Algorithmic Meta-theorems for Restrictions of Treewidth

Michael Lampis

Computer Science Department,
Graduate Center, City University of New York
mlampis@gc.cuny.edu

Abstract. Possibly the most famous algorithmic meta-theorem is Courcelle's theorem, which states that all MSO-expressible graph properties are decidable in linear time for graphs of bounded treewidth. Unfortunately, the running time's dependence on the formula describing the problem is in general a tower of exponentials of unbounded height, and there exist lower bounds proving that this cannot be improved even if we restrict ourselves to deciding FO logic on trees.
We investigate whether this parameter dependence can be improved by focusing on two proper subclasses of the class of bounded treewidth graphs: graphs of bounded vertex cover and graphs of bounded max-leaf number. We prove stronger algorithmic meta-theorems for these more restricted classes of graphs. More specifically, we show it is possible to decide any FO property in both of these classes with a singly exponential parameter dependence and that it is possible to decide MSO logic on graphs of bounded vertex cover with a doubly exponential parameter dependence. We also prove lower bound results which show that our upper bounds cannot be improved significantly, under widely believed complexity assumptions. Our work addresses an open problem posed by Michael Fellows.

1 Introduction

Algorithmic metatheorems are general statements of the form "*All problems sharing property P, restricted to a class of inputs I can be solved efficiently*". The archetypal, and possibly most celebrated, such metatheorem is Courcelle's theorem which states that every graph property expressible in monadic second-order (MSO_2) logic is decidable in linear time if restricted to graphs of bounded treewidth [3]. Metatheorems have been a subject of intensive research in the last years producing a wealth of interesting results. Some representative examples of metatheorems with a flavor similar to Courcelle's can be found in the work of Frick and Grohe [12], where it is shown that all properties expressible in first order (FO) logic are solvable in linear time on planar graphs, and the work of Dawar et al. [5], where it is shown that all FO-definable optimisation problems admit a PTAS on graphs excluding a fixed minor (see [14] and [15] for more results on the topic).

Many interesting extensions have followed Courcelle's seminal result: for instance, Courcelle's theorem has been extended to logics more suitable for the expression of optimisation problems [1]. It has also been investigated whether it's possible to obtain similar results for larger graph classes (see [4] for a metatheorem for bounded cliquewidth graphs, [11] for corresponding hardness results and [17] for hardness results for graphs of small but unbounded treewidth). Finally, lower bound results have been shown proving that the running times predicted by Courcelle's theorem can not be improved significantly in general [13].

This lower bound result is one of the main motivations of this work, because in some ways it is quite devastating. Though Courcelle's theorem shows that a vast class of problems is solvable in linear time on graphs of bounded treewidth, the "hidden constant" in this running time, that is, the running time's dependence on the input's other parameters, which are the graph's treewidth and the formula describing the problem, is in fact (in the worst case) a tower of exponentials. Unfortunately, in [13] it is shown that this tower of exponentials is unavoidable even if we restrict ourselves to deciding FO logic on trees.

In this paper our aim is to investigate if it is possible to go aroung this harsh lower bound by restricting the considered class of input graphs further. In other words, we are looking for meta-theorems which would imply that all of FO or MSO logic can be solved in time not only linear in the size of the graph, but also depending more reasonably on the secondary parameters, if we are willing to give up some of the generality of the class of bounded-treewidth graphs. We concentrate on two graph classes: graphs of bounded vertex cover and graphs of bounded max-leaf number. We note that the investigation of the existence of stronger meta-theorems for these classes has been posed explicitly as an open problem by Fellows in [7].

Though graphs of bounded vertex cover or max-leaf number are considerably more restricted than bounded treewidth graphs, these classes are still interesting from the algorithmic point of view and the complexity of hard problems parameterized by vertex cover or max-leaf number has been investigated in the past ([9], [8]). Furthermore, as mentioned, strong lower bounds are known to apply to slightly more general classes: for bounded feedback vertex set and bounded pathwidth graphs even FO logic is non-elementary, while even for binary trees (thus for graphs of bounded treewidth and max degree) FO logic is at least triply exponential (again by [13]). Bounded vertex cover and bounded max-leaf number evade all these lower bound arguments so it's natural to ask what is exactly the complexity of FO and MSO logic for these classes of graphs?

The main results of this paper show that meta-theorems stronger than Courcelle's can indeed be shown for these classes of graphs. In addition, we show that our meta-theorems for vertex cover cannot be significantly improved under standard complexity assumptions.

Specifically, for the class of graphs of vertex cover bounded by k we show that

- All graph problems expressible with an FO formula ϕ can be solved in time linear in the graph size and singly exponential in k and $|\phi|$.

– All graph problems expressible with an MSO_1 formula ϕ can be solved in time linear in the graph size and doubly exponential in k and $|\phi|$.
– Unless n-variable 3SAT can be solved in time $2^{o(n)}$ (that is, unless the Exponential Time Hypothesis fails), then no $f(k,\phi)\cdot poly(|G|)$ algorithm exists to decide MSO logic on graphs for any $f(k,\phi) = 2^{2^{o(k+|\phi|)}}$.
– Unless FPT=W[1], there is no algorithm which can decide if an FO formula ϕ with q quantifiers holds in a graph G of vertex cover k in time $f(k,q)n^c$, for any $f(k,q) = 2^{O(k+q)}$.

Furthermore, for the class of graphs of max-leaf number bounded by k we show that

– All graph problems expressible with an FO formula ϕ can be solved in time linear in the graph size and singly exponential in k and $|\phi|$.

Our upper bounds rely on techniques different from the standard dynamic programming on decompositions usually associated with treewidth. For max-leaf number we rely on the characterization of bounded max-leaf number graphs from [16] also used heavily in [8] and the fact that FO logic has limited counting power in paths. For vertex cover we exploit an observation that for FO logic two vertices that have the same neighbors are "equivalent" in a sense we will make precise. We state our results in this case in terms of a new graph "width" parameter that captures this graph property more precisely than bounded vertex cover. We call the new parameter neighborhood diversity, and the upper bounds for vertex cover follow by showing that bounded vertex cover is a special case of bounded neighborhood diversity. Our essentially matching lower bounds on the other hand are shown for vertex cover. In the last section of this paper we prove some additional results for neighborhood diversity, beyond the algorithmic meta-theorems of the rest of the paper, which we believe indicate that neighborhood diversity might be a graph structure parameter of independent interest and that its algorithmic and graph-theoretic properties may merit further investigation.

2 Definitions and Preliminaries

Model Checking, FO and MSO logic In this paper we will describe algorithmic meta-theorems, that is, general methods for solving all problems belonging in a class of problems. However, the presentation is simplified if one poses this approach as an attack on a single problem, the model checking problem. In the model checking problem we are given a logic formula ϕ, expressing a graph property, and a graph G, and we must decide if the property described by ϕ holds in G. In that case, we write $G \models \phi$. Clearly, if we can describe an efficient algorithm for model checking for a specific logic, this will imply the existence of efficient algorithms for all problems expressible in this logic. Let us now give more details about the logics we will deal with and the graphs which will be our input instances.

