leq k \cdot \|V(G)\|$?

EDGE AVERAGE PARAMETERIZED PROBLEM FOR **par**
Input: A graph G and a non-negative integer k
Parameter: k
Question: $\mathbf{par}(G) \leq k \cdot \|E(G)\|$?

In particular the following three problems are of great interest:

EDDGE AVERAGE MIN LINEAR ARRANGEMENT
Input: A graph G and a non-negative integer k
Parameter: k
Question: $\mathbf{mla}(G) \leq k \cdot \|E(G)\|$?

VERTEX AVERAGE MIN LINEAR ARRANGEMENT
Input: A graph G and a non-negative integer k
Parameter: k
Question: $\mathbf{mla}(G) \leq k \cdot \|V(G)\|$?

VERTEX AVERAGE PROFILE
Input: A graph G and a non-negative integer k
Parameter: k
Question: $\mathbf{prf}(G) \leq k \cdot \|V(G)\|$?

Notice that the YES-instances of the above problem are "almost" immersion closed. Indeed they are closed under the operations of edge lifting and edge removal. However, they are not closed under vertex removal.

The parameterized complexity of the three problems defined in this subsection remain a mystery as it seems that none of the current tools is able to give some evidence. In fact we will even avoid to make any conjecture on whether they are fixed parameter tractable or not. However, we would conjecture that they all have the same type of parameterized complexity.

6.3 Topology vs Geometry

There are some parameters that are defined using layouts and local transformations. We will give two examples of such parameters.

A *subdivision* of an edge e is the replacement of e by a path of length 2. We say that a graph H is a *subdivision* of G if H can be obtained by G after a finite sequence of subdivision operations on its edges. We denote as $\mathbf{SD}(G)$ the set of all the subdivisions of G. An edge in G is called *pendant* if one of its endpoints has degree 1. Given a graph G we denote as G^+ the graph occurring if we subdivide once all the pendant edges of G.

We define the following parameters:

$$\begin{aligned} \textsf{topological-bandwidth}(G) &= \min\{\textsf{bandwidth}(H) \mid H \in \mathbf{SD}(G)\}. \\ \textsf{proper-pathwidth}(G) &= \textsf{linear-width}(G^+). \end{aligned}$$

topological bandwidth has been studied extensively in [60] where it follows that it is polynomially computable for trees. proper-pathwidth is in FPT because it is closed under taking of minors. Moreover, a constructive version of this result follows as there is an easy parameterized reduction of this parameter to linear-width (see [75]). Curiously, there is also a parameterized reduction of pathwidth to proper pathwidth – see [5, 75]. proper pathwidth can be computed in polynomial time for trees using the techniques developed in [74, 32, 72].

In this section we will propose a generic definition of the above parameters different that the one in Section 3. In particular, we will use the graph relations $\leq_{\text{sb}}, \leq_{\text{tp}}, \leq_{\text{mn}}, \leq_{\text{im}}$ defined in Subection 4.1 to generate bandwidth, topological bandwidth, proper pathwidth, and cutwidth.

Let P_r^k be the graph obtained if in a r-vertex path P_r we connect all edges of distance $\leq k$ in P_r. Then, given a graph relation $\leq_*$, we define $\mathbf{par}_{\leq_*}(G) = \min\{k \mid G \leq_* P_r^k \text{ for some value of } r\}$.

Theorem 7. *The following hold*
(1) $\mathbf{par}_{\leq_{\text{sb}}} = \textsf{bandwidth}$,

(2) $\mathbf{par}_{\leq_{\text{tp}}}$ = topological bandwidth,
(3) $\mathbf{par}_{\leq_{\text{mn}}}$ = proper pathwidth

There is also a similar way to define cutwidth if instead of P_r^k we use the immersion relation and a path with all its edges of multiplicity k (we denote this graph as Q_r^k). Also the Emulation in a Path problem is equivalent to the problem asking whether a graph G is a subgraph of the graph R_n^k for some big enough n where $m = 0 \pmod k$. R_n is constructed if we take n/k cliques of size k and add edges between vertices of consecutive cliques. Formally, $R_n^k = (\{1, 2, \ldots, n\}, \{\{i, j\} \mid |\lfloor \frac{i}{k} \rfloor - \lfloor \frac{j}{k} \rfloor| \leq 1\})$.

