Monadic Second Order Logic on Graphs with Local Cardinality Constraints

STEFAN SZEIDER
Vienna University of Technology, Austria

We introduce the class of MSO-LCC problems which are problems of the following form: Given a graph G and for each vertex v of G a set $\alpha(v)$ of non-negative integers. Is there a set S of vertices or edges of G such that (i) S satisfies a fixed property expressible in monadic second order logic, and (ii) for each vertex v of G the number of vertices/edges in S adjacent/incident with v belongs to the set $\alpha(v)$? We demonstrate that several hard combinatorial problems such as Lovász's General Factor Problem can be naturally formulated as MSO-LCC problems. Our main result is the polynomial-time tractability of MSO-LCC problems for graph of bounded treewidth. We obtain this result by means of a tree-automata approach. In way of contrast we show that a more general class of MSO-LCC problems where cardinality constraints are applied to second-order variables that are arbitrarily quantified does not admit polynomial-time tractability for graphs of bounded treewidth unless P = NP.

Categories and Subject Descriptors: F.4.1 [**Theory of Computation**]: Mathematical Logic—*Computational logic*; G.2.2 [**Mathematics of Computing**]: Graph Theory—*Graph algorithms*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: MSO model checking, cardinality constraint, treewidth, W[1]-hardness, NP-hardness

1. INTRODUCTION

Treewidth is a graph invariant that indicates the "tree-likeness" or "global connectivity" of a graph. It was introduced independently by several authors in different context and terminology (see, e.g., [Bodlaender 1998]). Many combinatorial graph problems that are hard in general become easy for graphs of small treewidth, for example 3-Colorability and Hamiltonicity can be decided in linear time for graphs of treewidth bounded by a constant k (however, with a running time containing a constant factor that is exponential in k). Such algorithms are usually obtained by means of dynamic programming applied via a bottom-up traversal of a decomposition tree associated with the given graph. Courcelle's famous theorem [Courcelle 1987; 1990] provides a unified framework for establishing linear-time

A preliminary and shortened version appeared in the proceedings of MFCS'08, The 33rd International Symposium on Mathematical Foundations of Computer Science.
Work supported in part by the European Research Council (ERC), Grant 239962.
Author's address: S. Szeider, Institute of Information Systems, KBS, Vienna University of Technology, Favoritenstraße 9-11, A-1040 Vienna, Austria.

algorithms for problems on graphs of bounded treewidth. It covers all decision problems that can be expressed in the formalism of Monadic Second Order (MSO) logic; 3-Colorability and Hamiltonicity are such problems. Arnborg, Lagergren, and Seese [1991] established an extension of Courcelle's Theorem that allows to solve MSO-expressible optimization problems in linear time for graphs of bounded treewidth. This covers problems that ask for a set S of vertices or edges of a given graph that satisfies an MSO-expressible property and minimizes a linear cost function associated with vertices and edges of the given graph. Many classical NP-hard problems such as Vertex Cover and Dominating Set can be naturally expressed in this formalism [Arnborg et al. 1991]. The mentioned theorems are "algorithmic meta-theorems" and provide a powerful toolkit for a quick complexity classification of problems; several variants and generalizations have been established in the literature [Grohe 2007].

In this paper we prove a new algorithmic meta-theorem. We consider problems of the following form: Given a graph G and for each vertex v of G a set $\alpha(v)$ of non-negative integers. The question is whether there exists a set S of vertices or edges of G that

(i) satisfies a fixed MSO-expressible property, and
(ii) for each vertex v of G, the number of vertices in S adjacent to v plus the number of edges in S incident with v belongs to the set $\alpha(v)$.

We call such a problem an *MSO problem for graphs with local cardinality constraints*, or *MSO-LCC problem*, for short.

For example, the General Factor problem introduced by Lovász [1970; 1972] is the MSO-LCC problem where the MSO property simply states that S is a set of edges. In Section 4 we discuss further examples of MSO-LCC problems.

