Expanding the expressive power of Monadic Second-Order logic on restricted graph classes

Robert Ganian[1] and Jan Obdržálek[2]

[1] Vienna University of Technology, Austria[⋆⋆⋆]
rganian@gmail.com
[2] Faculty of Informatics, Masaryk University, Brno, Czech Republic[†]
obdrzalek@fi.muni.cz

Abstract. We combine integer linear programming and recent advances in Monadic Second-Order model checking to obtain two new algorithmic meta-theorems for graphs of bounded vertex-cover. The first shows that cardMSO$_1$, an extension of the well-known Monadic Second-Order logic by the addition of cardinality constraints, can be solved in FPT time parameterized by vertex cover. The second meta-theorem shows that the MSO partitioning problems introduced by Rao can also be solved in FPT time with the same parameter.
The significance of our contribution stems from the fact that these formalisms can describe problems which are W[1]-hard and even NP-hard on graphs of bounded tree-width. Additionally, our algorithms have only an elementary dependence on the parameter and formula. We also show that both results are easily extended from vertex cover to neighborhood diversity.

1 Introduction

It is a well-known result of Courcelle, Makowski and Rotics that MSO$_1$ (and LinEMSO$_1$) model checking is in FPT on graphs of bounded clique-width [4]. However, this leads to algorithms which are far from practical – the time complexity includes a tower of exponents, the height of which depends on the MSO$_1$ formula. Recently it has been shown that much faster model checking algorithms are possible if we consider more powerful parameters such as vertex cover [15] – with only an elementary dependence of the runtime on both the MSO$_1$ formula and parameter.

Vertex cover has been generally used to solve individual problems for which traditional width parameters fail to help (see e.g. [1,6,9,10]). Of course, none of these problems can be described by the standard MSO$_1$ or LinEMSO$_1$ formalism. This raises the following, crucial question: would it be possible to naturally extend the language of MSO$_1$ to include additional well-studied problems without sacrificing the positive algorithmic results on graphs of bounded vertex-cover?

⋆⋆⋆ Robert Ganian acknowledges support by ERC (COMPLEX REASON, 239962).
† Jan Obdržálek is supported by the research centre Institute for Theoretical Computer Science (ITI), project No. P202/12/G061.

We answer this question by introducing cardMSO$_1$ (Definition 2.3) as the extension of MSO$_1$ by linear cardinality constraints – linear inequalities on vertex set cardinalities and input-specified variables. The addition of linear inequalities significantly increases the descriptive power of the logic, and allows to capture interesting problems which are known to be hard on graphs of bounded tree-width. We refer to Section 4 for a discussion of the expressive power and applications of cardMSO$_1$, including a new result for the c-balanced partitioning problem (Theorem 4.1).

The first contribution of the article lies in providing an FPT-time model checking algorithm for cardMSO$_1$ on graphs of bounded vertex cover. This extends the results on MSO$_1$ model checking obtained by Lampis in [15], which introduce an elementary-time FPT MSO$_1$ model checking algorithm parameterized by vertex cover. However, the approach used there cannot be straightforwardly applied to formulas with linear inequalities (cf. Section 3 for further discussion).

Theorem 1.1. *There exists an algorithm which, given a graph G with vertex cover of size k and a* cardMSO$_1$ *formula φ with q variables, decides if $G \models \varphi$ in time $2^{2^{O(k+q)}+|\varphi|} + 2^k|V(G)|$.*

The core of our algorithm rests on a combination of recent advances in MSO$_1$ model checking and the use of Integer Linear Programming (ILP). While using ILP to solve individual difficult graph problems is not new [9], the goal here was to obtain new graph-algorithmic meta-theorems for frameworks containing a wide range of difficult problems. The result also generalizes to the neighborhood diversity parameter introduced in [15] and to MSO$_2$ (as discussed in Section 6).

In the second part of the article, we turn our attention to a different, already studied extension of MSO$_1$: the MSO partitioning framework of Rao [19]. MSO partitioning asks whether a graph may be partitioned into an arbitrary number of sets so that each set satisfies a fixed MSO$_1$ formula, and has been shown to be solvable in XP time on graphs of bounded clique-width. Although MSO partitioning is fundamentally different from cardMSO$_1$ and both formalisms expand the power of MSO$_1$ in different directions, we show that a combination of MSO$_1$ model checking and ILP may also be used to provide an efficient FPT model-checking algorithm for MSO$_1$ partitioning parameterized by vertex-cover or neighborhood diversity.

Theorem 1.2. *There exists an algorithm which, given a graph G with vertex cover of size k and a MSO partitioning instance (φ, r) with q variables, decides if $G \models (\varphi, r)$ in time $2^{2^{O(q2^k)}} \cdot |(\varphi, r)| + 2^k|V(G)|$.*

2 Preliminaries and Definitions

2.1 Vertex cover and types

In the following text all graphs are simple and without loops. For a graph G we use $V(G)$ and $E(G)$ to denote the sets of its vertices and edges, and use $N(v)$ to denote the set of neighbors of a vertex $v \in V(G)$.

The graph parameter we are primarily interested in is vertex cover. A key notion related to graphs of bounded vertex cover is the notion of a vertex type.