Our universe of discourse will be labeled, colored graphs. Specifically, we assume that the first part of the input is an undirected graph $G(V, E)$, a set of labels L, each associated with a vertex of V and a set of subsets of V, $\mathcal{C} = \{C_1, C_2, \ldots, C_c\}$, which we refer to as color classes. The interesting case here is unlabeled, uncolored graphs (that is, $L = \mathcal{C} = \emptyset$), but the additional generality in the definition of the problem makes it easier to describe a recursive algorithm.

The formulas of FO logic are those which can be constructed using vertex variables, denoted usually by $x_i, y_i, \ldots$, vertex labels denoted by l_i, color classes denoted by C_i, the predicates $E(x_i, x_j)$, $x_i \in C_j$, $x_i = x_j$ operating on vertex variables or labels, standard propositional connectives and the quantifiers $\exists, \forall$ operating on vertex variables. The semantics are defined in the usual way, with the $E()$ predicate being true if $(x_i, x_j) \in E$.

For MSO logic the additional propery is that we now introduce set variables denoted by X_i and allow the quantifiers and the $\in$ predicate to operate on them. The semantics are defined in the obvious way.

If the set variables are allowed to range over sets of vertices only then the logic is sometimes referred to as MSO_1. A variation is MSO_2 logic, where one is also allowed to use set variables that range overs sets of edges. To accomodate for this case one also usually modifies slightly the definition of FO formulas to allow edge variables and an incidence predicate $I(v, e)$ which is true is true if edge e is incident on vertex v.

Bounded Vertex Cover and neighborhood diversity We will work extensively with graphs of bounded vertex cover, that is, graphs for which there exists a small set of vertices whose removal also removes all edges. We will usually denote the size of a graph's vertex cover by k. Note that there exist linear-time FPT algorithms for finding an optimal vertex cover in graphs where k is small (see e.g. [2]).

Our technique relies on the fact that in a graph of vertex cover k, the vertices outside the vertex cover can be partitioned into at most 2^k sets, such that all the vertices in each set have exactly the same neighbors outside the set and each set contains no edges inside it. Since we do not make use of any other special property of graphs of small vertex cover, we are motivated to define a new graph parameter, called neighborhood diversity, which intuitively seems to give the largest graph family to which we can apply our method in a straightforward way.

Definition 1. *We will say that two vertices v, v' of a graph $G(V, E)$ have the same type iff they have the same colors and $N(v) \setminus \{v'\} = N(v') \setminus \{v\}$.*

Definition 2. *A colored graph $G(V, E)$ has neighborhood diversity at most w, if there exists a partition of V into at most w sets, such that all the vertices in each set have the same type.*

Lemma 1. *If an uncolored graph has vertex cover at most k, then it has neighborhood diversity at most $2^k + k$.*

In Section 7 we will show that neighborhood diversity can be computed in polynomial time and also prove some results which indicate it may be an interesting parameter in its own right. However, until then our main focus will be graphs of bounded vertex cover. We will prove most of our algorithmic results in terms of neighborhood diversity and then invoke Lemma 1 to obtain our main objective. We will usually assume that a partition of the graph into sets with the same neighbors is given to us, because otherwise one can easily be found in linear time by using the mentioned linear-time FPT algorithm for vertex cover and Lemma 1.

Bounded Max-Leaf Number We say that a connected graph G has max-leaf number at most l if no spanning tree of G has more than l leaves. The algorithmic properties of this class of graphs have been investigated in the past [6, 10, 8]. In this paper we rely heavily on a characterization of bounded max-leaf graphs by Kleitman and West [16] which is also heavily used in [8].

Theorem 1. *[16] If a graph G has max-leaf number at most l, then G is a subdivision of a graph on $O(l)$ vertices.*

What this theorem tells us intuitively is that in a graph $G(V, E)$ with max-leaf number l there exists a set S of $O(l)$ vertices such that $G[V \setminus S]$ is a collection of $O(l^2)$ paths. Furthermore, only the internal vertices of the paths can be connected to vertices of S in G.

It is well-known that a graph of max-leaf number at most l has a path decomposition of width at most $2l$. Furthermore, it must have maximum degree at most l. Bounded max-leaf number graphs are therefore a subclass of the intersection of bounded pathwidth and bounded degree graphs (in fact, they are a proper subclass, as witnessed by the existence of say $2 \times n$ grids). Let us mention again that deciding FO logic on binary trees has at least a triply exponential parameter dependence, so the results we present for graphs of bounded max-leaf number can also be seen as an improvement on the currently known results for FO logic on bounded degree graphs, for this more restricted case.

3 FO Logic for Bounded Vertex Cover

In this Section we show how any FO formula can be decided on graphs of bounded vertex cover, with a singly exponential parameter dependence. Our main argument is that for FO logic, two vertices which have the same neighbors are essentially equivalent. We will state our results in the more general case of bounded neighborhood diversity and then show the corresponding result for bounded vertex cover as a corollary.

Lemma 2. *Let $G(V, E)$ be a graph and $\phi(x)$ a FO formula with one free variable. Let $v, v' \in V$ be two distinct unlabeled vertices of G that have the same type. Then $G \models \phi(v)$ iff $G \models \phi(v')$.*

Proof. (Sketch) Recall that the standard way of deciding an FO formula on a graph is, whenever we encounter an existential quantifier to try all possible choices of a vertex for that variable. This creates an n-ary decision tree with height equal to the number of quantifiers in $\phi(x)$. Every leaf corresponds to a choice of vertices for the q quantified variables, which makes the formula true or false. Internal nodes are evaluated as the disjunction (for existential quantifiers) or conjunction (for universal quantifiers) of their children.

It is possible to create a one-to-one correspondence between the trees for $\phi(v)$ and $\phi(v')$, by essentially exchanging v and v', showing that $\phi(v)$ and $\phi(v')$ are equivalent. □

Theorem 2. *Let ϕ be a FO sentence of quantifier depth q. Let $G(V, E)$ be a labeled colored graph with neighborhood diversity at most w and l labeled vertices. Then, there is an algorithm that decides if $G \models \phi$ in time $O((w + l + q)^q \cdot |\phi|)$.*

Corollary 1. *There exists an algorithm which, given a FO sentence ϕ with q variables and an uncolored, unlabeled graph G with vertex cover at most k, decides if $G \models \phi$ in time $2^{O(kq+q \log q)}$.*

Thus, the running time is (only) singly exponential in the parameters, while a straightforward observation that bounded vertex cover graphs have bounded treewidth and an application of Courcelle's theorem would in general have a non-elementary running time. Of course, a natural question to ask now is whether it is possible to do even better, perhaps making the exponent linear in the parameter, which is $(k + q)$. As we will see later on, this is not possible if we accept some standard complexity assumptions.

4 FO Logic for Bounded Max-Leaf Number

In this section we describe a model checking algorithm for FO logic on graphs of small max-leaf number. Our main tool is the mentioned observation that all but a small fraction of the vertices have degree 2, and therefore (since we assume without loss of generality that the graph is connected) induce paths. We call a maximal set of connected vertices of degree 2 a topo-edge.

Our main argument is that when a topo-edge is very long (exponentially long in the number of quantifiers of the first-order sentence we are model checking) its precise length does not matter.

To make this more precise, let us first define a similarity relation on graphs.

