Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism (but were afraid to ask)

Dániel Marx*[1] and Michał Pilipczuk†[2]

1 **Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI), Hungary**
`dmarx@cs.bme.hu`

2 **Department of Informatics, University of Bergen, Norway**
`michal.pilipczuk@ii.uib.no`

Abstract

Given two graphs H and G, the Subgraph Isomorphism problem asks if H is isomorphic to a subgraph of G. While NP-hard in general, algorithms exist for various parameterized versions of the problem. However, the literature contains very little guidance on which combinations of parameters can or cannot be exploited algorithmically. Our goal is to systematically investigate the possible parameterized algorithms that can exist for Subgraph Isomorphism.

We develop a framework involving 10 relevant parameters for each of H and G (such as treewidth, pathwidth, genus, maximum degree, number of vertices, number of components, etc.), and ask if an algorithm with running time $f_1(p_1, p_2, \ldots, p_\ell) \cdot n^{f_2(p_{\ell+1},\ldots,p_k)}$ exists, where each of $p_1, \ldots, p_k$ is one of the 10 parameters depending only on H or G. We show that *all* the questions arising in this framework are answered by a set of 11 maximal positive results (algorithms) and a set of 17 maximal negative results (hardness proofs); some of these results already appear in the literature, while others are new in this paper.

On the algorithmic side, our study reveals for example that an unexpected combination of bounded degree, genus, and feedback vertex set number of G gives rise to a highly nontrivial algorithm for Subgraph Isomorphism. On the hardness side, we present W[1]-hardness proofs under extremely restricted conditions, such as when H is a bounded-degree tree of constant pathwidth and G is a planar graph of bounded pathwidth.

1998 ACM Subject Classification G.2.2 Graph algorithms

Keywords and phrases parameterized complexity, subgraph isomorphism

Digital Object Identifier 10.4230/LIPIcs.STACS.2014.542

1 Introduction

Subgraph Isomorphism is one of the most fundamental graph-theoretic problems: given two graphs H and G, the question is whether H is isomorphic to a subgraph of G. It can be easily seen that finding a k-clique, a k-path, a Hamiltonian cycle, a perfect matching, or a partition of the vertices into triangles are all special cases of Subgraph Isomorphism. Therefore, the

* Research supported by the European Research Council (ERC) grant "PARAMTIGHT: Parameterized complexity and the search for tight complexity results," reference 280152 and OTKA grant NK105645.

† Research supported by the European Research Council (ERC) grant "Rigorous Theory of Preprocessing", reference 267959.

problem is clearly NP-complete in general. There are well-known polynomial-time solvable special cases of the problem, for example, the special case of trees:

▶ **Theorem 1** ([27]). Subgraph Isomorphism *is P-time solvable if G and H are trees.*

Theorem 1 suggests that one should look at cases of Subgraph Isomorphism involving "tree like" graphs. The notion of treewidth measures, in some sense, how close a graph is to being a tree [3]. Treewidth has very important combinatorial and algorithmic applications; in particular, many algorithmic problems become easier on bounded-treewidth graphs. However, Subgraph Isomorphism is NP-hard even if both H and G have treewidth at most 2 [26].

Parameterized algorithms try to cope with NP-hardness by allowing exponential dependence of the running time on certain well-defined parameters of the input, but otherwise the running time depends only polynomially on the input size. We say that a problem is *fixed-parameter tractable* with a parameter k if it can be solved in time $f(k) \cdot n^{O(1)}$ for some computable function f depending only on k [13]. The definition can be easily extended to multiple parameters $k_1, \ldots, k_\ell$. The NP-hardness of Subgraph Isomorphism on graphs of treewidth at most 2 shows that the problem is not fixed-parameter tractable parameterized by treewidth (under standard complexity assumptions). However, there are tractability results that involve other parameters besides treewidth. For example, the following theorem, which follows easily from e.g. Courcelle's Theorem [6], shows the fixed-parameter tractability of Subgraph Isomorphism, jointly parameterized by the size of H and the treewidth of G:

▶ **Theorem 2** (cf. [13]). Subgraph Isomorphism *can be solved in time $f(|V(H)|, \mathbf{tw}(G)) \cdot n$ for some computable function f.*

Some of the results in the literature can be stated as algorithms where certain parameters do appear in the exponent of the running time, but others influence only the multiplicative factor. The classical color-coding algorithm of Alon, Yuster, and Zwick [1] is one such result:

▶ **Theorem 3** ([1]). Subgraph Isomorphism *can be solved in time $2^{O(|V(H)|)} \cdot n^{O(\mathbf{tw}(H))}$.*

One can interpret Theorem 3 as saying that if the treewidth of H is bounded by any fixed constant, then the problem becomes fixed-parameter tractable when parameterized by $|V(H)|$. Notice that treewidth appears in very different ways in Theorems 2 and 3: in the first result, the treewidth of G appears in the multiplicative factor, while in the second result, it is the treewidth of H that is relevant and it appears in the exponent. Yet another algorithm for Subgraph Isomorphism on bounded-treewidth graphs is due to Matoušek and Thomas [26]:

▶ **Theorem 4** ([26]). *For connected H,* Subgraph Isomorphism *can be solved in time $f(\Delta(H)) \cdot n^{O(\mathbf{tw}(G))}$ for some computable function f.*

Again, the dependence on treewidth takes a different form here: now it is the treewidth of G that appears in the exponent. Note that the connectivity condition cannot be omitted: there is an easy reduction from the NP-hard problem Bin Packing with unary sizes to the case of Subgraph Isomorphism where H and G both consist of a set of disjoint paths, i.e., have maximum degree 2 and treewidth 1. Therefore, as Theorem 4 shows, the complexity of the problem depends nontrivially on the number of connected components of the graphs as well.

As the examples above show, even the apparently simple question of how treewidth influences the complexity of Subgraph Isomorphism does not have a clear-cut answer: the treewidth of H and G influences the complexity in different ways, they can appear in the running time either as an exponent or as a multiplier, and the influence of treewidth can be interpreted only in combination with other parameters (such as the number of vertices or maximum degree of H). The situation becomes even more complex if we consider further parameters of the graphs as well. Cliquewidth, introduced by Courcelle and Olariu [8], is a graph measure that can be always bounded by a function of treewidth, but treewidth can be arbitrary large even for graphs of bounded cliquewidth (e.g., for cliques). Therefore, algorithms for graphs of bounded cliquewidth are strictly more general than those for graphs of bounded treewidth. By the results of Courcelle et al. [7], Theorem 2 can be generalized by replacing treewidth with cliquewidth. However, no such generalization is possible for Theorem 3: cliques have cliquewidth 2, thus replacing treewidth with cliquewidth in Theorem 3 would imply that Clique (parameterized by the size of the clique to be found) is fixed-parameter tractable, contrary to widely accepted complexity assumptions. In the case of Theorem 4, it is not at all clear if treewidth can be replaced by cliquewidth: we are not aware of any result in the literature on whether Subgraph Isomorphism is fixed-parameter tractable parameterized by the maximum degree of H if G is a connected graph whose cliquewidth is bounded by a fixed constant.

Theorem 2 can be generalized into a different direction using the concept of *bounded local treewidth.* Model checking with a fixed first-order formula is known to be linear-time solvable on graphs of bounded local treewidth [15], which implies that Subgraph Isomorphism can be solved in time $f(|V(H)|)\cdot n$ if G is planar, or more generally, in time $f(|V(H)|, \mathbf{genus}(G))\cdot n$ for arbitrary G. Having an algorithm for bounded-genus graphs, one can try to further generalize the results to graphs excluding a fixed minor or to graphs not containing the subdivision of a fixed graph (that is, to graphs not containing a fixed graph as a topological minor). Such a generalization is possible: a result of Dvořak et al. [10] states that model checking with a fixed first-order formula is linear-time solvable on graphs of bounded expansion, and it follows that Subgraph Isomorphism can be solved in time $f(|V(H)|, \mathbf{hadw}(G)) \cdot n$ or $f(|V(H)|, \mathbf{hadw}_{\mathrm{T}}(G)) \cdot n$, where $\mathbf{hadw}(G)$ (resp., $\mathbf{hadw}_{\mathrm{T}}(G)$) is the maximum size of a clique that is a minor (resp., topological minor) of G. These generalizations of Theorem 2 show that planarity, and more generally, topological restrictions on G can be helpful in solving Subgraph Isomorphism, and therefore the study of parameterizations of Subgraph Isomorphism should include these parameters as well.

Our goal is to perform a systematic study of the influence of the parameters: for all possible combination of parameters in the exponent and in the multiplicative factor, we would like to determine if there is an algorithm whose running time is of this form. The main thesis of the paper is the following: (1) as the influence of the parameters on the complexity is highly nontrivial and subtle, even small changes in the choice of parameters can have substantial and counterintuitive consequences, and (2) the current literature gives very little guidance on whether an algorithm with a particular combination of parameters exist.

2 Our framework

We present a framework in which the questions raised above can be systematically treated and completely answer every question arising in the framework. Our setting is the following. First, we define the following 10 graph parameters (we give a brief justification for each parameter why it is relevant for the study of Subgraph Isomorphism):

- *Number of vertices* $|V(\cdot)|$. As Theorems 2 and 3 show, $|V(H)|$ is a highly relevant parameter for the problem. Note, however, that the problem becomes trivial if $|V(G)|$ can appear in the multiplier or in the exponent, or if $|V(H)|$ can appear in the exponent.
- *Number of connected components* $\mathbf{cc}(\cdot)$. As Theorem 4 and the reduction from Bin Packing show, it makes a difference if we restrict the problem to connected graphs (or, more generally, if we allow the running time to depend on the number of components).