Definition 2.1 ([15]). *Let G be a graph. Two vertices $u, v \in V$ are of the same* type *T if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. We use $\mathcal{T}_G$ to denote the set of all types of G (or just $\mathcal{T}$ if G is clear from the context).*

Since each type is associated with its vertices, we also use T to denote the set of vertices of type T. Note that then $\mathcal{T}_G$ forms a partition of the set $V(G)$.

For the sake of simplicity, we adopt the convention that, on a graph with a fixed vertex cover X, we additionally separate each cover vertex into its own type. Then it is easy to see that each type is an independent set, and a graph with vertex cover of size k has at most $2^k + k$ types.

It is often useful to divide vertices of the same type further into subtypes. The subtypes are usually identified by a system of sets, and all subtypes of a given type form a partition of that type:

Definition 2.2. *Let G be a graph and $\mathcal{U} \subseteq 2^{V(G)}$ a set of subsets of $V(G)$. Then two vertices $u, v \in V(G)$* are of the same subtype *(w.r.t. $\mathcal{U}$) if $u, v \in T$ for some $T \in \mathcal{T}_G$ and $\forall U \in \mathcal{U}. u \in U \iff v \in U$. We denote by $\mathcal{S}_T^{\mathcal{U}}$ the set of all subtypes of a type $T \in \mathcal{T}_G$, and also define the set of all subtypes of (w.r.t. $\mathcal{U}$) as $\mathcal{S}_G^{\mathcal{U}}$. (If G and $\mathcal{U}$ are clear form the context, we may write $\mathcal{S}$ instead of $\mathcal{S}_G^{\mathcal{U}}$)*

Finally, notice that $|\mathcal{S}_G^{\mathcal{U}}| \leq 2^{|\mathcal{U}|}|\mathcal{T}_G|$.

2.2 MSO_1 and its cardinality extensions

Monadic Second Order logic (MSO_1) is a well established logic of graphs. It is the extension of first order logic with quantification over vertices and sets of vertices. MSO_1 in its basic form can only be used to describe decision problems. To solve optimization problems we may use $\mathrm{LinEMSO}_1$ [4], which is capable of finding maximum- and minimum-cardinality sets satisfying a certain MSO_1 formula. This is useful for providing simple descriptions of well-known optimization problems such as Minimum Dominating Set (adj is the adjacency relation):

$$\mathrm{Min}(X) \colon \forall a \exists b \in X \colon (adj(a, b) \vee a = b)$$

The crucial point is that $\mathrm{LinEMSO}_1$ only allows the optimization of set cardinalities over all assignments satisfying a MSO_1 formula. It is not possible to use $\mathrm{LinEMSO}_1$ to place restrictions on cardinalities of sets considered in the formula. In fact, such restrictions may be used to describe problems which are W[1]-hard on graphs of bounded tree-width, whereas all $\mathrm{LinEMSO}_1$-definable problems may be solved in FPT time even on graphs of bounded clique-width [4].

In this paper we define $\mathrm{cardMSO}_1$, an extension of MSO_1 which allows restrictions on set cardinalities.

Definition 2.3 ($\mathrm{cardMSO}_1$). *The language of $\mathrm{cardMSO}_1$ logic consists of expressions built from the following elements:*

– *variables $x, y \ldots$ for vertices, and $X, Y \ldots$ for sets of vertices*
– *the predicates $x \in X$ and $\mathrm{adj}(x, y)$ with the standard meaning*
– *equality for variables, quantifiers $\forall, \exists$ and the standard Boolean connectives*
– *tt and ff as the standard valuation constants representing true and false*
– *the expressions $[\rho_1 \leq \rho_2]$, where the syntax of the ρ expressions is defined as $\rho ::= n \mid |X| \mid \rho + \rho$, where $n \in \mathbb{Z}$ ranges over integer constants and X over (vertex) set variables.*

We call expressions of the form $[\rho_1 \leq \rho_2]$ linear (cardinality) constraints*, and write $[\rho_1 = \rho_2]$ as a shorthand for $[\rho_1 \leq \rho_2] \wedge [\rho_2 \leq \rho_1]$, and $[\rho_1 < \rho_2]$ for $[\rho_1 \leq \rho_2] \wedge \neg[\rho_2 \leq \rho_1]$. A* formula φ *of* $\mathrm{cardMSO}_1$ *is an expression of the form $\varphi = \exists Z_1 \ldots \exists Z_m . \overline{\varphi}$ such that $\overline{\varphi}$ is a* MSO_1 *formula and $Z_1, \ldots, Z_m$ are the only variables which appear in the linear constraints.*

To give the semantics *of* $\mathrm{cardMSO}_1$ *it is enough to define the semantics of cardinality constraints, the rest follows the standard* MSO_1 *semantics. Let $\mathcal{V} : \mathcal{X} \to \mathbb{Z}$ be a valuation of set variables. Then the truth value of $[\rho_1 \leq \rho_2]$ is obtained be replacing each occurrence of $|X|$ with the cardinality of $\mathcal{V}(X)$ and evaluating the expression as standard integer inequality.*

To give an example, the following $\mathrm{cardMSO}_1$ formula is true if, and only if, a graph is bipartite and both parts have the same cardinality:

$$\exists X_1 \exists X_2 . (\forall v \in V . (v \in X_1 \iff \neg v \in X_2)) \wedge [|X_1| = |X_2|] \wedge$$
$$(\forall u \in V . (\mathit{adj}(u, v) \implies ((u \in X_1 \wedge v \in X_2) \vee (u \in X_2 \wedge v \in X_1))))$$