Definition 3. *Let G_1, G_2, be two graphs. For a given q we will say that G_1 and G_2 are q-similar and write $G_1 \sim_q G_2$ iff G_1 contains a topo-edge of order at least 2^{q+1} consisting of unlabeled vertices, call it P, and G_2 can be obtained from G_1 by contracting one of the edges of P. We denote the transitive closure of the relation $\sim_q$ as $\sim_q^*$.*

Our main technical tool is now the following lemma.

Lemma 3. *Let ϕ be a FO formula with q quantifiers. Then, for any two graphs G_1, G_2 if $G_1 \sim_q G_2$ then $G_1 \models \phi$ iff $G_2 \models \phi$. Therefore, if $G_1 \sim_q^* G_2$ then $G_1 \models \phi$ iff $G_2 \models \phi$.*

Now we are ready to state our main result of this section.

Theorem 3. *Let G be a graph on n vertices with max-leaf number k and ϕ a FO formula with q quantifiers. Then, there exists an algorithm for deciding if $G \models \phi$ running in time $poly(n) + 2^{O(q^2+q\log k)}$.*

Proof. By applying Theorem 1 we know that G can be partitioned into a set of at most $O(k)$ vertices of degree at least 3 and a collection of paths. By applying Lemma 3 we know that there exists a G' such that $G \sim_q^* G'$ and G' consists of the same $O(k)$ vertices of degree at least 3 and at most $O(k^2)$ paths whose length is at most 2^{q+1}. Of course, G' can be found in time polynomial in n.

Now, we can apply the straightforward algorithm to model check ϕ on G'. For every quantifier we have at most $O(k+q)+O(k(k+q)2^{q+1})$ choices, which is $2^{O(q+\log k)}$. Exhausting all possibilities for each vertex gives the promised running time. □

5 MSO Logic for Bounded Vertex Cover

First, let us state a helpful extension of the results of the Section 3. From the following Lemma it follows naturally that the model checking problem for MSO_1 logic on bounded vertex cover graphs is in XP, that is, solvable in polynomial time for constant ϕ and k, but our objective later on will be to do better.

Lemma 4. *Let $\phi(X)$ be an MSO_1 formula with a free set variable X. Let G be a graph and S_1, S_2 two sets of vertices of G such that all vertices of $(S_1 \setminus S_2) \cup (S_2 \setminus S_1)$ are unlabeled and have the same type and furthermore $|S_1 \setminus S_2| = |S_2 \setminus S_1|$. Then $G \models \phi(S_1)$ iff $G \models \phi(S_2)$.*

Our main tool in this section is the following lemma.

Lemma 5. *Let $\phi(X)$ be an MSO_1 formula with one free set variable X, q_V quantified vertex variables and q_S quantified set variables. Let G be a graph and S_1, S_2 two sets of vertices of G such that all vertices of $(S_1 \setminus S_2) \cup (S_2 \setminus S_1)$ are unlabeled and belong in the same type T. Suppose that both $|S_1 \cap T|$ and $|S_2 \cap T|$ fall in the interval $[2^{q_S} q_V, |T| - 2^{q_S} q_V - 1]$. Then $G \models \phi(S_1)$ iff $G \models \phi(S_2)$.*

Proof. (Sketch) We can assume without loss of generality that $S_1 \subseteq S_2$, thanks to Lemma 4. In fact, we may assume that $S_2 = S_1 \cup \{u\}$ for some vertex u and repeated applications of the same argument yield the claimed result.

Our argument is that the truth of $\phi(S_1)$ and $\phi(S_2)$ can be decided by an algorithm which checks for each set variable every combination of sizes that its intersection has with each type and for each vertex variable one representative of each type. If u is never picked as a representative then the algorithm must

give the same answer for $\phi(S_1)$ and $\phi(S_2)$. This can be guaranteed if the type u belongs in always has more than q_V vertices. Every time the algorithm picks a value for a set variable, the type u belongs in is partitioned in two. Since the only thing that matters to the algorithm is the size of the set's intersection with u's type, we are free to select a set that puts u in the larger of the two new sub-types. Because $S_1 \cap T$ and $S_2 \cap T$ have constrained sizes, we can always guarantee that u is never selected. □

Theorem 4. *There exists an algorithm which, given a graph G with l labels, neighborhood diversity at most w and an MSO_1 formula ϕ with at most q_S set variables and q_V vertex variables, decides if $G \models \phi$ in time $2^{O\left(2^{q_S}(w+l)q_S^2 q_V \log q_V\right)} \cdot |\phi|$.*

Corollary 2. *There exists an algorithm which, given an MSO_1 sentence ϕ with q variables and an uncolored, unlabeled graph G with vertex cover at most k, decides if $G \models \phi$ in time $2^{2^{O(k+q)}}$.*

Again, this gives a dramatic improvement compared to Courcelle's theorem, though exponentially worse than the case of FO logic. This is an interesting point to consider because for treewidth there does not seem to be any major difference between the complexities of model checking FO and MSO_1 logic.

The natural question to ask here is once again, can we do significantly better? For example, perhaps the most natural question to ask is, is it possible to solve this problem in $2^{2^{o(k+q)}}$? As we will see later on, the answer is no, if we accept some standard complexity assumptions.

Finally, let us briefly discuss the case of MSO_2 logic. In general this logic is more powerful than MSO_1, so it is not straightforward to extend Theorem 4 in this case. However, if we are not interested in neighborhood diversity but just in vertex cover we can observe that all edges in a graph with vertex cover of size k have one of their endpoints in one of the k vertices of the vertex cover. Thus, any edge set X can be written as the union of k edge sets. In turn, each of these k edge sets can easily be replaced by vertex sets, without loss of information, since we already know one of the endpoints of each of these edges. Using this trick we can replace every edge set variable in an MSO_2 sentence with k vertex set variables, thus obtaining a $2^{2^{O(kq)}}$ algorithm for MSO_2 logic on graphs of bounded vertex cover.

6 Lower Bounds

In this Section we will prove some lower bound results for the model checking problems we are dealing with. Our proofs rely on a construction which reduces SAT to a model checking problem on a graph with small vertex cover.

Given a propositional 3-CNF formula ϕ_p with n variables and m clauses, we want to construct a graph G that encodes its structure, while having a small vertex cover. The main problem is encoding numbers up to n with graphs of small

vertex cover but this can easily be achieved by using the binary representation of numbers.

We begin constucting a graph by adding $7 \log n$ vertices, call them $u_{(i,j)}, 1 \leq i \leq 7, 1 \leq j \leq \log n$. Add all edges of the form $(u_{(i,j)}, u_{(k,j)})$ (so we now have $\log n$ disjoint copied of K_7). Let $N_i = \{u_{(i,j)} \mid 1 \leq j \leq \log n\}$.

For every variable x_i in ϕ_p add a new vertex to the graph, call it v_i. Define for every number i the set $X(i) = \{j \mid$ the j-th bit of the binary representation of i is $1\}$. Add the edges $(v_i, u_{(1,j)}), j \in X(i)$, that is connect every variable vertex with the vertices of N_1 that correspond to the binary representation of its index. Let $U = \{v_i \mid 1 \leq i \leq n\}$ be the vertices corresponding to variables.