Our main result is the following:

Theorem 1. *For every constant $k \geq 1$, every MSO-LCC problem can be solved in polynomial time for graphs of treewidth at most k.*

We establish Theorem 1 in two phases. Consider an MSO formula $\varphi(X)$ with free set variable X and a constant $k \geq 1$. Assume we are given a graph G with n vertices and treewidth k, and local cardinality constraints α. In the first phase we use Bodlaender's Theorem [Bodlaender 1996] and the tree automata approach of Arnborg et al. [1991] to construct in time $O(n)$ a certain auxiliary structure, a *solution tree*, which is a tree decomposition of G equipped with additional labels. The solution tree is a succinct representation of all solutions, that is, of all sets S of vertices and edges for which $\varphi(S)$ is true in G. We think that the concept of solution trees can be useful for other extensions or generalizations of the known MSO theorems. In the second phase we deal with the cardinality constraints by means of dynamic programming along the solution tree. This second phase can be carried out in time $n^{O(k)}$. By dynamic programming we can also produces a solution S (if it exists) within the same asymptotic running time.

In Section 5 we discuss extensions and limitations of Theorem 1:

Basic Extensions. First we show how our methods can be extended to deal with MSO-LCC problems defined by an MSO formula with more than one free set vari-

able, each with its own cardinality constraint. We further show how we can deal with integral vertex and edge weights (given in unary) as well as with a variant of cardinality constraints where a set $S \subseteq V(G) \cup E(G)$ only needs to satisfy $\alpha(v)$ for vertices v that belong to S.

Quantified MSO-LCC Problems. We consider the more general class of Q-MSO-LCC problems where cardinality constraints are applied to second-order variables that are arbitrarily quantified (not just existentially as for MSO-LCC problems). We show, however, that there exist Q-MSO-LCC problems that are already NP-hard for graphs of treewidth 2. Thus it is unlikely that Theorem 1 can be generalized in this direction.

Fixed-Parameter Intractability. The running time of the algorithm provided by the proof of Theorem 1 is polynomial since the treewidth bound k is considered constant; however, the order of the polynomial depends on k. One might wonder if this dependency is necessary. We explain that, subject to a complexity theoretic assumption, this dependency cannot be eliminated. Consequently, in contrast to the linear time bound of Courcelle's Theorem, it is unlikely that all MSO-LCC problems can be solved in linear time for graphs of bounded treewidth.

2. PRELIMINARIES

2.1 Graphs and Local Cardinality Constraints

All considered graphs are finite, simple, and undirected. Let G be a graph. $V(G)$ and $E(G)$ denote the vertex set and the edge set of G, respectively. The *order* of G is the number of its vertices. We denote an edge between vertices u and v by uv (or equivalently by vu), and we denote the degree of a vertex v in G by $d_G(v)$. Each vertex or edge x can be labeled with an element of some fixed finite set. The labels allow us to consider different "sorts" of vertices and edges. For a vertex $v \in V(G)$ we denote by $N_G(v)$ the set of neighbors of v in G (that is, the set of vertices $u \in V(G)$ such that $uv \in E(G)$) and by $I_G(v)$ the set of edges of G incident with v (that is, the set of edges $uv \in E(G)$). We denote by $G[S]$ the subgraph of a G induced by a set $S \subseteq V(G)$; that is, $V(G[S]) = S$ and $E(G[S]) = \{\, uv \in E(G) \mid u, v \in S \,\}$. For a set $S \subseteq V(G)$ we define $G - S = G[V(G) \setminus S]$.

A *graph with local cardinality constraints* is a pair (G, α) where G is a graph and α is a mapping that assigns to each vertex $v \in V(G)$ a set $\alpha(v)$ of non-negative integers. For a set $S \subseteq V(G) \cup E(G)$, a subgraph H of G, and a vertex $v \in V(H)$, we write

$$\mathsf{touch}(H, S, v) = |N_H(v) \cap S| + |I_H(v) \cap S|.$$

We say that a set $S \subseteq V(G) \cup E(G)$ *satisfies* α if $\mathsf{touch}(G, S, v) \in \alpha(v)$ holds for all $v \in V(G)$. Since $\mathsf{touch}(G, S, v) \leq |V(G)| - 1 + |E(G)|$, we can always assume that $\alpha(v) \subseteq \{0, \ldots, |V(G)| + |E(G)| - 1\}$.

2.2 Tree Decompositions

A *tree decomposition* of a graph G is a pair (T, χ) where T is a tree and χ is a mapping that assigns to each node $t \in V(T)$ a set $\chi(t) \subseteq V(G)$ such that the following conditions hold (we refer to the vertices of T as "nodes" to make the distinction between T and G clearer).