For a $\mathrm{cardMSO}_1$ formula $\varphi = \exists Z_1 \ldots \exists Z_m . \overline{\varphi}$ we call $\exists Z_1 \ldots \exists Z_m$ the *prefix* of φ, and the variables Z_i *prefix variables*. We also put $\mathcal{Z}(\varphi) = \{Z_1, \ldots, Z_m\}$, and often write just $\mathcal{Z}$ if φ is clear from the context. Note that, since all prefix variables are existentially quantified set variables, checking whether $G \models \varphi$ (for some graph G) is equivalent to finding a variable assignment $\chi : \mathcal{Z} \to 2^{V(G)}$ such that $G \models_\chi \overline{\varphi}$. We call such χ the *prefix assignment* (for G and φ). Note that the sets $\chi(Z_i)$ can be used to determine subtypes, and therefore we often write $\mathcal{S}_G^\chi$ with the obvious meaning.

2.3 ILP Programming

Integer Linear Programming (ILP) is a well-known framework for formulating problems, and will be used extensively in our approach. We provide only a brief overview of the framework:

Definition 2.4 (p-Variable ILP Feasibility (p-ILP)). *Given matrices $A \in \mathbb{Z}^{m \times p}$ and $b \in \mathbb{Z}^{m \times 1}$, the* p-Variable ILP Feasibility (p-ILP) problem *is whether there exists a vector $x \in \mathbb{Z}^{p \times 1}$ such that $A \cdot x \leq b$. The number of variables p is the parameter.*

Lenstra [16] showed that p-ILP, together with its optimization variant p-OPT-ILP, can be solved in FPT time. His running time was subsequently improved by Kannan [14] and Frank and Tardos [11].

Theorem 2.5 ([16,14,11,9]). *p-ILP and p-OPT-ILP can be solved using $O(p^{2.5p+o(p)} \cdot L)$ arithmetic operations in space polynomial in L, L being the number of bits in the input.*

3 cardMSO$_1$ Model Checking

The main purpose of this section is to give a proof of Theorem 1.1. The proof builds upon the following result of Lampis:

Lemma 3.1 ([15]). *Let φ be an* MSO$_1$ *formula with q_S set variables and q_v vertex variables. Let G_1 be a graph, $v \in V(G_1)$ a vertex of type T such that $|T| > 2^{q_S} \cdot q_v$, and G_2 a graph obtained from G_1 by deleting v. Then $G_1 \models \varphi$ iff $G_2 \models \varphi$.*

In other words, a formula φ of MSO$_1$ cannot distinguish between two graphs G_1 and G_2 which differ only in the cardinalities of some types, as long as the cardinalities in both graphs are at least $2^{q_S} \cdot q_v$.

This gives us an efficient algorithm for model checking MSO$_1$ on graphs of bounded vertex cover: We first "shrink" the sizes of types to $2^{q_S} \cdot q_v$ and then recursively evaluate the formula, at each quantifier trying all possible choices for each set and vertex variable[1].

Theorem 3.2 ([15]). *There exists an algorithm which, for a* MSO$_1$ *sentence φ with q variables and a graph G with n vertices and vertex cover of size at most k, decides $G \models \varphi$ in time $2^{2^{O(k+q)}} + O(2^k n)$.*

However, a straightforward adaptation of the approach sketched above does not work with linear constraints. To see this, simply consider e.g. the formula $\exists Z_1 \exists Z_2.[|Z_1| = |Z_2| + 1]$. Changing the cardinality of Z_1 by even a single vertex can alter whether the linear constraint is evaluated as true or false, even if $|Z_1 \cap T|$ is large for some type T. On the other hand, observe that the truth value of a linear inequality $[\rho_1 \leq \rho_2]$ depends only on the prefix variables, not on the rest of the formula. With this in mind, we continue by sketching the general strategy for proving Theorem 1.1:

Given a graph G and a formula φ we begin by creating the graph G_φ from G by reducing the size of each type to $2^{q_S} \cdot q_v$. Since this construction can impact the possible values of linear constraints in φ, we replace each linear constraint with either *tt* or *ff*, effectively claiming which linear constraints we expect to be satisfied in G (for some assignment to prefix variables). We try all 2^l possible truth valuations of linear constraints.

For each MSO$_1$ formula ψ obtained from φ by fixing some truth valuation of linear constraints we now check whether $G_\varphi \models \psi$, generating all prefix assignments χ for which $G_\varphi \models_\chi \overline{\psi}$. The remaining step is to check whether some prefix assignment (in G_φ) can be extended to a prefix assignment in G in such

[1] Note that both Lemma 3.1 and Theorem 3.2 implicitly utilize the symmetry between vertices of the same type.

a way that ψ would still hold in G and all linear cardinality constraints would evaluate to their guessed values. This check is performed by the construction of an p-ILP formulation which is feasible if, and only if, there is such an extension.

We will now formalize the proof we have just sketched. First, we need a few definitions. We start by formalizing the process of "shrinking" (some types of) a graph.