Notice that both immersion and minor relation are extensions of the topological containment relation ($H \leq_{mn} G$ or $H \leq_{im} G$ implies that $H \leq_{tp} G$), while the topological containment relation is an extension of the subgraph relation. In that sense we can say that the immersion or minor relations represent a more "flexible" relation between graphs than in the case of the subgraph relation that seems to be more "rigid". One may argue that bandwidth incorporates characteristics of the graphs mostly associated with their geometry while proper pathwidth (and the related parameters of pathwidth and linear width) are less "strict" or, in a sense, more topological than geometrical. We feel that this provides some high level evidence for the differences in the parameterized complexity of these parameters. Here what is open is the "intermediate-case" of topological bandwidth. As the topological containment relation is not simple (in the sense this was defined in Subsection 4.1) we would guess that topological bandwidth is hard for some class of the W-hierarchy. This conjecture is supported also by the parameterized intractability of several "geometrically rigid" parameters such as domino-pathwidth and bounded persistence pathwidth (see [30], where persistence is some sort of measure for the rigidity of a path decomposition).

Also the same table prompts us to conjecture that Emulation in a Path remains NP-complete on trees.

parameter	relation	parameterized complexity	complexity on trees
emulation in a path	$\leq_{\text{tp}} R_n^k$	NP-complete	OPEN
bandwidth	$\leq_{\text{sb}} P_n^k$	W[t]-hard for any $t \geq 1$	NP-complete
topological bandwidth	$\leq_{\text{tp}} P_n^k$	OPEN	P
proper pathwidth	$\leq_{\text{mn}} P_n^k$	FPT	P
cutwidth	$\leq_{\text{im}} Q_n^k$	FPT	P

Alas, cutwidth cannot be defined in terms of P_r^k. However, if $\mathbf{par}_{\leq_{\text{im}}}(G) = k$

then $k/2 \leq \mathsf{cutwidth}(G) \leq k(k+1)/2$. Indeed, this holds as the cutwidth of P_r^k is at most $k(k+1)/2$ (for $r \geq 2k$ the equality holds) and because of the following fact (we let the proof as an exercise): If G has cutwidth at most k then there exists a graph H with bandwidth at most $2k$ and where $G \leq_{\text{im}} H$.

The above result implies that there is an FPT-algorithm that either says that $\mathbf{par}_{\leq_{\text{im}}}(G) > k$ or returns that $\mathbf{par}_{\leq_{\text{im}}}(G) \leq c \cdot k^2$ for some constant c. Does this says anything about the parameterized tractability of $\mathbf{par}_{\leq_{\text{im}}}(G)$? Is this contradictory to the assumption that checking whether $\mathbf{par}_{\leq_{\text{im}}}(G) \leq k$ is hard for some of the levels of the parameterized complexity hierarchy?

6.4 Dual-bandwidth

Perhaps the most "unknown" graph layout parameter is dual-bandwidth as there are just a few combinatorial results on it in [15, 31, 49]. We conjecture that its complexity (parameterized or not) follows the behavior of bandwidth. We maintain some hope that the duality of their definitions can help to construct a parameterized reduction of bandwidth to dual bandwidth.

6.5 What about the constants?

For all the graph layout invariants that have been classified so far in FPT the running times were of the form $O(f(k) \cdot n)$. While one can feel happy with the "linearity" of these algorithms, we cannot do the same when we really want to have a "simple" bound for $f(k)$. Usually $f(k)$ is a function of size $2^{O(k^2)}$ or worst and this makes the algorithms (given that one overcomes the technical difficulties in implementing them) inefficient even for reasonable values of k. Consequently, it sounds acceptable to make some "bargain" between the complexity of the polynomial and the exponential part: is there a $O(2^{O(k)}n^a)$ algorithm for some $a \geq 2$ for some of the invariants that we already know that are in FPT?