(1) $V(G) = \bigcup_{t \in V(T)} \chi(t)$ and $E(G) \subseteq \bigcup_{t \in V(T)} \{\, uv \mid u, v \in \chi(t) \,\}$.
(2) The sets $\chi(t_1) \setminus \chi(t)$ and $\chi(t_2) \setminus \chi(t)$ are disjoint for any three nodes $t, t_1, t_2 \in V(T)$ such that t lies on a path from t_1 to t_2 in T.

The *width* of (T, χ) is $\max_{t \in V(T)} |\chi(t)| - 1$. The *treewidth* $\mathsf{tw}(G)$ of G is the smallest integer k such that G has a tree decomposition of width k. For a node $t \in V(T)$ we write $\chi^*(t) = \chi(t) \cup E(G[\chi(t)])$.

It is easy to see that for a graph G and a set $X \subseteq V(G)$ always $\mathsf{tw}(G - X) \geq \mathsf{tw}(G) - |X|$ [Kloks 1994].

For fixed $k \geq 1$, one can decide for a given graph G of order n in time $O(n)$ whether $\mathsf{tw}(G) \leq k$, and if so, compute a tree decomposition of G of width $\leq k$ and $O(n)$ nodes (Bodlaender's Theorem [Bodlaender 1996]). However, without a fixed upper-bound k, it is NP-hard to determine the treewidth [Arnborg et al. 1987].

A graph of treewidth k and order n has at most $kn - k(k+1)/2$ edges [Rose 1974]. Hence for graphs of treewidth bounded by a constant, the number of edges is linear in the number of vertices. Thus linear running times for algorithms on graphs of bounded treewidth can be expressed as $O(n)$.

For our purposes it is convenient to consider tree decompositions in a certain normal form. A triple (T, r, χ) is a *nice tree decomposition* if (T, χ) is a tree decomposition with $\chi(t) \neq \emptyset$ for all $t \in V(T)$, and, considering T as a rooted tree with root r, for each node $t \in V(T)$ one of the following four conditions holds:

(1) t is a *leaf*.
(2) t has exactly one child t' and $|\chi(t) \setminus \chi(t')| = 1$ (we call t an *introduce node*).
(3) t has exactly one child t' and $|\chi(t') \setminus \chi(t)| = 1$ (we call t a *forget node*).
(4) t has exactly two children t', t'' and $\chi(t) = \chi(t') = \chi(t'')$ (we call t a *join node*).

We assume an arbitrary but fixed ordering of the two children of a join node, so that we can refer to its left and right child.

It is easy to see that one can transform efficiently a tree decomposition into a nice tree decomposition of the same width. In fact, if the width is bounded by a constant, then this transformation can be carried out in linear time [Kloks 1994]. Thus Bodlaender's Theorem can be strengthened to provide nice tree decompositions.

2.3 Monadic Second Order Logic

We consider *Monadic Second Order* (MSO) logic on (labeled) graphs in terms of their incidence structure whose universe contains vertices and edges; the incidence between vertices and edges is represented by a binary relation. We assume an infinite supply of *individual variables* $x, x_1, x_2, \ldots$ and of *set variables* $X, X_1, X_2, \ldots$ The *atomic formulas* are $E(x)$ ("x is an edge"), $V(x)$ ("x is a vertex"), $I(x, y)$ ("vertex x is incident with edge y"), $x = y$ (equality), $P_a(x)$ ("vertex or edge x has label a"), and $X(x)$ ("vertex or edge x is an element of set X").

MSO formulas are built up from atomic formulas using the usual Boolean connectives $(\neg, \wedge, \vee, \rightarrow, \leftrightarrow)$, quantification over individual variables ($\forall x$, $\exists x$), and quantification over set variables ($\forall X$, $\exists X$).

We write $\varphi(X)$ to denote an MSO formula with free set variable X. Let G be a graph, $S \subseteq V(G) \cup E(G)$, and let $\varphi(X)$ be an MSO formula. We write

$$G \models \varphi(S)$$

to indicate that φ is true for G if X is interpreted as S. For a graph with local cardinality constraints (G, α) we write

$$(G, \alpha) \models \varphi(S)$$

if $G \models \varphi(S)$ and S satisfies α.

2.4 Tree Automata

Let Σ be finite set. A Σ-*tree* T is a rooted binary tree whose nodes are labeled with elements of Σ. We assume that the children of a node are given in some arbitrary but fixed order, so that we can speak of a left child and a right child; we indicate the ordering by subscripts l and r or 1 and 2.