- *Maximum degree* $\Delta(\cdot)$. The maximum degree of H plays an important role in Theorem 4, thus exploring the effect of this parameter is clearly motivated. In general, many parameterized problems become easier on bounded-degree graphs, mainly because then the distance-d neighborhood of each vertex has bounded size for bounded d.
- *Treewidth* $\mathbf{tw}(\cdot)$. Theorems 2–4 give classical algorithms where treewidth appears in different ways; understanding how exactly treewidth can influence complexity is one of the most important concrete goals of the paper.
- *Pathwidth* $\mathbf{pw}(\cdot)$. As pathwidth is always at least treewidth, but can be strictly larger, algorithms parameterized by pathwidth can exist even if no algorithms parameterized by treewidth are possible. Given the importance of treewidth, it is natural to explore the possibility of algorithms in the more restricted setting of bounded-pathwidth graphs.
- *Feedback vertex set number* $\mathbf{fvs}(\cdot)$. A feedback vertex set is a set of vertices whose deletion makes the graph a forest; the feedback vertex set number is the size of the smallest such set. Similarly to graphs of bounded pathwidth, graphs of bounded feedback vertex set number form a subclass of bounded-treewidth graphs, hence it is natural to explore what algorithms we can obtain with this parameterization. Note that Graph Isomorphism (not subgraph!) is fixed-parameter tractable parameterized by feedback vertex set number [19], while only $n^{O(\mathbf{tw}(G))}$ time algorithms are known parameterized by treewidth [2, 29]. This shows that $\mathbf{fvs}(\cdot)$ can be a useful parameter for problems involving isomorphisms.
- *Cliquewidth* $\mathbf{cw}(\cdot)$. As cliquewidth is bounded by a function of treewidth, parameterization by cliquewidth leads to more general algorithms than parameterization by treewidth. However, treewidth can be replaced by cliquewidth in Theorem 2, but not in Theorem 3. Therefore, understanding the role of cliquewidth is a nontrivial and interesting challenge.
- *Genus* $\mathbf{genus}(\cdot)$. Understanding the complexity of Subgraph Isomorphism on planar graphs (and more generally, on bounded-genus graphs) is a natural goal, especially in light of the positive results that arise from the generalizations of Theorem 2.
- *Hadwiger number* $\mathbf{hadw}(\cdot)$. That is, the size of the largest clique that is the minor of the graph. A graph containing a K_k-minor needs to have genus $\Omega(k^2)$; therefore, algorithms for graphs excluding a fixed clique as a minor generalize algorithms for bounded-genus graphs. In many cases, such a generalization is possible, thanks to structure theorems and algorithmic advances for H-minor free graphs [9, 17, 30].
- *Topological Hadwiger number* $\mathbf{hadw}_{\mathrm{T}}(\cdot)$. That is, the size of the largest clique whose subdivision is a subgraph of the graph. A graph containing the subdivision of a K_k contains K_k as a minor. Therefore, algorithms for graphs excluding a fixed topological clique minor generalize algorithms for graphs excluding a fixed clique minor. Recent work show that some algorithmic results for graphs excluding a fixed minor can be generalized to excluded topological minors [14, 16, 18]. In particular, the structure theorem of Grohe and Marx [18] states, in a precise technical sense, that graphs excluding a fixed topological minor are composed from parts that are either "almost bounded-degree" or exclude a fixed minor. Therefore, it is interesting to investigate in our setting how this parameter interacts with the parameters smallest excluded clique minor and maximum degree.

Given this list of 10 parameters, we would like to understand if an algorithm with running time of the form $f_1(p_1, p_2, \ldots, p_\ell) \cdot n^{f_2(p_{\ell+1}, \ldots, p_k)}$ exists, where each p_i is one of these 10 parameters applied on either H and G, and f_1, f_2 are arbitrary computable functions of these parameters. We call such a sequence of parameters a *description,* and we say that an algorithm is *compatible* with the description if its running time is of this form. Observe that Theorems 2 and 3 can be stated as the existence of algorithms compatible with particular descriptions. However, Theorem 4 has the extra condition that H is connected (or in other words, the number of connected components of H is 1) and therefore it does not seem to fit into this framework. In order to include such statements into our investigations, we extend the definition of descriptions with some number of *constraints* that restrict the value of certain parameters to particular constants. Specifically, we consider the following 5 constraints on H and G, each of which corresponds to a particularly motivated special case of the problem:

- *Genus is 0.* That is, the graph is planar. Any positive result on planar graphs is clearly of interest, even if it does not generalize to arbitrary fixed genus. Conversely, whenever possible, we would like to state hardness results for planar graphs, rather than for bounded-genus with an unspecified bound on the genus.
- *Number of components is 1.* Any positive result under this restriction is quite motivated, and as the examples above show, the problem can become simpler on connected graphs.
- *Treewidth is at most 1.* That is, the graph is a forest. Trees can behave very differently than bounded-treewidth graphs (compare Theorem 1 with the fact the the problem is NP-hard on graphs of treewidth 2), thus investigating the special case of forests might turn up additional algorithmic results.
- *Maximum degree is at most 2.* That is, the graph consists of disjoint paths and cycles. Clearly, this class is very restricted, but as the NP-hardness of Hamiltonian Cycle shows, this property of H does not guarantee tractability without further assumptions.
- *Maximum degree is at most 3.* To provide contrast with the case of maximum degree at most 2, we would like to state negative results for graphs of maximum degree at most 3.