Definition 3.3. *Given a graph G and a* cardMSO$_1$ *formula $\varphi = \exists Z_1 \ldots \exists Z_m . \overline{\varphi}$ with q_v vertex and $m + q_S$ set variables, we define the* reduced graph G_φ *to be the graph obtained from G by the following prescription:*

1. *For each type $T \in \mathcal{T}_G$ s.t. $|T| > 2^{q_S+m} q_v$ we delete the "extra" vertices of type T so that exactly $2^{q_S+m} q_v$ vertices of this type remain, and*
2. *we take the subgraph induced by the remaining vertices.*

Note that vertices of a type with cardinality at most $2^{q_S+m} q_v$ are never deleted in the process of "shrinking" G, and $|V(G_\varphi)| \leq |\mathcal{T}_{G_\varphi}| \cdot 2^{q_S+m} q_v$. Next we formalize the process of fixing the truth values of linear cardinality constraints.

Definition 3.4. *Let $l(\varphi) = \{l_1, \ldots, l_k\}$ be the list of all linear cardinality constraints in the formula φ. Let $\alpha : l(\varphi) \to \{tt, ff\}$, called the* pre-evaluation function*, be an assignment of truth values to all linear constraints. Then by $\alpha(\varphi)$ we denote the formula obtained from φ by replacing each linear constraint l_i by $\alpha(l_i)$, and call $\alpha(\varphi)$ the* pre-evaluation of φ*. Note that $\alpha(\varphi)$ is a* MSO$_1$ *formula.*

As we mentioned earlier, the truth value for each linear cardinality constraint depends only on the values of prefix variables. Therefore all linear constraints can be evaluated once we have fixed a prefix assignment. We say that a prefix assignment χ, of a cardMSO$_1$ formula φ, *complies with* a pre-evaluation α if each linear constraint $l \in l(\varphi)$ evaluates to true (under χ) if, and only if, $\alpha(l) = tt$.

We also need a notion of extending a prefix assignment for G_φ to G. In the following definition we use the implicit matching between the subtypes S of G and the subtypes S_φ of its subgraph G_φ.

Definition 3.5. *Given a graph G and a* cardMSO$_1$ *formula $\varphi = \exists Z_1 \ldots \exists Z_m . \overline{\varphi}$ with q_v vertex and q_S set variables in $\overline{\varphi}$, we say that a prefix assignment χ for G extends a prefix assignments χ_φ for G_φ if for all $S \in \mathcal{S}^\chi_G$:*

1. *$S = S_\varphi$ if $|S_\varphi| \leq 2^{q_S} q_v$*
2. *$S \supseteq S_\varphi$ if $|S_\varphi| > 2^{q_S} q_v$*

Finally we will need the following statement, which directly follows from the proof of Lemma 3.1 [15]:

Lemma 3.6. *Let $\varphi = \exists Z_1 \ldots \exists Z_m . \overline{\varphi}$ be an* MSO$_1$ *formula, with q_S set variables in $\overline{\varphi}$ and q_v vertex variables, and let $\chi_1 : \mathcal{Z} \to 2^{V(G_1)}$ be a prefix assignment in G_1. Let $v \in V(G_1)$ be a vertex of subtype $S \in \mathcal{S}^\chi_{G_1}$ such that $|S| > 2^{q_s} q_v$, and G_2 a graph obtained from G_1 by deleting v. Then $G_1 \models_{\chi_1} \varphi$ iff $G_2 \models_{\chi_2} \varphi$, where χ_2 is the prefix assignment induced by χ_1 on G_2.*

For the remainder of this section let us fix a cardMSO$_1$ formula $\varphi = \exists Z_1 \ldots \exists Z_m . \overline{\varphi}$ with q_v vertex variables, q_S set variables in $\overline{\varphi}$ and with linear cardinality constraints $l(\varphi) = \{l_1, \ldots, l_k\}$. We are now ready to state the main lemma:

Lemma 3.7. *Let G be a graph, φ be a* cardMSO$_1$ *formula, χ_φ be a prefix assignment for G_φ, and α a pre-evaluation such that $G_\varphi \models_{\chi_\varphi} \alpha(\overline{\varphi})$. Then we can, in time $O(|\mathcal{T}_G| \cdot 2^m |l(\varphi)|)$, construct a p-ILP formulation which is feasible iff χ_φ can be extended to a prefix assignment χ for G such that (a) χ complies with α, and (b) $G \models_\chi \overline{\varphi}$. Moreover, the formulation has $|\mathcal{T}_G| \cdot 2^m$ variables.*

Proof. We start by showing the construction of the p-ILP formulation. The set of variables is created as follows: For each subtype $S \in \mathcal{S}_{G_\varphi}^{\chi_\varphi}$ we introduce a variable x_S which will represent the cardinality of S in G. There are three groups of constraints:

1. We need to make sure that, for each type $T \in \mathcal{T}_G$, the cardinalities of all subtypes of T sum up to the cardinality of a type T. This is easily achieved by including a constraint $\sum_{S \subseteq T} x_S = |T|$ for each type T (note that here $|T|$ is a constant).

2. We need to guarantee that χ extends χ_φ. Therefore we include $x_S = |S_\varphi|$ for each subtype with $|S_\varphi| \leq 2^{q_S} q_v$, and $x_S > |S_\varphi|$ if $|S_\varphi| > 2^{q_S} q_v$.

3. We need to check that χ complies with α, i.e. that each linear constraint l is either true or false based on the value of $\alpha(l)$. For each constraint l we first replace each occurrence of $|Z_i|$ with the sum of cardinalities of all subtypes which are contained in Z_i, i.e. by $\sum_{S_\varphi \subseteq Z_i} x_S$. Then if $\alpha(l) = tt$, we simply insert the modified constraint into the formulation. Otherwise we first reverse the inequality (e.g. $>$ instead of $\leq$), and then also insert it.