For every clause c_i in ϕ_p add a new vertex to the graph, call it w_i. If the first literal in c_i is a positive variable x_k then add the edges $(w_i, u_{(2,j)}, j \in X(k)$. If the first literal is a negated variable $\neg x_k$, add the edges $(w_i, u_{(3,j)}, j \in X(k)$. Proceed in a similar way for the second and third literal, that is, if the second literal is positive connect w_i with the vertices that correspond to the binary representation of the variable in N_4, otherwise in N_5. For the third literal do the same with N_6 or N_7. Let $W = \{w_i \mid 1 \leq i \leq m\}$ be the vertices corresponding to clauses.

Finally, set the color classes to be $\{N_1, N_2, \ldots, N_7, U, W\}$.

Now, looking at the graph it is easy to see if a vertex v_i corresponds to a variable that appears positive in the clause represented by a vertex w_i. They must satisfy the formula

$$pos(v_i, w_j) = \bigvee_{k=2,4,6} \forall x(x \in N_1 \rightarrow \exists y((E(v_i, x) \leftrightarrow E(w_j, y)) \wedge y \in N_k \wedge E(x, y)))$$

It is not hard to define $neg(v_i, w_j)$ in a similar way. Now it is straight-forward to check if ϕ_p was satisfiable:

$$\begin{aligned} \phi = \exists S(\forall x x \in S \rightarrow x \in U) \wedge (\forall w w \in W \rightarrow \exists x x \in U \wedge \\ ((pos(x, w) \wedge x \in S) \vee (neg(x, w) \wedge x \notin S))) \end{aligned}$$

Clearly, ϕ holds in the constructed graph iff ϕ_p is satisfiable. S corresponds to the set of variables set to true in a satisfying assignment. Let us also briefly remark that it is relatively easy to eliminate the colors and labels from the construction above, therefore the lower bounds given below apply to the natural form of the problem.

Lemma 6. *$G \models \phi$ iff ϕ_p is satisfiable. Furthermore, ϕ has size $O(1)$ and G has a vertex cover of size $O(\log n)$.*

Theorem 5. *Let ϕ be a MSO formula with q quantifiers and G a graph with vertex cover k. Then, unless 3-SAT can be solved in time $2^{o(n)}$, there is no algorithm which decides if $G \models \phi$ in time $O(2^{2^{o(k+q)}} \cdot poly(n))$.*

Theorem 6. *Let ϕ be a FO formula with q_v vertex quantifiers and G a graph with vertex cover k. Then, unless FPT=W[1], there is no algorithm which decides if $G \models \phi$ in time $O(2^{O(k+q_v)} \cdot poly(n))$.*

7 Neighborhood Diversity

In this Section we give some general results on the new graph parameter we have defined, neighborhood diversity. We will use $nd(G), tw(G), cw(G)$ and $vc(G)$ to denote the neighborhood diversity, treewidth, cliquewidth and minimum vertex cover of a graph G. We will call a partition of the vertex set of a graph G into w sets such that all vertices in every set share the same type a neighborhood partition of width w.

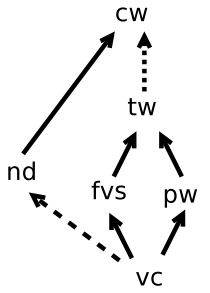
First, some general results

Theorem 7. *1. Let $V_1, V_2, \ldots, V_w$ be a neighborhood partition of the vertices of a graph $G(V, E)$. Then each V_i induces either a clique or an independent set. Furthermore, for all i, j the graph either includes all possible edges from V_i to V_j or none.*

2. For every graph G we have $nd(G) \leq 2^{vc(G)} + vc(G)$ and $cw(G) \leq nd(G) + 1$. Furthermore, there exist graphs of constant treewidth and unbounded neighborhood diversity and vice-versa.

3. There exists an algorithm which runs in polynomial time and given a graph $G(V, E)$ finds a neighborhood partition of the graph with minimum width.

Taking into account the observations of Theorem 7 we summarize what we know about the graph-theoretic and algorithmic properties of neighborhood diversity and related measures in Figure 1.



	FO	MSO	MSO_2
Cliquewidth	$tow(w)$	$tow(w)$	$tow(w)$
Treewidth	$tow(w)$	$tow(w)$	$tow(w)$
Vertex Cover	$2^{O(w)}$	$2^{2^{O(w)}}$	$2^{2^{O(w)}}$
Neighborhood Diversity	$poly(w)$	$2^{O(w)}$	Open

Fig. 1. A summary of the relations between neighborhood diversity and other graph widths. Included are cliquewidth, treewidth, pathwidth, feedback vertex set and vertex cover. Arrows indicate generalization, for example bounded vertex cover is a special case of bounded feedback vertex set. Dashed arrows indicate that the generalization may increase the parameter exponentially, for example treewidth w implies cliquewidth at most 2^w. The table summarizes the best known model checking algorithm's dependence on each width for the corresponding logic.

There are several interesting points to make here. First, though our work is motivated by a specific goal, beating the lower bounds that apply to graphs of bounded treewidth by concentrating on a special case, it seems that what we have achieved is at least somewhat better; we have managed to improve the algorithmic meta-theorems that were known by focusing on a class which is not necessarily smaller than bounded treewidth, only different. However, our class is a special case of another known width which generalizes treewidth as well, namely cliquewidth. Since the lower bound results which apply to treewidth apply to cliquewidth as well, this work can perhaps be viewed more appropriately as an improvement on the results of [4] for bounded cliquewidth graphs.

Second, is the case of MSO_2 logic. The very interesting hardness results shown in [11] demonstrate that the tractability of MSO_2 logic is in a sense the price one has to pay for the additional generality that cliquewidth provides over treewidth. It is natural to ask if the results of [11] can be strengthened to apply to neighborhood diversity or MSO_2 logic can be shown tractable parameterized by neighborhood diversity.

Though we cannot yet fully answer the above question related to MSO_2, we can offer some first indications that this direction might merit further investigation. In [11] it is shown that MSO_2 model checking is not fixed-parameter tractable when the input graph's cliquewidth is the parameter by considering three specific MSO_2-expressible problems and showing that they are W-hard. The problems considered are Hamiltonian cycle, Graph Chromatic Number and Edge Dominating Set. We can show that these three problems can be solved efficiently on graphs of small neighborhood diversity. Since small neighborhood diversity is a special case of small cliquewidth, where these problems are hard, this result could be of independent interest.

Theorem 8. *Given a graph G whose neighborhood diversity is w, there exist algorithms running in time $O(f(w) \cdot poly(|G|))$ that decide Hamiltonian cycle, Graph Chromatic Number and Edge Dominating Set.*

8 Conclusions and Open Problems

In this paper we presented algorithmic meta-theorems which improve the running times implied by previously known meta-theorems for more restricted inputs. In this way we have partially explored the trade-off which can be achieved between running time and generality. This is an interesting area for further investigations and much more can be done.

For bounded max-leaf number the complexity of MSO logic is unknown. Quite likely, it is possible to improve upon the Courcelle's theorem for this case as well, but the problem remains open. Also, it would be nice to obtain a lower bound for FO logic in this case showing that it is impossible to achieve $2^{o(q^2)}$, i.e. that the exponent must be quadratic. For neighborhood diversity the most interesting open problem is the complexity of MSO_2.