6.6 Kernels?

Let $\mathcal{L}$ be a parameterized problem, i.e. $\mathcal{L}$ consists of pairs (I, k) where k is the *parameter* of the problem. *Reduction to problem kernel* is the replacement of problem inputs (I, k) by a reduced problem with inputs (I', k') (the kernel) such that $k' = O(k)$, $|I'| = f(k)$ (f is the *size* of the kernel and is a function depending only on k) and $(I, k) \in \mathcal{L} \Leftrightarrow (I', k') \in \mathcal{L}$. (We refer to [26] for discussions on fixed parameter tractability and the ways of constructing kernels.)

Many problems on graphs admit reductions to problem kernels that in most cases are of polynomial size (or even linear size). However, no kernel has ever been found for any linear layout parameter. We would not exclude the existence

of a kernel of exponential size for some graph layout parameter. However, we would conjecture that no linear size kernel exists for the graph layout parameters that we considered so far.

Acknowledgements

The authors are grateful to Fedor V. Fomin for his comments and remarks. Also they wish to thank Hans Bodlaender, as some of the presented results and open problems were inspired during earlier discussions and research with him. Finally, the last author dedicates his work to the memory of Γ.Μ. that passed away while this article was under construction.

References

[1] K. A. Abrahamson, R. G. Downey, and M. R. Fellows. Fixed-parameter tractability and completeness. IV. On completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, 1995.

[2] D. Adolphson and T. C. Hu. Optimal linear ordering. *SIAM Journal on Applied Mathematics*, 25(3):403–423, 1973.

[3] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Mathematics*, 8:277–284, 1993.

[4] S. Bezrukov. Edge isoperimetric problems on graphs (a survey). In *Graph Theory and Combinatorial Biology*, pages 157–197. Janos Bolyai Math. Soc., 1999.

[5] D. Bienstock and P. Seymour. Monotonicity in graph searching. *Journal of Algorithms*, 12(2):239–245, 1991.

[6] H. Bodlaender, M. R. Fellows, and M. T. Hallet. Beyond NP-completeness for problems of bounded width: hardness for the W-hierarchy. In *26th. ACM Symposium on Theory of Computing*, pages 449–458, 1994.

[7] H. Bodlaender, J. Gustedt, and J. A. Telle. Linear-time register allocation for a fixed number of registers. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 1998)*, pages 574–583, New York, 1998. ACM.

[8] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[9] H. L. Bodlaender, M. R. Fellows, and D. M. Thilikos. Starting with nondeterminism: the systematic derivation of linear-time graph layout algorithms. In *Proc. 26th International Symposium on Mathematical Foundations of Computer Science, MFCS 2003*, volume 2747 of *Lectures Notes in Computer Science*, pages 239 – 248. Springer-Verlag, 2003.

[10] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21:358–402, 1996.

[11] H. L. Bodlaender and D. M. Thilikos. Constructive linear time algorithms for branchwidth. In *Automata, languages and programming (Bologna, 1997)*, volume 1256 of *Lecture Notes in Computer Science*, pages 627–637. Springer, Berlin, 1997.

[12] H. L. Bodlaender and D. M. Thilikos. Computing small search numbers in linear time. Technical Report UU-CS-1998-05, Dept. of Computer Science, Utrecht University, 1998.

[13] H. L. Bodlaender and D. M. Thilikos. Computing small search numbers in linear time. In *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway*, pages 37–48, 2004.

[14] H. D. Booth, R. Govindan, M. A. Langston, and S. Ramachandramurthi. Fast algorithms for K_4 immersion testing. *J. Algorithms*, 30(2):344–378, 1999.

[15] T. Calamoneri, A. Massini, and I. Vrt'o. New results on edge-bandwidth. *Theoret. Comput. Sci.*, 307(3):503–513, 2003. Selected papers in honor of Lawrence Harper.