To prove the forward implication, let us assume that the p-ILP formulation is feasible. To define χ we start with $\chi = \chi_\varphi$. Then for each subtype $S \in \mathcal{S}_G$ if $x_S > |S_\varphi|$ we add $x_S - |S_\varphi|$ unassigned vertices of type T, where T is the supertype of S. This is always possible thanks to constraints in 1. and 2. The constraints in 3. guarantee that χ complies with α. Finally $G \models_\chi \overline{\varphi}$ by Lemma 3.6.

For the reverse implication let $S \in \mathcal{S}_G$ be the subtype identified by the set $\mathcal{Y} \subset \mathcal{Z}$. Then we put $x_S = |\{v \in V(G) | \forall Z \in \mathcal{Z} . v \in \chi(Z) \iff Z \in \mathcal{Y}\}|$, and the p-ILP formulation is satisfiable by our construction. Finally, it is easy to verify that the size of this p-ILP formulation is at most $O(|\mathcal{T}_G| \cdot 2^{q_S} |l(\varphi)|)$. ∎

Proof of Theorem 1.1. We start by constructing G_φ from G, which may be done by finding a vertex cover in time $O(2^k \cdot n)$, dividing vertices into at most $2^k + k$ types (in linear time once we have a vertex cover) and keeping at most $2^{q_S + m} q_v$ vertices in each type.

Now for each pre-evaluation $\alpha : l(\varphi) \to \{tt, ff\}$ we do the following: We run the trivial recursive MSO$_1$ model checking algorithm on G_φ, by trying all possible assignments of vertices of G_φ to set and vertex variables. Each time we find a satisfying assignment, we remember the values of the prefix variables $\mathcal{Z}$, and proceed to finding the next satisfying assignment. Since the prefix variables

of φ (and $\alpha(\varphi)$) are existentially quantified, their value is fixed before $\alpha(\overline{\varphi})$ starts being evaluated and therefore is the same at any point of evaluating $\alpha(\overline{\varphi})$. At the end of this stage we end up with at most $(2^{|V(G_\varphi)|})^m$ different satisfying prefix assignments of $Z_1, \ldots, Z_m$ for each pre-evaluation α.

We now need to check whether some combination of a pre-evaluation α and its satisfying prefix assignment χ_φ from the previous step can be extended to a satisfying assignment for $\overline{\varphi}$ and G. This can be done by Lemma 3.7.

To prove correctness, assume that there exists a satisfying assignment χ for G. We create G'_φ by, for each $T \in \mathcal{T}_G$ such that $|T| > 2^{q_S+m}q_v$, inductively deleting vertices from subtypes $S \subseteq T$ such that $|S| > 2^{q_s}q_v$, until $|T| = 2^{q_S+m}q_v$ for every T. Observe that G'_φ is isomorphic to G_φ and that there is a satisfying assignment χ' induced by χ on G'_φ. Then applying the isomorphism to χ' creates a satisfying assignment χ_2 on G_φ, and Lemma 3.7 ensures that our p-ILP formulation is feasible for χ_2.

To compute the time complexity of this algorithm, note that we first need time $O(2^k \cdot n)$ to compute G_φ. Then for each of the $2^{|l|}$ pre-evaluations we compute all the satisfying prefix assignments in time $2^{2^{O(k+q_S+m)}q_v}$ by Theorem 3.2. For each of the at most $(2^{|V(G_\varphi)|})^m = (2^{(2^k+k)\cdot 2^{q_S+m}q_v})^m$ satisfying prefix assignments for G_φ, we check whether it can be extended to an assignment for G, which can be done in time at most $2^{2^{O(k+q_S+m)}}$ by applying Theorem 2.5 on the p-ILP formulation constructed by Lemma 3.7. We therefore need time $O(2^k \cdot n) + 2^m \cdot (2^{2^{O(k+q_S+m)}q_v+|l|} + (2^{(2^k+k)\cdot 2^{q_S+m}q_v})^m \cdot 2^{2^{O(k+q_S+m)}})$, and the bound follows. ∎

Remark: The space complexity of the algorithm presented above may be improved by successively applying Lemma 3.7 to each iteratively computed satisfying prefix assignment (for each pre-evaluation).

Before moving on to the next section, we show how these results can be extended towards well-structured dense graphs. It is easy to verify that the only reference to an actual vertex cover of our graph is in Theorem 3.2 – all other proofs rely purely on bounding the number of types. In [15] Lampis also considered a new parameter called *neighborhood diversity*, which is the number of different types of a graph. I.e. graph G has neighborhood diversity k iff $|\mathcal{T}_G| = k$. Since there exist classes of graphs with unbounded vertex cover but bounded neighborhood diversity (for instance the class of complete graphs), parameterizing by neighborhood diversity may in some cases lead to better results than using vertex cover.