A (deterministic, bottom-up) *tree automaton* for Σ-trees is a tuple $M = (Q, \Sigma, \delta, q_0, Q_{\mathsf{acc}})$, consisting of a finite set Q of *states*, a *transition function* $\delta : Q \times Q \times \Sigma \to Q$, an *initial state* $q_0 \in Q$, and a set $Q_{\mathsf{acc}} \subseteq Q$ of *accepting states*.

Given a Σ-tree T, the automaton assigns deterministically states to the nodes of T, traversing the tree in a bottom-up ordering: leaves with label $a \in \Sigma$ are assigned state $\delta(q_0, q_0, a)$; if the two children of a node v with label a have been assigned states q_l and q_r, respectively, then v is assigned state $\delta(q_l, q_r, a)$. M *accepts* T if it assigns the root an accepting state $q \in Q_{\mathsf{acc}}$.

We define MSO logic on Σ-trees in terms of the relational structure whose universe consists of the nodes of the tree, and where two binary relations represent left and right descent, respectively. In particular, we use the atomic formulas $L(u, v_l)$ ("v_l is the left child of u"), $R(u, v_r)$ ("v_r is the right child of u"), $x = y$ (equality), $P_a(x)$ ("node x has label $a \in \Sigma$"), and $X(y)$ ("node y is an element of set X").

By a classical result of Thatcher and Wright [1968], one can, given a closed MSO formula φ for Σ-trees, effectively construct a Σ-tree automaton M such that M accepts a Σ-tree T if and only if $T \models \varphi$. This result carries over to open formulas by the following considerations.

For a Σ-tree T and $S \subseteq V(T)$, let T_S denote the $\Sigma \times \{0, 1\}$-tree obtained from T by extending the labels of T with one bit that indicates whether the labeled node belongs to S or not. Now, it is easy to generalize Thatcher and Wright's result as follows (see, e.g., [Arnborg et al. 1991]): Given an MSO formula $\varphi(X)$ for Σ-trees, one can effectively construct a $\Sigma \times \{0, 1\}$-tree automaton $M_{\varphi(X)} = (Q, \Sigma \times \{0, 1\}, \delta, q_0, Q_{\mathsf{acc}})$ such that for each Σ-tree T and $S \subseteq V(T)$, $T \models \varphi(S)$ if and only if $M_{\varphi(X)}$ accepts T_S.

2.5 Tree Interpretations

Let $\varphi(X)$ be a fixed MSO formula on (labeled) graphs and $k \geq 1$ a constant. Arnborg et al. [1991] have shown that, given a graph G and a tree decomposition of G of width k, one can construct in linear time

(1) a Σ-tree T^I where $\Sigma = \{0, 1\}^{k'}$ for $k' \in O(k^2)$,
(2) an MSO formula $\varphi^I(X)$ for Σ-trees, and
(3) a surjective mapping π from the set $L(T^I)$ of leaves of T^I onto $V(G) \cup E(G)$,

such that for each set $S^I \subseteq V(T^I)$ the following holds: $T^I \models \varphi^I(S^I)$ if and only if

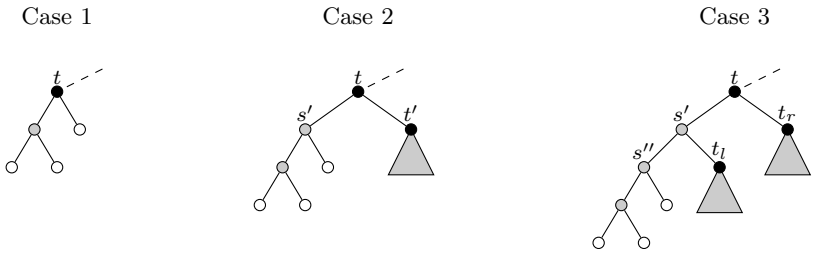

Fig. 1. Construction of tree interpretations.

(a) $S^I \subseteq L(T^I)$,
(b) for all $t, t' \in L(T^I)$ with $\pi(t) = \pi(t')$, $t \in S^I$ if and only if $t' \in S^I$, and
(c) $G \models \varphi(\pi(S^I))$.

We call the tuple $(\Sigma, T^I, \varphi^I(X), \pi)$ a *tree interpretation* of $(G, \varphi(X))$.