[16] M. Cesati. Compendium of Parameterized Problems. *Department of Computer Science, Systems, and Industrial Engineering, University of Rome "Tor Vergata"*, 22 February 2001.

[17] M. Cesati and H. Wareham. Parameterized complexity analysis in robot motion planning. In *In Proceedings of the 25th IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, Los Alamitos, CA, 1995. IEEE Press.

[18] B. Courcelle, R. G. Downey, and M. R. Fellows. A note on the computability of graph minor obstruction sets for monadic second order ideals. In *Proceedings of the First Japan-New Zealand Workshop on Logic in Computer Science (Auckland, 1997)*, volume 3, pages 1194–1198 (electronic), 1997.

[19] N. D. Dendris, L. M. Kirousis, and D. M. Thilikos. Fugitive-search games on graphs and related parameters. *Theoret. Comput. Sci.*, 172(1-2):233–254, 1997.

[20] J. Díaz, A. Gibbons, M. Paterson, and J. Toran. The minsumcut problem. In R. S. F. Dehen and N. Santoro, editors, *Algorithms and data structures*, volume 519 of *LNCS*, pages 65–79, 1991.

[21] J. Díaz, J. Petit, and M. Serna. A survey on graph layout problems. *ACM Computing Surveys*, 34(3):313–356, 2002.

[22] R. Downey and M. Fellows. Fixed-parameter tractability and completeness. III. Some structural aspects of the *W* hierarchy. In *Complexity theory*, pages 191–225. Cambridge Univ. Press, Cambridge, 1993.

[23] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. In *Proceedings of the Twenty-first Manitoba Conference on Numerical Mathematics and Computing (Winnipeg, MB, 1991)*, volume 87, pages 161–178, 1992.

[24] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. I. Basic results. *SIAM J. Comput.*, 24(4):873–921, 1995.

[25] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for *W*[1]. *Theoretical Computer Science*, 141(1-2):109–131, 1995.

[26] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer-Verlag, New York, 1999.

[27] R. G. Downey and M. R. Fellows. Parameterized complexity after (almost) ten years: review and open questions. In *Combinatorics, computation & logic '99 (Auckland)*, pages 1–33. Springer, 1999.

[28] R. G. Downey, M. R. Fellows, and U. Stege. Computational tractability: the view from Mars. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, (69):73–97, 1999.

[29] R. G. Downey, M. R. Fellows, and U. Taylor. The parameterized complexity of relational database queries and an improved characterization of *W*[1]. In *Combinatorics, complexity, & logic (Auckland, 1996)*, Springer Ser. Discrete Math. Theor. Comput. Sci., pages 194–213. Springer, Singapore, 1997.

[30] R. G. Downey and C. McCartin. Bounded persistence pathwidth. In *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway*, pages 13–24, 2004.

[31] D. Eichhorn, D. Mubayi, K. O'Bryant, and D. B. West. The edge-bandwidth of theta graphs. *J. Graph Theory*, 35(2):89–98, 2000.

[32] J. A. Ellis, I. H. Sudborough, and J. S. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113(1):50–79, 1994.

[33] M. R. Fellows. Parameterized complexity: The main ideas and some research frontiers. In *Algorithms and Computation, 12th International Symposium, ISAAC 2001*, pages 291–307, 2001.

[34] M. R. Fellows and M. A. Langston. Fast self-reduction algorithms for combinatorial problems of VLSI design. In *VLSI algorithms and architectures (Corfu, 1988)*, volume 319 of *Lecture Notes in Comput. Sci.*, pages 278–287. Springer, New York, 1988.

[35] M. R. Fellows and M. A. Langston. Layout permutation problems and well-partially-ordered sets. In *Advanced research in VLSI (Cambridge, MA, 1988)*, pages 315–327. MIT Press, Cambridge, MA, 1988.