Going further, it would also make sense to investigate whether restricting the model checking problem to graphs of bounded vertex cover or max-leaf number

can also allow us to solve logics wider than MSO_2. Some indications that this may be possible are given in [9]

References

1. S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991.
2. J. Chen, I. A. Kanj, and G. Xia. Improved parameterized upper bounds for vertex cover. In R. Kralovic and P. Urzyczyn, editors, *MFCS*, volume 4162 of *Lecture Notes in Computer Science*, pages 238–249. Springer, 2006.
3. B. Courcelle. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Inf. Comput.*, 85(1):12–75, 1990.
4. 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.
5. A. Dawar, M. Grohe, S. Kreutzer, and N. Schweikardt. Approximation schemes for first-order definable optimisation problems. In *LICS*, pages 411–420. IEEE Computer Society, 2006.
6. V. Estivill-Castro, M. R. Fellows, M. A. Langston, and F. A. Rosamond. FPT is P-Time Extremal Structure I. In H. Broersma, M. Johnson, and S. Szeider, editors, *ACiD*, volume 4 of *Texts in Algorithmics*, pages 1–41. King's College, London, 2005.
7. M. R. Fellows. Open problems in parameterized complexity, AGAPE spring school on fixed parameter and exact algorithms, 2009.
8. M. R. Fellows, D. Lokshtanov, N. Misra, M. Mnich, F. A. Rosamond, and S. Saurabh. The Complexity Ecology of Parameters: An Illustration Using Bounded Max Leaf Number. *Theory Comput. Syst.*, 45(4):822–848, 2009.
9. M. R. Fellows, D. Lokshtanov, N. Misra, F. A. Rosamond, and S. Saurabh. Graph layout problems parameterized by vertex cover. In S.-H. Hong, H. Nagamochi, and T. Fukunaga, editors, *ISAAC*, volume 5369 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2008.
10. M. R. Fellows and F. A. Rosamond. The Complexity Ecology of Parameters: An Illustration Using Bounded Max Leaf Number. In S. B. Cooper, B. Löwe, and A. Sorbi, editors, *CiE*, volume 4497 of *Lecture Notes in Computer Science*, pages 268–277. Springer, 2007.
11. F. V. Fomin, P. A. Golovach, D. Lokshtanov, and S. Saurabh. Clique-width: on the price of generality. In C. Mathieu, editor, *SODA*, pages 825–834. SIAM, 2009.
12. M. Frick and M. Grohe. Deciding first-order properties of locally tree-decomposable structures. *J. ACM*, 48(6):1184–1206, 2001.
13. M. Frick and M. Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130(1-3):3–31, 2004.
14. M. Grohe. Logic, graphs, and algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(091), 2007.
15. P. Hlinený, S. il Oum, D. Seese, and G. Gottlob. Width parameters beyond tree-width and their applications. *Comput. J.*, 51(3):326–362, 2008.
16. D. Kleitman and D. West. Spanning trees with many leaves. *SIAM Journal on Discrete Mathematics*, 4:99, 1991.
17. S. Kreutzer and S. Tazari. On brambles, grid-like minors, and parameterized intractability of monadic second order logic. In *SODA*, 2010.

A Omitted Proofs

A.1 Proof of Lemma 1

Proof. Construct k singleton sets, one for each vertex in the vertex cover and at most 2^k additional sets, one for each subset of vertices of the vertex cover. Place each of the vertices of the independent set in one of these sets, specifically the one which corresponds to its neighborhood in the vertex cover. □

A.2 Proof of Lemma 2

Proof. Suppose without loss of generality that $\phi(x)$ is in prenex normal form and has quantifier depth q. We remind the reader that the computation for $\phi(v)$ can be evaluated by means of a rooted n-ary computation tree of height q, where $n = |V|$. Informally, the children of the root represent the n possible choices for the first quantified variable of the formula, their children the choices for the second and so on. Each leaf represents a possible q-tuple of choices for the variables and makes the formula true or false. Internal nodes compute a value either as the logical disjunction of their children (for existentially quantified variables) or the logical conjunction (for universally quantified variables). The value computed at the root is the truth value of $\phi(v)$.

We will prove the statement by showing a simple correspondence between the computation trees for $\phi(v)$ and $\phi(v')$. Let T and T' be the two trees, and label every node of each tree at distance i from the root with a different tuple of i vertices of G (note that the labels of the tree are not to be confused with the labels of G). Let $sw_{v,v'} : \bigcup_{i=1,\ldots,q} V^i \to \bigcup_{i=1,\ldots,q} V^i$ be the "swap" function which when given a tuple of vertices of V, returns the same tuple with all occurences of v replaced by v' and vice-versa. As a shorthand, when Q is a tuple of vertices and u a vertex we will write (Q, u) to mean the tuple containing all the elements of Q with u added at the end. With this notation the children of a node with label Q are the nodes with labels in the set $\{(Q, u) \mid u \in V\}$.

Every leaf in both trees has a q-tuple as a label. Let Q_1 be such a q-tuple which is the label of a leaf in T and $sw_{v,v'}(Q_1)$ the tuple we get from Q_1 by swapping v with v'. The claim is that the leaf of T with label Q_1 and the leaf of T' with label $sw_{v,v'}(Q_1)$ evaluate to the same value. In other words, if we take $\phi(v)$ and replace all quantified variables with the vertices of Q_1 the formula will evaluate to the same result as when we replace all the quantified variables of $\phi(v')$ with the vertices of $sw_{v,v'}(Q_1)$. This is true because ϕ is a boolean function of edge, color and equality predicates; color predicates and edge predicates involving one of v, v' with another vertex are unaffected by swapping v and v', since these two have the same neighbors and belong in the same color classes. Equality predicates are also unaffected since all occurences of v are replaced by v' and vice-versa, thus equality predicates involving these two and some other vertex will still evaluate to false, while predicates only involving these two will be unaffected because equality is symmetric. Finally, edge predicates involving only v and v' are unaffected since $E()$ is symmetric. Thus, we have established

a one-to-one correspondence between the leaves of T and T' via the function $sw_{v,v'}$, preserving truth values.

Now, we need to establish a correspondence between the internal nodes, again via $sw_{v,v'}$. Consider a node of T with label Q_1 and the node of T' with label $sw_{v,v'}(Q_1)$. The children of the former have labels in the set $C_1 = \{(Q_1, u) \mid u \in V\}$. The children of the latter have labels in $C_2 = \{(sw_{v,v'}(Q_1), u) \mid u \in V\}$. It is not hard to see that $C_2 = \{sw_{v,v'}(Q) \mid Q \in C_1\}$, or in other words, the correspondence between nodes is transferred up the levels of the trees.

The only remaining part is to establish that if two nodes in T and T' have labels corresponding via $sw_{v,v'}$, then they compute the same value. We already established this for the leaves. For internal nodes, this follows from the fact that the sets of children of two corresponding nodes are also in one-to-one correspondence via $sw_{v,v'}$ and that the nodes are both of the same type (existential or universal) since only nodes at the same level can be corresponding. Thus, by an inductive argument, all the children of the roots of the two trees compute the same values and therefore $\phi(v)$ and $\phi(v')$ are equivalent. □

A.3 Proof of Theorem 2

Proof. We will rely heavily on Lemma 2 and describe an inductive argument. If $q = 0$ the problem is of course trivial so assume that $q > 0$ and the theorem holds for sentences of depth at most $q-1$. Also, assume wlog that ϕ is in prenex normal form and furthermore, that $\phi = \exists x \psi(x)$, since the universal case can be easily decided if we solve the existential case, by deciding on the negation of ϕ.