The tree T^I and the mapping π are obtained from a rooted tree decomposition (T, r, χ) in two phases. In phase 1 new children are added to each vertex $t \in V(T)$, each child representing a vertex $v \in \chi(t)$ (indicated by $\pi(t) = v$) or an edge uv with $u, v \in \chi(t)$ (indicated by $\pi(t) = uv$). In phase 2 additional nodes are inserted to make the tree binary.

For our purposes it is convenient to start this construction with a *nice* tree decomposition (see Section 2.2). After phase 1 is completed, each non-leaf has at least two children. In phase 2, as long as there is a node t with more than two children $t_1, \ldots, t_j$, $j > 2$, we delete the edges $tt_1, \ldots, tt_{j-1}$, add a new node t' and edges $tt', t't_1, \ldots, t't_{j-1}$. We distinguish in T^I between *black* nodes (nodes inherited from T), *white* nodes (nodes introduced in phase 1 as leaves) and *gray* nodes (nodes introduced in phase 2). It is easy to see that one can always carry out the above construction such that each black node t has in T^I exactly two children, and one of the following three cases prevails (see Fig. 1 for illustrations).

(1) The left child of t is gray, the right child of t is white (this happens exactly when t is a leaf of T).
(2) The left child of t is gray and the right child of t is black, all nodes below t's left child are white or gray (t has exactly one child t' in T).
(3) The left child s' of t is gray, the right child of t is black; the left child of s' and all nodes below it are white or gray; the right child of s' is black (t has two children t_l, t_r in T).

3. PROOF OF THE MAIN RESULT

3.1 Solution Trees

In this section we introduce solution trees and show how they can be constructed in linear time (Lemma 1).

A tuple $\mathcal{S} = (\mathcal{T}, Q, \sigma, \lambda)$ is a *solution tree* for a graph G if the following conditions hold.

(1) $\mathcal{T} = (T, r, \chi)$ is a nice tree decomposition of G.

(2) Q is a finite set.

(3) σ is a labeling that assigns to each node $t \in V(T)$ a set $\sigma(t)$ of pairs $P = (q, U)$ where $q \in Q$ and $U \subseteq \chi^*(t)$ (recall from Section 2.2 the definition of $\chi^*(t)$). We call the pair P a *solution state*.

(4) λ is a mapping that assigns to each node $t \in V(T)$ and each solution state $P \in \sigma(t)$ a set $\lambda(t, P)$ with the following properties.
   (a) If t is a leaf then $\lambda(t, P) = \emptyset$.
   (b) If t has exactly one child t' then $\lambda(t, P) \subseteq \sigma(t')$.
   (c) If t has exactly two children t_1 and t_2 then $\lambda(t, P) \subseteq \sigma(t_1) \times \sigma(t_2)$.

The purpose of a solution state $P = (q, U) \in \sigma(t)$ is to indicate that there is a solution $S \subseteq V(G) \cup E(G)$ with $G \models \varphi(S)$ that, projected on $G[\chi^*(t)]$, yields the set U; the element q represents properties of S regarding vertices and edges that appeared in sets $\chi^*(t')$ below t. The purpose of the mapping λ is to indicate direct dependencies of solution states $P \in \sigma(t)$ from solution states $P' \in \sigma(t')$ of children t' of t.

The *width* of a solution tree $\mathcal{S}$ is the width of its underlying tree decomposition $\mathcal{T}$. We say that $\mathcal{S}$ *accepts* a set $S \subseteq V(G) \cup E(G)$ if for each tree node $t \in V(T)$ we can pick a solution state $P_t = (q_t, U_t) \in \sigma(t)$ such that $U_t = S \cap \chi^*(t)$ and the following two conditions hold:

(1) If t has exactly one child t' then $P_{t'} \in \lambda(t, P_t)$.

(2) If t has exactly two children t_1 and t_2 then $(P_{t_1}, P_{t_2}) \in \lambda(t, P_t)$.

We say that $\mathcal{S}$ *accepts* $S \subseteq V(G) \cup E(G)$ *with* P *at* t if the above holds such that $P = P_t$.

Let $\varphi(X)$ be an MSO formula on (labeled) graphs. A solution tree $\mathcal{S}$ for G *characterizes* $\varphi(X)$ if $\mathcal{S}$ accepts a set $S \subseteq V(G) \cup E(G)$ if and only if $G \models \varphi(S)$.