Corollary 3.8. *There exists an algorithm which, given a graph G with neighborhood diversity k and a* cardMSO_1 *formula φ with q variables, decides if $G \models \varphi$ in time $2^{k2^{O(q)}+|\varphi|} + k \cdot poly(|V(G)|)$.*

Proof. The proof is nearly identical to the proof of Theorem 1.1. The only change is that we begin by computing the neighborhood diversity and the associated partition into types (which may be done in polynomial time, cf. Theorem

5 in [15]), and we of course use the fact that the number of types is now at most k instead of $2^k + k$. ∎

4 Applications

4.1 Equitable problems

Perhaps the most natural class of problems which may be captured by cardMSO$_1$ but not by MSO$_1$ (or even MSO$_2$) are equitable problems. Equitable problems generally ask for a partitioning of the graph into a (usually fixed) number of specific sets of equal (± 1) cardinality.
Equitable c-coloring [18] is probably the most extensively studied example of an equitable problem. It asks for a partitioning of a graph into c equitable independent sets and has applications in scheduling, garbage collection, load balancing and other fields (see e.g. [5,3]). While even equitable 3-coloring is W[1]-hard on graphs of bounded tree-width [8], equitable c-coloring may easily be expressed in cardMSO$_1$:

$$\exists A, B, C : partition(A, B, C) \wedge \forall x, y : ((x, y \in A \vee x, y \in B \vee x, y \in C) \implies \neg adj(x, y)) \\ \wedge equi(A, B) \wedge equi(A, C) \wedge equi(B, C), \text{ where}$$

- $partition(A, B, C) = \big(\forall x : (x \in A \vee \neg x \in B \vee \neg x \in C) \wedge (\neg x \in A \vee x \in B \vee \neg x \in C) \wedge (\neg x \in A \vee \neg x \in B \vee x \in C)\big)$.
- $equi(T, U) = (\big[|T| = |U| + 1\big] \vee \big[|T| + 1 = |U|\big] \vee \big[|T| = |U|\big])$.

Equitable connected c-partition [6] is another studied equitable problem which is known to be W[1]-hard even on graphs of bounded path-width but which admits a simple description in cardMSO$_1$:

$$\exists A, B, C : partition(A, B, C) \wedge conn(A) \wedge conn(B) \wedge conn(C) \\ \wedge equi(A, B) \wedge equi(A, C) \wedge equi(B, C), \text{ where}$$

- $conn(U) = \big(\forall T : (\forall x : x \in T \implies x \in U) \implies (T = U \vee (\neg \exists a : a \in T) \vee \exists a, b : a \in U \wedge \neg a \in T \wedge b \in T \wedge adj(a, b)\big)$.

4.2 Solution size as input

cardMSO$_1$ allows us to restrict the set cardinalities by constants given as part of the input. For instance, the formula below expresses the existence of an Independent Dominating Set of cardinality k:

$$\exists X : (\forall a, b \in X. \neg adj(a, b)) \wedge \\ \wedge (\forall b \in V. b \in X \vee (\exists a \in X.\, adj(a, b))) \wedge [|X| = k]$$

Notice that there is an equivalent MSO$_1$ formula for any fixed k. However, the number of variables in the MSO$_1$ formula would depend on k, which would negatively impact on the runtime of model checking. On the other hand, using an input-specified variable only requires us to change a constant in the p-ILP formulation, with no impact on runtime.

4.3 c-balanced partitioning

Finally, we show an example of how our approach can be used to obtain new results even for optimization problems, which are (by definition) not expressible by cardMSO$_1$. While the presented algorithm does not rely directly on Theorem 1.1, it is based on the same fundamental ideas.

The problem we focus on is c-balanced partitioning, which asks for a partition of the graph into c equitable sets such that the number of edges between different sets is minimized. The problem was first introduced in [17], has applications in parallel computing, electronic circuit design and sparse linear solvers and has been studied extensively (see e.g. [7,2]). The problem is notoriously hard to approximate, and while an exact XP algorithm exists for the c-balanced partitioning of trees parameterized by c [7,17], no parameterized algorithm is known for graphs of bounded tree-width.

Theorem 4.1. *There exists an algorithm which, given a graph G with vertex cover of size k and a constant c, solves c-balanced partitioning in time $2^{2^{O(k+c)}} + 2^k|V(G)|$.*

Proof. We begin by applying the machinery of Theorem 1.1 to the cardMSO$_1$ formula φ for equitable c-partitioning φ:

$$\exists A, B, C : partition(A, B, C) \wedge equi(A, B) \wedge equi(A, C) \wedge equi(B, C)$$

Recall that this means trying all possible assignments of the c set variables in G_φ and testing whether each assignment can be extended to G in a manner satisfying φ. Unlike in Theorem 1.1 though, we need to tweak the p-ILP formulations to not only check the existence of an extension χ for our pre-evaluation α, but also to find the χ which minimizes the size of the cut between vertex sets.

To do so, we add one variable β into the formulation and use a p-OPT-ILP formulation minimizing β. We also add a single equality into the formulation to make β equal to the size of the cut between the c vertex sets. While it is not possible to count the edges directly, the fact that we always have a fixed satisfying prefix assignment in G_φ allows us to calculate β as:

$$\beta = const_0 + \textstyle\sum_{S \in U} const_S x_S, \text{ where}$$

- $const_0$ is the number of edges between all pairs of cover vertices with different types (this is obtained from the prefix assignment in G_φ),
- U is the set of subtypes which do not contain cover vertices (recall that each cover vertex has its own subtype),
- x_S is the ILP variable for the cardinality of subtype S (cf. Lemma 3.7),
- For each subtype S, $const_S$ is the number of adjacent vertices in the cover assigned to a different vertex set than S. The values of $const_S$ depend only on the subtype S and the chosen prefix assignment χ_φ in G_φ.