Suppose that V can be partitioned into $V_1, V_2, \ldots V_w$ as required by the definition of neighborhood diversity. Now, by Lemma 2 if $v, v' \in V_i$ for some i, and neither of the two is labeled then $G \models \psi(v)$ iff $G \models \psi(v')$. Thus, we need to model check at most $(w + l)$ sentences of $q - 1$ quantifiers to decide ϕ: we try replacing x with each of the l labeled vertices or with one arbitrarily chosen representative from each V_i. In the process we introduce a new label. Repeating this process constructs a computation tree with at most $\prod_{i=0}^{q-1}(w + l + i) = O\left((w + l + q)^q\right)$ leaves. The result of the computation tree can be evaluated in time linear in its size. □

A.4 Proof of Lemma 3

Proof. We will prove the first statement by induction on q and the second statement follows directly from it. For $q = 0$ the statement is trivial because ϕ can only refer to labeled vertices and G_1, G_2 are identical with respect to these vertices.

Suppose that the statement is true for at most $q - 1$ quantifiers. It suffices to show the statement for q quantifiers for a formula ϕ of the form $\exists x \psi(x)$, and the statement then easily follows for formulas which are boolean combinations of formulas of at most $q - 1$ quantifiers. So, suppose that $G_1 \models \exists x \psi(x)$. This means that there exists a vertex in G_1 such that if we label it with a new label

l to obtain a graph G'_1 (which is G_1 with the label l added) we have $G'_1 \models \psi(l)$. Now we must take cases for the vertex where l is placed.

If l is placed on a vertex outside of P then it is not hard to see that $G_2 \models \phi$: we place l on the same vertex on G_2 (and obtain G'_2) and now we have $G'_1 \sim_{(q-1)} G'_2$ so from the inductive hypothesis $G'_2 \models \psi(l)$.

Now the interesting case is when l is placed on a vertex of P. Number the vertices of P from 1 to $|P|$, starting from one of the endpoints of the path induced by P. Partition P into two parts: P_2 contains the last 2^q vertices and P_1 the rest. In G_2 we use the same numbering for the vertices of the path (of course now the numbering is from 1 to $|P| - 1$, since on edge has been contracted).

Suppose that l is placed on a vertex of P_1. We place l on the same vertex in G_2. Now, we have $G'_1 \sim_{(q-1)} G'_2$, because in both graphs P has been broken into two paths P' and P''. P' has the same size on both (depending on the position where l was placed) and P'' has size at least 2^q on G'_1 and one less than that on G'_2. So, by the inductive hypothesis $G'_1 \models \psi(l)$ iff $G'_2 \models \psi(l)$.

Finally, if l is placed on a vertex of P_2 we place l on a vertex of G_2 that has the same distance from the end of the path and two q-similar graphs G'_1, G'_2 are obtained, because the smaller part of the two into which P is broken has the same size on both graphs and the larger has size at least 2^q. So by the inductive hypothesis $G'_1 \models \psi(l)$ iff $G'_2 \models \psi(l)$.

The converse directions where we know that $G_2 \models \phi$ and need to show that this implies $G_1 \models \phi$ can be established with a similar argument. □

A.5 Proof of Lemma 4

Proof. The proof follows ideas similar to those of Lemma 2. Suppose that $\phi(X)$ has q quantifiers in total, then it is possible to decide if $G \models \phi(S)$ using a computation tree such that for each quantified variable we have nodes in the tree with n children and for each quantified set variable we have nodes with 2^n children, with each child corresponding to a possible choice for that variable. Again, we can label each node of the tree with a tuple of at most q elements, but now the elements can be either individual vertices or sets of vertices.

Observe that it suffices to prove the claim when $|S_1 \setminus S_2| = |S_2 \setminus S_1| = 1$, because then we can apply the claim repeatedly to transform S_1 to S_2 by exchanging the different vertices one by one. So, suppose that $S_1 \setminus S_2 = v$ and $S_2 \setminus S_1 = v'$, and v and v' have the same type.

Now, the $sw_{v,v'}$ function of Lemma 2 can be extended to act on sets of vertices in a straightforward way. Consider the computation trees for $\phi(S_1)$ and $\phi(S_2)$. Once again we must show that $sw_{v,v'}$ is a one-to-one correspondence between the leaves of the two trees that preserves truth values. For edge and equality predicates we can use the same arguments as in Lemma 2, so the only difference can be with predicates of the form $x \in X$. However, it is not hard to see that the truth values of these is not affected when $x \neq v, v'$ and also when X is one of the supplied colors of the graph, since v, v' have the same colors. Finally, the truth value is also unaffected if X is a variable set, since $sw_{v,v'}$ is applied both

to vertex and set variables. Now, the correspondence is lifted up the levels of the tree using similar arguments and this completes the proof. □

A.6 Proof of Lemma 5

Proof. We are dealing with the case where two sets are different, but their different elements are all of the same type. To give some intuition, in the base case of $q_S = 0$ for this particular type both sets have the property that the sets themselves and their complements have at least q_V vertices of the type. This will prove important because $\phi(X)$ will be a FO sentence after we decide on a set for X and as we will see an FO sentence cannot distinguish between two different large enough sets (informally, we could say that an FO sentence with q quantifiers can only count up to q). We will show how to extend this to general q_S by shrinking the interval of sizes where we claim that sets are equivalent, because every set variable X_i essentially doubles the amount we can count, by partitioning vertices into two sets, those in X_i and those in its complement.

First, assume without loss of generality that $|S_1| \leq |S_2|$. Now because of Lemma 4 we can further assume without loss of generality that $S_1 \subseteq S_2$, because there exists a set S_2' of the same size as S_2 such that $S_1 \subseteq S_2'$ and $G \models \phi(S_2)$ iff $G \models \phi(S_2')$. Furthermore, we may focus on the case where $S_2 = S_1 \cup \{u\}$ for some vertex $u \notin S_1$, because if we prove the statement for sets whose sizes only differ by 1, then we can apply it repeatedly to get the statement for sets which have a larger difference.

We will now rely on Lemma 4 to construct an XP algorithm for deciding $\phi(S_1)$ and $\phi(S_2)$. The trivial algorithm we have already discussed would consider 2^n sets every time a set variable has to be assigned a value and n vertices every time a vertex variable has to be assigned a value. However, because of Lemma 4 we can consider only $O(2^l n^w)$ different assignments for a set variable. This is because the equivalence between different sets of the same size established allows us to sample one set for each combination of sizes that the set will have with each of the w types (the 2^l factor comes from the fact that labeled vetices are "special" and we have to decide for each one individually). Note though that deciding on an assignment of a set can in the worst case double w, since we are adding a new color to the graph representing the set. Thus, for the next set we would have to consider $O(2^l n^{2w})$ choices and so on. Furthermore, from the proof of Lemma 4 it is straightforward to derive a slightly stronger version of Lemma 2 which holds for MSO sentences. Using this we conclude that we need to check through $w + l$ samples when we are deciding on a vertex variable and this introduces a new label.