We restrict our attention to these 5 specific constraints. For example, we do not specifically investigate possible algorithms that work on, say, graphs of feedback vertex set size 1 or of pathwidth 2: we can argue that such algorithms are interesting only if they can be generalized to every fixed bound on the feedback vertex set size or on pathwidth (whereas an algorithm for planar graphs is interesting even if it does not generalize to higher genera).

3 Results

Our formulation of the general framework includes an enormous number of concrete research questions. Even without considering the 5 specific constraints, we have 19 parameters (10 for H and 9 for G) and each parameter can be either in the exponent of the running time, in the multiplier of the running time, or does not appear at all in the running time. Therefore, there are at least $3^{19} \approx 10^9$ descriptions and corresponding complexity questions in this framework. The present paper answers all these questions (under standard complexity assumptions).

In order to reduce the number of questions we observe that there are some clear implications between them. Clearly, the $f_1(|V(H)|) \cdot n^{f_2(\mathbf{tw}(H))}$ time algorithm of Theorem 3 implies the existence of, say, an $f_1(|V(H)|, \mathbf{genus}(G)) \cdot n^{f_2(\mathbf{pw}(H), \Delta(G))}$ time algorithm: $\mathbf{pw}(H)$ is always at least $\mathbf{tw}(H)$ and the fact that the latter running time can depend on $\mathbf{genus}(G)$ and $\Delta(G)$ can be ignored. The main claim of the paper is that every question arising in the framework can be answered by a set of 11 positive and 17 negative results:

> The positive and negative results presented in Table 1 imply a positive or negative answer to every question arising in this framework. (*)

That is, either there is a positive result for a more restrictive description, or a negative result for a less restrictive restriction. The following two examples show how one can deduce the answer to specific questions from Table 1.

Short Description	Theorem	*H*										*G*								
		$\lvert V(\cdot)\rvert$	cc	Δ	fvs	pw	tw	cw	genus	hadw	hadw_T	cc	Δ	fvs	pw	tw	cw	genus	hadw	hadw_T
FO model checking	Theorem P.1	M															M			
	Theorem P.2	M																		M
Color coding	Theorem P.3	M					E													
Matoušek-Thomas	Theorem P.4		M	M												E				
Paths&Cycles → Paths&Cycles	Theorem P.5											E	2							
Dynamic Programming	Theorem P.6		E	2												M				
	Theorem P.7		E	2													E			
	Theorem P.8*		M													1				
FVS and CSPs	Theorem P.9*		M	2									M	M						
	Theorem P.10*		E										M	M				E		
	Theorem P.11*		E	E									M	M					E	
Bin Packing	Theorem N.1											M	2			1				
	Theorem N.2		1				1							E	E			0		
	Theorem N.3			2								1	3		E	1				
Planar cubic HamPath	Theorem N.4		1	2			1						3					0		
Clique	Theorem N.5	M	1					E												
HamPath in bounded **cw**	Theorem N.6		1	2			1										M			
Grid Tiling, 1-in-n gadgets	Theorem N.7*		M			E	1					1	3	M	M			0		
	Theorem N.8*		1			E	1						M	M	M			M	E	
	Theorem N.9*		1			E	1						3	M	M			M		
	Theorem N.10*		1	3		E	1						M	M	M		E	M		
Grid Tiling, moustache gadgets	Theorem N.11*		1	3		E	1							M	M			0		
	Theorem N.12*		1			E	1						3		M			0		
Small planar graph	Theorem N.13*	M	1	3					0											
Exact Planar Arc Supply	Theorem N.14*		M	2			1					1		M	M			0		
	Theorem N.15*		M	2			1					1	3		M			0		
	Theorem N.16*		M	2			1					1		M	M		E	M		
	Theorem N.17*		M	2			1					1	M		M		E	M		

Figure 1 Positive (blue) and negative (rosa) results in the paper; the numbering refers to the full version. Asterisks denote new findings not known before. Entry **M** denotes that the parameter appears in the multiplier, entry **E** denotes that the parameter appears in the exponent, while explicit integer constants denote constraining the parameter to be bounded by the respective constant value.