For each satisfying prefix assignment χ_φ in G_φ, the p-OPT-ILP formulation will not only check that this may be extended to an assignment χ in G, but also

find the assignment in G which minimizes β. All that is left is to store the best computed β for each satisfying prefix assignment and find the satisfying prefix assignment with minimum β after the algorithm from Theorem 1.1 finishes.

For correctness, assume that there exists a solution which is smaller than the minimal β found by the algorithm. Such a solution would correspond to an assignment of φ in G, which may be reduced to a prefix assignment χ of a pre-evaluation $\alpha(\varphi)$ in G_φ. If we construct the p-ILP formulation for χ and $\alpha(\varphi)$, then the obtained β would equal the size of the cut. However, our algorithm computes the β for all pre-evaluations and satisfying prefix assignments in G_φ, so this gives a contradiction. ∎

5 MSO Partitioning

The MSO (or MSO_1) partitioning framework was introduced by Rao in [19] and allows the description of many problems which cannot be formulated in MSO, such as Chromatic number, Domatic number, Partitioning into Cliques etc. While a few of these problems (e.g. Chromatic number) may be solved on graphs of bounded tree-width in FPT time by using additional structural properties of tree-width, MSO partitioning problems in general are W[1]-hard on such graphs.

Definition 5.1 (MSO partitioning). *Given a MSO formula φ, a graph G and an integer r, can $V(G)$ be partitioned into sets $X_1, X_2, \dots X_r$ such that $\forall i \in \{1, 2, \dots, r\} : X_i \models \varphi$?*

Similarly to Section 3, we will show that a combination of ILP and MSO model checking allows us to design efficient FPT algorithms for MSO partitioning problems on graphs of bounded vertex cover. However, here the total number of sets is specified on the input and so the number of subtypes is not fixed, which prevents us from capturing the cardinality of subtypes by ILP variables. Instead we use the notion of *shape*:

Definition 5.2. *Given a graph G and a MSO_1 formula φ with q_S, q_v set and vertex variables respectively, two sets $A, B \subseteq V(G)$ have the same* shape *iff for each type T it holds that either $|A \cap T| = |B \cap T|$ or both $|A \cap T|, |B \cap T| > 2^{q_S} q_v$.*

Let A be any set of shape s. We define $|s \cap T|$, for any type T, as:

$$|s \cap T| = \begin{cases} |A \cap T| & \text{if } |A \cap T| \leq 2^{q_S} q_v \\ \top & \text{otherwise} \end{cases}$$

Since φ is a MSO_1 formula, from Lemma 3.1 we immediately get:

(5.3) For any two sets $A, B \subseteq V(G)$ of the same shape, it holds that $A \models \varphi$ iff $B \models \varphi$, and

(5.4) Given a MSO formula with q variables, a graph G with vertex cover of size k has at most $(2^{q_S} q_v)^{2^k+k}$ distinct shapes.

With these in hand, we may proceed to:

Proof of Theorem 1.2. First, we consider all at most $(2^{q_S} q_v)^{2^k+k}$ shapes of a set X. For each such shape s, we decide whether a set X_s of shape s satisfies φ by Theorem 3.2. We then create an ILP formulation with one variable x_s for each shape s satisfying φ. The purpose of x_s is to capture the number of sets X_s of shape s in the partitioning of G.

Two conditions need to hold for the number of sets of various shapes. First, the total number of sets needs to be r. This is trivial to model in our formulation by simply adding the constraint that the sum of all x_s equals r.

Second, it must be possible to map each vertex in G to one and only one set X (to ensure that the sets form a partition). Notice that if a partition were to only contain shapes with at most $2^{q_S} q_v$ vertices in each T, then the cardinality of $s \cap T$ would be fixed and so the following set of constraints for each $T \in \mathcal{T}$ would suffice:

$\sum_{\forall s} x_s \cdot |s \cap T| = |T|$

However, in general the partition will also contain shapes with more than $2^{q_S} q_v$ vertices in T, and in this case we do not have access to the exact cardinality of their intersection with T. To this end, for each $T \in \mathcal{T}$ we add the following two sets of constraints:

a) $\sum_{\forall s:|s\cap T|\leq 2^{q_S} q_v} x_s \cdot |s \cap T| + \sum_{\forall s:|s\cap T|=\top} x_s \cdot (2^{q_S} q_v) \leq |T|$
b) $\sum_{\forall s:|s\cap T|\leq 2^{q_S} q_v} x_s \cdot |s \cap T| + \sum_{\forall s:|s\cap T|=\top} x_s \cdot |T| \geq |T|$

Here a) ensures that a partitioning of G into $\sum_{\forall s} x_s$ sets of shape s can "fit" into each T and b) ensures that there are no vertices which cannot be mapped to any set. Notice that if the partition contains any shape s which intersects with T in over $2^{q_S} q_v$ vertices then b) is automatically satisfied, since all unmapped vertices in T can always be added to s without changing $X_s \models \varphi$.

If the p-ILP formulation specified above has a feasible solution, then we can construct a solution to (φ, r) on G by partitioning G as follows: For each shape s we create sets $X_{s,1} \dots X_{s,x_s}$. Then in each type T in G, we map $|T \cap s|$ yet-unmapped vertices to each set $X_{s,i}$. Constraints a) make sure this is possible. If there are any vertices left unmapped in T, then due to constraint b) there must exist some set X' such that $|X' \cap T| > 2^{q_S} q_v$. We map the remaining unmapped vertices in T to any such set X', resulting in a partition of G. Finally, the fact that each of our sets satisfies φ follows from our selection of shapes.