Suppose that we use the algorithm sketched above to decide $\phi(S_1)$ and $\phi(S_2)$. The crucial point now is that this algorithm has a lot of freedom in picking the sample sets and vertices it considers. In particular, when assigning value to a vertex variable the algorithm can always avoid the vertex u if there are still other vertices of the same type. It is not hard to see that if the algorithm never assigns u to any vertex variable when deciding $\phi(S_1)$ and $\phi(S_2)$ the result will necessarily

be the same for both sentences. So we need to argue why the algorithm can always avoid using u.

To achieve this we can exploit the freedom the algorithm has when picking sets. Every time the algorithm picks a set to be considered the set of vertices of the same type as u is partitioned into two sets. Because it does not matter which vertices are included in a set and only the size of the set's partition with a type matters, we can make sure that u is always placed in the larger of the two new types by exchanging with another vertex appropriately. Because of the restriction on the sizes of S_1 and S_2 we know that initially u belongs in a type shared by at least $2^{q_S} q_V$ other vertices. It is not hard to see that this invariant is maintained by the algorithm when picking a set if we place u in the larger of the two new types when picking a set and we pick a different sample from its type when we pick a vertex. Thus, we have established that there exists an algorithm that decides $\phi(S_1)$ and $\phi(S_2)$ without ever assigning u to a vertex variable, which means that the algorithm must decide the same value for both sentences. □

A.7 Proof of Theorem 4

Proof. Our algorithm now will rely heavily on Lemma 5. When picking an assignment for a set variable, for each of the w types of vertices we need to decide on the size of its intersection with the set. Because of Lemma 5 we can limit ourselves to considering $2^{q_S+1} q_V$ different sizes for the first set, which gives $(2^{q_S+1} q_V)^w$ choices for the first set variable. However, because every time we decide on a set we start working on a graph with one more color, the number of vertex types may at most double. From these we can derive an easy upper bound on the number of alternatives we will consider for each set variable as $2^{2^{q_S} w(q_S+1+\log q_V)}$. Since we have q_S set variables in total this gives $2^{q_S 2^{q_S} w(q_S+1+\log q_V)}$. For each vertex variable we have to consider at most $2^{q_S} w + l + q_V$ alternatives, so for all q_V variables at most $(2^{q_S} w + l + q_V)^{q_V}$. The product of these two upper bounds is an upper bound on the total number of alternatives our algorithm will consider, giving the promised running time. □

A.8 Proof of Theorem 5

Proof. We have already observed that the construction we described has $k = O(\log n)$. Since the construction can clearly be performed in polynomial time, an algorithm running in time $O(2^{2^{o(k+q)}} \cdot poly(n))$ would imply an algorithm for SAT running in $2^{o(n)} \cdot poly(n)$. □

A.9 Proof of Theorem 6

Proof. We use the same construction, but begin our reduction from Weighted 3-SAT, a well-known W[1]-hard parameterized problem. Suppose we are given a 3-CNF formula and a number w and we are asked if the formula can be satisfied

by setting exactly w of its variables to true. The formula ϕ we construct is exactly the same, except that we replace the $\exists S$ with $\exists x_1 \exists x_2 \ldots \exists x_w (\bigwedge_{1 \leq i < j \leq w} x_i \neq x_j)$ and all occurences of $x \in S$ with $\bigvee_{1 \leq i \leq w} x = x_i$. It is not hard to see that the informal meaning of ϕ now is to ask whether there exists a set of exactly w distinct variables such that setting them to true makes the formula true.

We now have $q_v = w + O(1)$ so an algorithm running in time $2^{O(k+q_V)} \cdot poly(n)$ would imply an algorithm for Weighted 3-SAT running in $2^{O(w)} \cdot poly(n)$, and thus that FPT=W[1]. □

A.10 Proof of Theorem 7

Proof. For the first statement, to show that every V_i induces either a clique or an independent set, we may assume that $|V_i| \geq 3$, otherwise the statement is trivial. Suppose that some V_i includes at least one edge (u, v). Then for every other pair of vertices w, w' we know that w must be connected to v since w and u have the same type. With a symmetric argument we conclude that all the edges $(w, u), (w, v), (w', u), (w', v)$ must exist in the graph. Finally, because w and u have the same type and we concluded that (w', u) is an edge, we must have (w, w') as well. This is true for any pair of vertices (w, w') so if V_i has at least one edge it is a clique. Another way to see this observation is to say that the property of two vertices having the same type is an equivalence relation.

For the edges between V_i and V_j, suppose that there exists at least an edge (u, v) between them and let $w \in V_i$, $w' \in V_j$. v has the same type as w', therefore (u, w') must be an edge. Now, w has the same type as u so (w, w') must also be an edge, and once again this is true for any w, w'.

We have already shown the first part of the second statement. For the part with cliquewidth, we remind the reader that the graphs of cliquewidth k are those which can be constructed by repeated application of the following operations: introducing a new vertex with a label in $\{1, \ldots, k\}$, joining all vertices of label i with all vertices of label j, renaming all vertices of label i to label j and taking disjoint union of two graphs of cliquewidth at most k. We must show how to construct a graph in such a way starting from a neighborhood partition of width w, using at most $w + 1$ labels. The labels in $\{1, \ldots, w\}$ will only be used for the vertices of the corresponding set in the partition, while the extra label will be used to construct the cliques. For each V_i, if V_i is an independent set introduce $|V_i|$ new vertices with label i. If V_i is a clique repeat $|V_i|$ times: introduce a new vertex of label $w + 1$, join all vertices of label i to $w + 1$ and rename $w + 1$ to i. After all the vertices have been introduced, for all i, j for which the graph had all edges between V_i and V_j join the vertices labeled i with those labeled j.

To see why treewidth is incomparable to neighborhood diversity consider the examples of a complete bipartite graph $K_{n,n}$ and a path on n vertices.

Finally, let us argue why neighborhood diversity is computable in polynomial time. First, observe that neighborhood diversity is closed under the taking of induced subgraphs, that is, if $G'(V', E')$ is an induced subgraph of $G(V, E)$ then $nd(G') \leq nd(G)$, because a neighborhood partition of G is also valid for G'. We will work inductively: order the vertices of the input graph G in an arbitrary way

and suppose that we have found an optimal neighborhood partition of the graph induced by the first k vertices into w sets, $V_1, V_2, \ldots, V_w$. From our observation regarding induced subgraphs we know that the optimal partition of the graph induced by the first $k+1$ vertices will need at least w sets. Let u be the next vertex. There are two cases: either u can be placed in some V_i giving us a valid and optimal neighborhood partition of the first $k+1$ vertices or not, and this can easily be verified in polynomial time. In the second case, there must exist in each V_i a vertex v_i such that v_i and u have different types. This means that we have a set of $w+1$ vertices which have mutually incompatible types, which implies that the optimal neighborhood partition needs at least $w+1$ sets. This can be achieved by adding to the partition we have a new singleton set $\{u\}$. □

A.11 Proof of Theorem 8

Proof. We will make use of an auxiliary graph G' on w vertices. Each vertex of G' corresponds to a set in an optimal neighborhood partition of G and two vertices of G' have an edge iff the corresponding sets of the partition of G have all possible edges between them.