▶ **Example 5.** Is there an algorithm for Subgraph Isomorphism with running time $n^{f(\mathbf{fvs}(G))}$ when G is a planar graph of maximum degree 3 and H is connected? Looking at Table 1, the line of Theorem P.10 shows the existence of an algorithm with running time $f_1(\mathbf{fvs}(G), \Delta(G)) \cdot n^{f_2(\mathbf{genus}(G), \mathbf{cc}(H))}$. When restricted to the case when G is a planar graph (i.e., $\mathbf{genus}(G) = 0$) with $\Delta(G) \leq 3$ and H is connected (i.e., $\mathbf{cc}(H) = 1$), then running time of this algorithm can be expressed as $f(\mathbf{fvs}(G)) \cdot n^{O(1)}$. This is in fact better than the running time $n^{f(\mathbf{fvs}(G))}$ we asked for, hence the answer is positive.

▶ **Example 6.** Is there an algorithm for Subgraph Isomorphism with running time $f(\mathbf{tw}(G)) \cdot n^{g(\Delta(G))}$ when G is a connected planar graph? Looking at Table 1, the line of Theorem N.7 gives a negative result for algorithms with running time $f_1(\mathbf{cc}(H), \mathbf{pw}(G), \mathbf{fvs}(G)) \cdot n^{f_2(\mathbf{pw}(H))}$ when restricted to instances where H is a forest and G is a connected planar graph of maximum degree 3. Note that $\mathbf{tw}(G) \leq \mathbf{pw}(G)$, so an $f(\mathbf{tw}(G)) \cdot n^{g(\Delta(G))}$ time algorithm for connected planar graphs would give an $f(\mathbf{pw}(G)) \cdot n^{O(1)}$ time algorithm for connected planar graphs of maximum degree 3, which is a better running time then the one ruled out by Theorem N.7. Therefore, the answer is negative.

To make claim (*) formal and verifiable, we define an ordering relation between descriptions in a way that guarantees that if description D_1 is *stronger* than D_2, then an algorithm compatible with D_1 implies the existence of an algorithm compatible with D_2. Roughly speaking, the definition of this ordering takes into account three immediate implications:

- Removing a parameter makes the description stronger.
- Moving a parameter from the exponent to the multiplier makes the description stronger.
- We consider a list of combinatorial relations between the parameters and their implications on the descriptions: for example, $\mathbf{tw}(H) \leq \mathbf{pw}(H)$ implies that replacing $\mathbf{pw}(H)$ with $\mathbf{tw}(H)$ makes the description stronger. Our list of relations include some more complicated and less obvious connections, such as $\mathbf{tw}(H)$ can be bounded by a function of $\mathbf{cw}(H)$ and $\Delta(H)$, thus replacing $\mathbf{cw}(H)$ and $\Delta(H)$ with $\mathbf{tw}(H)$ makes the description stronger.

The precise definition of the ordering of the descriptions appears in the full version of the paper. Given the ordering, we need to show the positive results only for the maximally strong descriptions and the negative results for the minimally strong descriptions. Our main result is that every question arising in the framework can be explained by a set of 11 maximally strong positive results and a set of 17 minimally strong negative results listed in Table 1.

▶ **Theorem 7.** *For every description D, either (a) Table 1 contains a positive result for a description D' such that D' is stronger than D, or (b) Table 1 contains a negative result for a description D' such that D is stronger than D'.*

At this point, the reader might wonder how it is possible to prove Theorem 7, that is to verify that the positive and negative results on Table 1 indeed cover every possible description. Interestingly, formulating the task of checking whether a set of positive and negative results on an unbounded set of parameters explains every possible description leads to an NP-hard problem (we omit the details). Therefore, we have implemented a simple backtracking algorithm that checks if every description is explained by the set of positive and negative results given in the input. We did not make a particular effort to optimize the program, as it was sufficiently fast for our purposes on contemporary desktop computers. The program indeed verifies that our set of positive and negative results is complete. We have used this program extensively during our research to find descriptions that are not yet explained by our current set of results. By focusing on one concrete unexplained description, we could always either find a corresponding algorithm or prove a hardness result, which we could add to our set of results. By iterating this process, we have eventually arrived at a set of results that is complete. The program and the data files are available as electronic supplementary material of the arxiv version of the present paper [25].

As the systematic study of our framework involves proving dozens of results that require combination of many different tools, in this extended abstract we only survey our framework and state the results, giving a short glimpse into the most important findings and techniques used for proving them. For a full discussion of the results, including all the proofs, we refer to the full version of the paper that can be found on arxiv [25].

4 Algorithms

Let us highlight some of the new algorithmic results discovered by the exhaustive analysis of our framework. While the negative results suggest that the treewidth of G appearing in the multiplicative factor of the running time helps very little if the size of H can be large, we show that the more relaxed parameter feedback vertex set is useful on bounded-degree planar graphs. Specifically, we prove the following result:

▶ **Theorem 8.** Subgraph Isomorphism *can be solved in time* $f(\Delta(G), \mathbf{fvs}(G)) \cdot n^{O(1)}$ *if* H *is connected and* G *is planar.*

The proof of Theorem 8 turns the Subgraph Isomorphism problem into a Constraint Satisfaction Problem (CSP) whose primal graph is planar. We observe that this CSP instance has a special variable v that we call a *projection sink:* roughly speaking, it has the property that v can be reached from every other variable via a sequence of constraints that are projections. We prove the somewhat unexpected result that a planar CSP instance having a projection sink is polynomial-time solvable, which allows us to solve the Subgraph Isomorphism instance within the claimed time bound. This new property of having a projection sink and the corresponding polynomial-time algorithm for CSPs with this property can be interesting on its own and possibly useful in other contexts.