On the other hand, if a solution to (φ, r) on G exists, then surely each set in the partition has some shape and so it would be found by the p-ILP formulation. The total runtime is the sum of finding the vertex cover, the time of model-checking all the shapes and the runtime of p-ILP, i.e. $O(2^k|V(G)) + 2^{2^{O(k+q)}} \cdot q^{(2^k+k)} + q^{(2^k+k)\cdot q^{O(2^k+k)}}$. ∎

Theorem 1.2 straightforwardly extends to neighborhood diversity as well. Directly bounding the number of types by k results in a bound of $(2^{q_S} q_v)^k$ on the number of distinct shapes in Claim 5.4, and so we get:

Corollary 5.5. *There exists an algorithm which, given a graph G with neighborhood diversity at most k and a MSO partitioning instance (φ, r) with q variables, decides if $G \models (\varphi, r)$ in time $2^{2^{O(qk)}} \cdot |(\varphi, r)| + k|V(G)|$.*

6 Concluding Notes

The article provides two new meta-theorems for graphs of bounded vertex cover. Both considered formalisms can describe problems which are W[1]-hard on graphs of bounded clique-width and even tree-width. On the other hand, we provide FPT algorithms for both cardMSO$_1$ and MSO partitioning which have an elementary dependence on both the formula and parameter (as opposed to the results of Courcelle et al. for tree-width).

The obtained time complexities are actually fairly close to the lower bounds provided in [15] for MSO$_1$ model checking (already $2^{2^{o(k+q)}} \cdot poly(n)$ would violate ETH); this is surprising since the considered formalisms are significantly more powerful than MSO$_1$. Our methods may also be of independent interest, as they show how to use p-ILP as a powerful tool for solving general model checking problems.

Let us conclude with future work and possible extensions of our results. As correctly observed by Lampis in [15], any MSO$_2$ formula can be expressed by MSO$_1$ on graphs of bounded vertex cover. This means that an (appropriately defined) cardMSO$_2$ or MSO$_2$ partitioning formula could be translated to an equivalent cardMSO$_1$ or MSO partitioning formula on graphs of bounded vertex cover. However, the details of these formalisms would need to be laid out in future work.

Another direction would be to extend the results of Theorems 1.1 and 1.2 to more general parameters, such as twin-cover [12] or shrub-depth [13]. Finally, it would be interesting to extend cardMSO$_1$ to capture more hard problems. Theorem 4.1 provides a good indication that the formalism could be adapted to also describe a number of optimization problems on graphs.

References

1. A. Adiga, R. Chitnis, and S. Saurabh. Parameterized algorithms for boxicity. In *ISAAC (1)*, pages 366–377, 2010.
2. K. Andreev and H. Räcke. Balanced graph partitioning. *Theory Comput. Syst.*, 39(6):929–939, 2006.
3. J. Bazewicz and E. Al. *Scheduling Computer and Manufacturing Processes.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 2001.
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. S. K. Das, I. Finocchi, and R. Petreschi. Conflict-free star-access in parallel memory systems. *J. Parallel Distrib. Comput.*, 66(11):1431–1441, Nov. 2006.
6. R. Enciso, M. R. Fellows, J. Guo, I. Kanj, F. Rosamond, and O. Suchý. What makes equitable connected partition easy. In *Parameterized and Exact Computation*, pages 122–133. Springer-Verlag, 2009.
7. A. E. Feldmann and L. Foschini. Balanced Partitions of Trees and Applications. In *STACS 2012*, volume 14 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 100–111, Dagstuhl, Germany, 2012. Schloss Dagstuhl.
8. M. Fellows, F. V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider, and C. Thomassen. On the complexity of some colorful problems parameterized by treewidth. In *COCOA'07*, pages 366–377. Springer-Verlag, 2007.
9. M. R. Fellows, D. Lokshtanov, N. Misra, F. A. Rosamond, and S. Saurabh. Graph layout problems parameterized by vertex cover. In *ISAAC (1)*, pages 294–305. Springer-Verlag, 2008.
10. J. Fiala, P. A. Golovach, and J. Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theor. Comput. Sci.*, 412(23):2513–2523, 2011.
11. A. Frank and É. Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.
12. R. Ganian. Twin-cover: Beyond vertex cover in parameterized algorithmics. In *IPEC*, number 7112 in LNCS, 2011.
13. R. Ganian, P. Hliněný, J. Nešetřil, J. Obdržálek, P. O. de Mendez, and R. Ramadurai. When trees grow low: Shrubs and fast MSO_1. In *MFCS'12*, volume 7464 of *LNCS*, pages 419–430. Springer, 2012.
14. R. Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12:415–440, 1987.
15. M. Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.
16. H. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.
17. R. M. MacGregor. *On partitioning a graph: a theoretical and empirical study.* PhD thesis, University of California, Berkeley, 1978.
18. W. Meyer. Equitable coloring. *American Mathematical Monthly*, 80:920–922, 1973.
19. M. Rao. MSOL partitioning problems on graphs of bounded treewidth and clique-width. *Theoret. Comput. Sci.*, 377:260–267, 2007.