[36] M. R. Fellows and M. A. Langston. Nonconstructive tools for proving polynomial-time decidability. *J. Assoc. Comput. Mach.*, 35(3):727–739, 1988.

[37] M. R. Fellows and M. A. Langston. On well-partial-order theory and its application to combinatorial problems of VLSI design. *SIAM Journal on Discrete Mathematics*, 5(1):117–126, 1992.

[38] M. R. Fellows and M. A. Langston. On search, decision, and the efficiency of polynomial-time algorithms. *J. Comput. System Sci.*, 49(3):769–779, 1994.

[39] M. R. Fellows and L. M. R. An analogue of the myhill-nerode theorem and its use in computing finite-basis characterisations (extended abstract). In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science, FOCS 1989*, pages 520–525, 1989.

[40] E. C. Freuder. A sufficient condition for backtrack-free search. *J. Assoc. Comput. Mach.*, 29(1):24–32, 1982.

[41] H. Friedman, N. Robertson, and P. Seymour. The metamathematics of the graph minor theorem. In *Logic and combinatorics (Arcata, Calif., 1985)*, volume 65 of *Contemp. Math.*, pages 229–261. Amer. Math. Soc., Providence, RI, 1987.

[42] M. Garey, D. Johnson, and L.Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[43] M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. W. H. Freeman and Co., San Francisco, Calif., 1979.

[44] F. Gavril. Some NP-complete problems on graphs. In *Proc. 11th Conference on Information Sciences and Systems*, pages 91–95, John Hopkins Univ., Baltimore, 1977.

[45] G. Gottlob, F. Scarcello, and M. Sideri. Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artificial Intelligence*, 138(1-2):55–86, 2002. Knowledge representation and logic programming (El Paso, TX, 1999).

[46] L. Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1(3):385–393, 1966.

[47] L. H. Harper. Optimal assignments of numbers to vertices. *J. Soc. Indust. Appl. Math.*, 12:131–135, 1964.

[48] T. Jiang, D. Mubayi, A. Shastri, and D. B. West. Edge-bandwidth of graphs. *SIAM J. Discrete Math.*, 12(3):307–316 (electronic), 1999.

[49] T. Jiang, D. Mubayi, A. Shastri, and D. B. West. Edge-bandwidth of graphs. *SIAM J. Discrete Math.*, 12(3):307–316 (electronic), 1999.

[50] N. G. Kinnersley. The vertex separation number of a graph equals its path-width. *Information Processing Letters*, 42(6):345–350, 1992.

[51] L. M. Kirousis and C. H. Papadimitriou. Searching and pebbling. *Theoret. Comput. Sci.*, 47(2):205–218, 1986.

[52] L. M. Kirousis and D. M. Thilikos. The linkage of a graph. *SIAM J. Comput.*, 25(3):626–647, 1996.

[53] D. Kuo and G. J. Chang. The profile minimization problem in trees. *SIAM J. Comput.*, 23(1):71–81, 1994.

[54] Y. Lai and K. Williams. A survey of solved problems and applictions on bandwidth, edge-sum and profile of graphs. *Journal of Graph Theory*, 2:75–94, 1999.

[55] T. Lengauer. Black-white pebbles and graph separation. *Acta Informatica*, 16:465–475, 1981.

[56] O. Lichtenstein and A. Pneuli. Chacking that the finite state concurrents programs satisfy their linear specification. In *Proceedings of the 12th ACM Symposioum of Principles of Programming Languages*, pages 97–107, 1997.

[57] D. R. Lick and A. T. White. k-degenerate graphs. *Canad. J. Math.*, 22:1082–1096, 1970.

[58] Lin and Yuan. Profile minimization problem for matrices and graphs. *Acta Mathematicae Applicatae Sinica. English series*, 10(1):107–112, 1994.

[59] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979.

[60] F. S. Makedon, C. H. Papadimitriou, and I. H. Sudborough. Topological bandwidth. *SIAM Journal on Algebraic and Discrete Methods*, 6(3):418–444, 1985.