We generalize the result from planar graphs to bounded-genus graphs and to graphs excluding a fixed minor in the following way:

▶ **Theorem 9.** Subgraph Isomorphism *can be solved in time*
1. $f_1(\Delta(G), \mathbf{fvs}(G)) \cdot n^{f_2(\mathbf{genus}(G), \mathbf{cc}(H))}$, *and*
2. $f_1(\Delta(G), \mathbf{fvs}(G)) \cdot n^{f_2(\mathbf{hadw}(G), \Delta(H), \mathbf{cc}(H))}$.

For (1), we need only well-known diameter-treewidth relations for bounded-genus graphs [12], but (2) needs a nontrivial application of structure theorems for graphs excluding a fixed minor and handling vortices in almost-embeddable graphs. Note that these two results are incomparable: in (2), the exponent contains $\Delta(H)$ as well, thus it does not generalize (1). Intuitively, the reason for this is that when lifting the algorithm from the bounded-genus case to the minor-free case, high-degree apices turn out to be problematic. On the other hand, Theorem N.8 shows that incorporating other parameters is (probably) unavoidable when moving to the more general minor-free setting. We find it interesting that our study revealed that the bounded-genus case and the minor-free case are provably different when the parameterized complexity of Subgraph Isomorphism is concerned.

The reader might find it unmotivated to present algorithms that depend on so many parameters in strange ways, but let us emphasize that these results are maximally strong results in our framework. That is, no weakening of the description can lead to an algorithm (under standard complexity assumptions): for example, $\mathbf{genus}(H)$ or $\mathbf{cc}(H)$ cannot be moved from the exponent to the multiplier, or $\Delta(H)$ cannot be omitted from the exponent in (2). Therefore, these result show, in a well-defined sense, the limits of what can be achieved. Finding such maximal results is precisely the goal of developing and analyzing our framework: it seems unlikely that one would come up with results of the form of Theorem 9 without an exhaustive investigation of all the possible combinations of parameters.

On the other hand, we generalize Theorem 1 from trees to forests, parameterized by the number of connected components of H. This seemingly easy task turns out to be surprisingly challenging. The dynamic programming algorithm of Theorem 1 relies on a step that involves computing maximum matching in a bipartite graph. The complications arising from the existence of multiple components of H makes this matching step more constrained and significantly harder. In fact, the only way we were able to solve these matching problems is by the randomized algebraic matching algorithm of Mulmuley et al. [28]. Therefore, our result is a randomized algorithm for this problem:

▶ **Theorem 10.** Subgraph Isomorphism *can be solved in randomized time* $f(\mathbf{cc}(H)) \cdot n^{O(1)}$ *with false negatives, if* H *and* G *are forests.*

Again, we find it a success of our framework that it directed attention to this particularly interesting special case of the problem. Obtaining a deterministic algorithm for this variant is an interesting open problem.

5 Hardness proofs

Two different technologies are needed for proving negative results about algorithms satisfying certain descriptions: NP-hardness and W[1]-hardness. Recall that a W[1]-hard problem is unlikely to be fixed-parameter tractable and one can show that a problem is W[1]-hard by presenting a parameterized reduction from a known W[1]-hard problem (such as Clique) to it. The most important property of a parameterized reduction is that the parameter value of the constructed instance can be bounded by a function of the parameter of the source instance; see [13] for more details.

- To give evidence that no $n^{f(p_1,\dots,p_k)}$ time algorithm for Subgraph Isomorphism exists, one would like to show that Subgraph Isomorphism remains NP-hard on instances where the value of the parameters $p_1, \dots, p_k$ are bounded by some universal constant.
- To give evidence that no $f_1(p_1, p_2, \dots, p_\ell) \cdot n^{f_2(p_{\ell+1},\dots,p_k)}$ time algorithm for Subgraph Isomorphism exists, one would like to show that Subgraph Isomorphism is W[1]-hard parameterized by $p_1, \dots, p_\ell$ on instances where the values of $p_{\ell+1}, \dots, p_k$ are bounded by some universal constant. That is, what is needed is a parameterized reduction from a known W[1]-hard problem to Subgraph Isomorphism in such a way that parameters $p_1, \dots, p_\ell$ of the constructed instance are bounded by a function of the parameters of the source instance, while the values of $p_{\ell+1}, \dots, p_k$ are bounded by some universal constant.