First, for the chromatic number. Observe that if a set V_i of a neighborhood partition of G induces an independent set, we can delete all of its vertices but one, without affecting the graph's chromatic number, because there always exists an optimal coloring where all the vertices of V_i take the same color. So, we can assume without loss of generality that all the sets V_i of a neighborhood partition of G induce cliques (some of them of order one).

Consider now a coloring of the graph G' with the following objective function: for each color i used, its weight is the size of the largest clique that corresponds to a vertex of G' colored with i. The objective is to minimize the sum of the weights of the colors used. It is not hard to see that this problem can be solved in time $O(w^w \cdot \log n)$ by checking through all possible colorings of the vertices of G'. Also, from such a coloring of G' we can infer a coloring of G that uses as many colors as the weight of the coloring: for every color i used in G' create a new set of colors of size equal to the color's weight. This is sufficient to color all the cliques of G that correspond to vertices of G' colored with i.

What remains is to argue why this leads to an optimal coloring. Suppose we have an optimal coloring of G and order the sets of a neighborhood partition in order of decreasing size, that is, $|V_1| \geq |V_2| \geq \ldots \geq |V_w|$. We will say that V_i and V_j have "similar" colors in this optimal coloring of G when there is a color that appears in both V_i and V_j. From the coloring of G we infer a coloring of G' as follows: while there are still uncolored vertices of G', take the first set of the partition of G (in order of size) that corresponds to a still uncolored vertex of G'. Use a new color for its corresponding vertex in G' and also for all the vertices that correspond to sets with colors similar to it.

When we are done, we will have a proper coloring of G', because if two sets V_i,V_j are joined by an edge they cannot have similar colors. Furthermore, the weight of the coloring of G' we obtain is a lower bound on the number of colors used in the original coloring of G we assumed. This is because when we pick a

set V_i and use it to introduce a new color we know that it does not have similar colors with any of the sets we have picked so far. Because all the sets picked induce cliques and do not have similar colors (i.e. no color is reused) we know that the original coloring of G uses at least as many colors as the sum of the sizes of the sets picked. Thus, if our algorithm found that the optimal solution to the weighted coloring problem for G' has weight w, this means that w colors are needed to color G, because a coloring of G with $w-1$ colors would give a solution to the coloring problem of G' with weight at most $w-1$.

For the Hamiltonian cycle problem, we will once again use the graph G'. We define the weight of every vertex of G' to be the size of its corresponding set in the neighborhood partition of G. Now, the problem of finding a Hamiltonian cycle in G can be reduced to the problem of finding a closed walk of G', such that every vertex that corresponds to an independent set is visited a number of times exactly equal to its weight, while every vertex corresponding to a clique is visited at least once and at most as many times as its weight.

This problem of looking for a walk on G' can be solved in time $O(n^{w^2})$. Replace each edge with two directed arcs of opposite direction. Now, for each of the at most w^2 arcs, we must decide how many times it will be used, a value upper-bounded by n. If we have decided on such values for all arcs we can easily check if a walk with the desired properties can be made from them. Replace each arc with a number of parallel arcs of the same direction equal to the value decided for it. Now, we can obtain a walk if the resulting multi-graph is Eulerian (that is, all vertices have the same in-degree as out-degree) and also the in-degrees of the vertices follow the conditions we have stated for the number of times the vertex must be visited.

In order to improve this to an FPT algorithm, we rely on an old but seminal result by Lenstra which states that the feasibility of an ILP programs of size n with k variables can be solved in time $f(k) \cdot poly(n)$, i.e. bounded-variable ILP is FPT (this is also the main tool used in [9]). This is a result that has attracted considerable interest in the parameterized complexity community and it has long been a topic of interest to find examples of its application. Here we observe that in the above algorithm we are trying to decide on values for w^2 variables. For each variable the constraints can easily be expressed as linear inequalities: for each vertex we have to make sure that the in-degree is equal to the out-degree and also that the in-degree falls in a specified interval. Therefore, by expressing our problem as a system of linear inequalities we obtain an FPT algorithm.

Finally, in the edge dominating set problem, we are asked to find a set of edges of minimum size such that all other edges share an endpoint with one of the edges we selected. This problem is equivalent to the minimum maximal matching problem, where we are trying to find a minimum size independent set of edges that cannot be extended by picking another edge of the graph. To see why the optimal solution to the edge dominating set problem is always a matching, suppose that we have a solution S which includes two edges $(u, v), (u, v')$. Now, if all the neighbors of v' are incident on an edge of S we can simply remove (u, v') from S and improve the size of the solution. If there is a neighbor w of v'

that is not incident on an edge of S we can replace (u, v') with (w, v') in S. To see why a solution to the edge dominating set problem is a maximal matching, suppose that it was not. Then there would be two unmatched vertices connected by an edge, which would imply that this edge is not dominated.

Our algorithm will proceed as follows: for every minimal vertex cover V' of G' repeat the following (there are at most 2^w vertex covers to be considered): from V' infer a vertex cover of G by placing into the vertex cover all the vertices that belong in a type whose corresponding vertex is in V'. Also place in the vertex cover all but one (arbitrarily chosen) vertex of every vertex type that induces a clique but whose corresponding vertex is not in V'. Call the resulting vertex cover of G V''. Find a maximum matching on the graph induced by V'', call it M_1. Take the bipartite graph induced by the unmatched vertices of V'' and $V \setminus V''$ and find a maximum matching there, call it M_2. The solution produced is $M_1 \cup M_2$. After repeating this for all vertex covers of G', pick the smallest solution.

Now we need to argue why this solution is optimal. Let S be an optimal solution for G. We say that a set of the neighborhood partition V_i is full if all of its vertices are incident on edges of S. If we take in G' the corresponding vertices of the full sets of G, they must form a vertex cover of G', otherwise there would be two neighboring vertices with neither having any edge of S incident to it, which would mean that S is not maximal. This is a vertex cover of G' considered by our algorithm, since our algorithm considers all vertex covers of G', call it V'. Let V'' be again the vertex cover of G our algorithm derived from V' by also including a minimal number of vertices from each remaining clique. Let V^* be the set of vertices of G incident on some edge of S, which must also be a vertex cover of G. Without loss of generality we will assume that $V'' \subseteq V^*$, because the two vertex covers of G agree on taking all vertices of the full sets and V'' takes a minimal number of vertices from every other clique. Even if V^* leaves out a different vertex from some clique because all the vertices of the clique have the same neighbors we can apply an exchanging argument and transform S appropriately without increasing its size so that both sets leave out the same vertex.

Now note that $|M_2| \leq |V''| - 2|M_1|$. So our algorithm's solution has size at most $|V''| - |M_1|$. On the other hand the optimal solution S includes some edges with both endpoints in V'', call this set S_1. Because M_1 is a maximum matching, $|S_1| \leq |M_1|$. From what we have so far, the fact that all vertices of V^* are matched by S and the fact that V'' is a vertex cover, so $V^* \setminus V''$ induces no edges we have $|V^*| = |V^* \cap V''| + |V^* \setminus V''| = |V''| + |V''| - 2|S_1| \geq 2|V''| - 2|M_1|$. This implies that $|S| \geq |V''| - |M_1|$ which concludes the proof. □