[61] D. Matula. A min–max theorem for graphs with application to graph coloring. *SIAM Reviews*, 10:481–482, 1968.

[62] D. W. Matula, G. Marble, and J. D. Isaacson. Graph coloring algorithms. In *Graph theory and computing*, pages 109–122. Academic Press, New York, 1972.

[63] B. Mohar and Poljak. Eigenvalues in combinatorial optimization. In *Combinatorial and Graph-Theoretical Problems in Linear Algebra*, volume 50 of *IMA volumes in Mathematics and its Applications*, pages 107–151, 1993.

[64] R. H. Möhring. Graph problems related to gate matrix layout and PLA folding. In *Computational graph theory*, volume 7 of *Comput. Suppl.*, pages 17–51. Springer, Vienna, 1990.

[65] B. Monien and I. H. Sudborough. Min cut is NP-complete for edge weighted trees. *Theoretical Computer Science*, 58(1-3):209–229, 1988.

[66] C. Papadimitriou. The np-completeness of the bandwidth minimization problem. *Computing*, 16:263–270, 1976.

[67] C. H. Papadimitriou and M. Yannakakis. On the complexity of database queries. *J. Comput. System Sci.*, 58(3):407–427, 1999. Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (Tucson, AZ, 1997).

[68] B. B. Plaut. *Theoretical and algorithmic approaches to field-programmable gate array-paritioning*. PhD thesis, The University of Tennessee Nkoxville, 1999.

[69] N. Robertson and P. D. Seymour. Graph minors. I. excluding a forest. *Journal of Combinatorial Theory, series B*, 35:39–61, 1983.

[70] N. Robertson and P. D. Seymour. Graph minors—a survey. In *Surveys in combinatorics 1985 (Glasgow, 1985)*, volume 103 of *London Math. Soc. Lecture Note Ser.*, pages 153–171. Cambridge Univ. Press, Cambridge, 1985.

[71] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

[72] K. Skodinis. Construction of linear tree-layouts which are optimal with respect to vertex separation in linear time. *Journal of Algorithms*, 47(1):40–59, 2003.

[73] G. Szekeres and H. S. Wilf. An inequality for the chromatic number of a graph. *J. Combinatorial Theory*, 4:1–3, 1968.

[74] A. Takahashi, S. Ueno, and Y. Kajitani. Mixed searching and proper-path-width. *Theoretical Computer Science*, 137(2):253–268, 1995.

[75] D. M. Thilikos. Algorithms and obstructions for linear-width and related search parameters. *Discrete Applied Mathematics*, 105:239–271, 2000.

[76] D. M. Thilikos and H. L. Bodlaender. Constructive linear time algorithms for branchwidth. Technical Report UU-CS-2000-38, Dept. of Computer Science, Utrecht University, 2000.

[77] D. M. Thilikos, M. Serna, and H. L. Bodlaender. Cutwidth II: Algorithms for partial w-trees of bounded degree. *Journal of Algorithms*, To appear.

[78] D. M. Thilikos, M. J. Serna, and H. L. Bodlaender. Cutwidth I: A linear time fixed parameter algorithm. *Journal of Algorithms*, To appear.

[79] D. M. Thilikos, M. J. Serna, and H. L. Bodlaender. Constructive linear time algorithms for small cutwidth and carving-width. In D. Lee and S.-H. Teng, editors, *Proc. 11th ISAAC 2000*, volume 1969 of *Lectures Notes in Computer Science*, pages 192–203. Springer-Verlag, 2000.

[80] R. Thomas. Tree-decompositions of graphs. Lecture notes, School of Mathematics. Georgia Institute of Technology, Atlanta, Georgia 30332, USA, 1996.

[81] W. Unger. The complexity of the approximation of the bandwidth problem. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 82, Washington, DC, USA, 1998. IEEE Computer Society.

[82] J. van Leeuwen. *Handbook of theoretical computer science (vol. A): algorithms and complexity*, chapter Graph algorithms, pages 525–631. MIT Press, Cambridge, MA, USA, 1990.