Additionally, the reductions need to take into account the extra constraints (planarity, treewidth 1, etc.) appearing in the description. The nontrivial results of this paper are of the second type: we prove the W[1]-hardness of Subgraph Isomorphism with certain parameters, under the assumption that certain other parameters are bounded by a universal constant. Intuitively, a substantial difference between NP-hardness proofs and W[1]-hardness proofs is that in a typical NP-hardness proof from, say, 3-SAT, one replaces each variable and clause with a small gadget having a *constant* number of states, whereas in a typical W[1]-hardness proof from, say, Clique, one creates a bounded number of large gadgets having an *unbounded* number of states, e.g., the states correspond to the vertices of the original graph. Therefore, usually the first goal in W[1]-hardness proofs is to construct gadgets that are able to express a large number of states.

Most of our W[1]-hardness results are for planar graphs or for graphs close to planar. As many parameterized problems become fixed-parameter tractable on planar graphs, there is only a handful of planar W[1]-hardness proofs in the literature [4, 5, 11, 24]. These hardness proofs need to construct gadgets that are both planar and able to express a large number of states, which can be a challenging task. A canonical problem that can serve as a useful starting point for W[1]-hardness proofs on planar graphs is Grid Tiling [23, 24]. Most of our W[1]-hardness proofs indeed use Grid Tiling as the source problem. In some cases we use a new problem, Exact Planar Arc Supply, which we prove to be W[1]-hard and which is inspired by the problem Planar Arc Supply introduced by Bodlaender et al. [4].

Besides planarity (or near-planarity), our hardness proofs need to overcome other challenges as well: we bound combinations of maximum degree (of H or G), pathwidth, cliquewidth etc. The following theorem demonstrates the type of restricted results we are able to get. Note that the more parameters appear in the running time and the more restrictions H and G have, the stronger the hardness result is.

▶ **Theorem 11.** *Assuming $FPT \neq W[1]$, there is no algorithm for* Subgraph Isomorphism *with running time*

- $f_1(\mathbf{pw}(G)) \cdot n^{f_2(\mathbf{pw}(H))}$, *even if both H and G are connected planar graphs of maximum degree 3 and H is a tree, or*
- $f_1(\Delta(G), \mathbf{pw}(G), \mathbf{fvs}(G), \mathbf{genus}(G)) \cdot n^{f_2(\mathbf{pw}(H), \mathbf{cw}(G))}$, *even if both H and G are connected and H is a tree of maximum degree 3.*

6 Conclusions

In this paper we have developed a framework for studying different parameterizations of Subgraph Isomorphism and completely answered every question arising in this framework. Systematic studies of parameterizations have been performed before for various problems [20, 21, 22, 31], but never on such a massive scale as in the present paper. We have demonstrated that even if the number of questions is on the order of billions, finding the maximal set of positive results and the maximal set of negative results that explain every specific question of the framework is a doable project and might involve only a few dozen concrete results. At such a large scale, even verifying that a set of results explains every possible question is a daunting task. We have resorted to the help of a computer program that checks this efficiently; the program can be helpful for similar investigations in the future.

While developing the framework and showing that it can be completely explained by a small set of results is the conceptually most novel part of the paper, we would like to emphasize that some of the concrete positive and negative results are highly nontrivial and technically novel. On the algorithmic side, we have discovered a simple, but unexpectedly challenging case: packing a forest H into a forest G, parameterized by the number of connected components of H. We presented a nontrivial randomized dynamic programming algorithm for this problem using algebraic matching algorithms. Our investigations turned up an unexpected combination of parameters that results in tractable cases: maximum degree, feedback vertex set number, and genus of G. In a somewhat surprising manner, tractability relies on the fact that a certain property, the existence of a projection sink, allows us to dramatically reduce treewidth in bounded-genus CSP instances. This new result on CSPs can be of independent interest. We have generalized the result to graphs excluding a fixed minor (with a slightly different parameterization). The generalization is not just a straightforward application of known structure theorems: we had to use a fairly complicated dynamic programming scheme on tree decompositions to exploit the existence of a projection sink and we had to handle almost embeddable graphs including all the gory details of vortices.

On the hardness side, many of our W[1]-hardness proofs involve planar (or bounded-genus) graphs. W[1]-hardness proofs are typically involved, as they require complicated gadget constructions. Reducing from the Grid Tiling problem helps streamlining the reductions, but the actual gadgets have to be constructed in a problem-specific way. In our case the construction of gadgets is particularly challenging since we have to satisfy extreme restrictions.

It might not be apparent from the paper, but the authors did exercise some restraint when defining the framework. Only those graph parameters were included in the framework that already had some interesting nontrivial connection to the Subgraph Isomorphism problem. One could extend the framework with further parameters, such as chromatic number, girth, or (edge) connectivity, but it is not clear whether these parameters would influence the complexity of the problem in an interesting way and whether these parameters would add anything to the message of the results besides further complications. Moreover, recall that for similar reasons we have constrained ourselves to 5 particularly interesting constraints corresponding to small fixed values of certain parameters.

The reader might wonder: do the authors advocate this kind of massive investigation for each and every problem? It seems that the Subgraph Isomorphism problem is particularly suited for such treatment. First, previous results suggest that a wide range of parameters influence the complexity of the problem in nontrivial ways. Second, the Subgraph Isomorphism problem involves two graphs H and G and the same parameter for H or G can play very different role. This effectively doubles the number of parameters that need to be considered. Therefore, the problem has a very complicated ecology of parameters that can be understood only with a large-scale formal investigation. For other problems, say, Vertex Coloring, the complexity landscape is expected to be much simpler, and probably fewer new results (if any) need to be invented to explain every combination of parameters. Therefore, we suggest exploring problems using a detailed framework similar to ours only if there is evidence for complex interaction of parameters. Variants of Subgraph Isomorphism might be natural candidates for such investigations: for example, (i) the homomorphism problem for graphs, (ii) colored versions of Subgraph Isomorphism, (iii) extension versions of Subgraph Isomorphism (where we have to extend a partial subgraph isomorphism given in the input), or (iv) the counting version of Subgraph Isomorphism (this problem was suggested by Petteri Kaski).

References

1 N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

2 H. L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *J. Algorithms*, 11(4):631–643, 1990.

3 H. L. Bodlaender and A. M. C. A. Koster. Combinatorial Optimization on Graphs of Bounded Treewidth. *The Computer Journal*, 51(3):255–269, 2008.

4 H. L. Bodlaender, D. Lokshtanov, and E. Penninkx. Planar Capacitated Dominating Set is W[1]-hard. In *IWPEC*, pages 50–60, 2009.

5 L. Cai, M. R. Fellows, D. W. Juedes, and F. A. Rosamond. The complexity of polynomial-time approximation. *Theory Comput. Syst.*, 41(3):459–477, 2007.

6 B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

7 B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.

8 B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

9 E. D. Demaine, M. Hajiaghayi, and K. Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *FOCS*, pages 637–646, 2005.

10 Z. Dvořak, D. Král, and R. Thomas. Deciding first-order properties for sparse graphs. In *FOCS*, pages 133–142, 2010.

11 R. Enciso, M. R. Fellows, J. Guo, I. A. Kanj, F. A. Rosamond, and O. Suchý. What makes equitable connected partition easy. In *IWPEC*, pages 122–133, 2009.

12 D. Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27:275–291, 2000.

13 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1 edition, March 2006.

14 F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Linear kernels for (connected) dominating set on graphs with excluded topological subgraphs. In *STACS*, pages 92–103, 2013.

15 M. Frick and M. Grohe. Deciding first-order properties of locally tree-decomposable structures. *J. ACM*, 48(6):1184–1206, 2001.

16 M. Grohe, K. Kawarabayashi, D. Marx, and P. Wollan. Finding topological subgraphs is fixed-parameter tractable. In *STOC*, pages 479–488, 2011.

17 M. Grohe, K. Kawarabayashi, and B. A. Reed. A simple algorithm for the graph minor decomposition - logic meets structural graph theory. In *SODA*, pages 414–431, 2013.

18 M. Grohe and D. Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. In *STOC*, pages 173–192, 2012.

19 S. Kratsch and P. Schweitzer. Isomorphism for graphs of bounded feedback vertex set number. In *SWAT*, pages 81–92, 2010.

20 M. Kronegger, A. Pfandler, and R. Pichler. Parameterized complexity of optimal planning: A detailed map. In *IJCAI*, 2013.

21 M. Lackner and A. Pfandler. Fixed-parameter algorithms for closed world reasoning. In *ECAI*, pages 492–497, 2012.

22 M. Lackner and A. Pfandler. Fixed-parameter algorithms for finding minimal models. In *KR*, 2012.

23 D. Marx. On the optimality of planar and geometric approximation schemes. In *FOCS*, pages 338–348, 2007.

24 D. Marx. A tight lower bound for planar multiway cut with fixed number of terminals. In *ICALP (1)*, pages 677–688, 2012.

25 D. Marx and M. Pilipczuk. Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism (but were afraid to ask). *CoRR*, abs/1307.2187, 2013.

26 J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1-3):343–364, 1992.

27 D. W. Matula. Subtree isomorphism in $O(n^{5/2})$. In *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 91–106. Elsevier, 1978.

28 K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

29 I. N. Ponomarenko. The isomorphism problem for classes of graphs that are invariant with respect to contraction. *Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI)*, 174(Teor. Slozhn. Vychisl. 3):147–177, 182, 1988. In Russian.

30 N. Robertson and P.D. Seymour. Graph minors XVI. Excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 77:1–27, 1999.

31 M. Samer and S. Szeider. Constraint satisfaction with bounded treewidth revisited. *J. Comput. Syst. Sci.*, 76(2):103–114, 2010.