If we are given a solution tree $\mathcal{S} = (\mathcal{T}, Q, \sigma, \lambda)$ for G that characterizes $\varphi(X)$, then we can easily decide whether there exists a solution $S \subseteq V(G) \cup E(G)$ such that $G \models \varphi(S)$ and find such S if it exists. Let $\mathcal{T} = (T, r, \chi)$. Clearly, if $\sigma(r) = \emptyset$ then there is no solution. On the other hand, if $\sigma(r) \neq \emptyset$, then we can assign to each node t of T a solution state $P_t = (q_t, U_t) \in \sigma(t)$ by a single top-down traversal of T, choosing $P_{t'} \in \lambda(t, P_t)$ if t has one child t' and choosing $(P_{t_1}, P_{t_2}) \in \lambda(t, P_t)$ if t has two children t_1 and t_2. The set $S = \bigcup_{t \in V(T)} U_t$ is then a solution.

LEMMA 1. *Let $\varphi(X)$ be a fixed MSO formula on (labeled) graphs and let k be a fixed positive integer. Given a graph G of order n and treewidth k, we can compute in time $O(n)$ a solution tree $\mathcal{S} = (\mathcal{T}, Q, \sigma, \lambda)$ for G of width k that characterizes $\varphi(X)$ where $|V(T)| = O(n)$ and the set Q depends on $\varphi(X)$ and k only.*

PROOF. We compute a solution tree $\mathcal{S}$ as follows.

Step 1. We compute a nice tree decomposition $\mathcal{T} = (T, r, \chi)$ of G with $O(n)$ nodes and of width k. This can be accomplished in time $O(n)$ (see Section 2.2).

Step 2. We compute a tree interpretation $(\Sigma, T^I, \varphi^I(X), \pi)$ of $(G, \varphi(X))$ in time $O(n)$ (see Section 2.5). Thus, T^I is a Σ-tree, $\Sigma = \{0,1\}^{k'}$, $k' \in O(k^2)$, and $\varphi^I(X)$ is an MSO formula on Σ-trees.

Step 3. We compute a tree automaton $M_{\varphi^I(X)} = (Q, \Sigma \times \{0,1\}, \delta, q_0, Q_{\mathsf{acc}})$ for $\Sigma \times \{0,1\}$-trees that accepts $\varphi^I(X)$ (see Section 2.4). This step depends only on $\varphi(X)$ and k and can therefore be carried out in constant time.

Before proceeding with the algorithm we introduce further definitions and make some observations. Let $S^I \subseteq V(T^I)$, $S = \pi(S^I) \subseteq E(G) \cup V(G)$ and let T^I_S be the $\Sigma \times \{0,1\}$-tree that corresponds to T^I and S^I (see the end of Section 2.4). Assume $M_{\varphi^I(X)}$ accepts $T^I_{S^I}$. By the definition of tree interpretations it follows that non-leaf nodes are labeled with pairs $(a,0)$ since S^I contains leaves only (by Condition 3(a) of the definition of tree interpretations). Also, any two leaves t_1, t_2 with $\pi(t_1) = \pi(t_2)$ share the same label. When we deterministically execute $M_{\varphi^I(X)}$ on $T^I_{S^I}$, we maintain additional information on the status of edges and vertices of G regarding membership in S. For that purpose we assign to the nodes of $T^I_{S^I}$ solution states (and not just states as usual). For a node t with label (a,b) we distinguish two cases:

(i) If t is a leaf (i.e., t is white), then we assign it the solution state $(\delta(q_0, q_0, (a,b)), U)$ where $U = \{\pi(t)\}$ if $b = 1$ and $U = \emptyset$ otherwise.

(ii) If t is a non-leaf (consequently $b = 0$), then we compute its solution state from the solution states of its children. Assume the children of t are assigned the solution states $P_l = (q_l, U_l)$ and $P_r = (q_r, U_r)$, respectively. We assign t the solution state $P = (\delta(q_l, q_r, (a,0)), U)$ where $U = U_l \cup U_r$ if t is gray and $U = (U_l \cup U_r) \cap \chi^*(t)$ if t is black.

We write $\delta_{\mathsf{sol}}(t, P_l, P_r) = P$ to indicate that the solution state P is computed from P_l and P_r at node t. We observe that this process assigns to each black node t a solution state (q, U) with $U = S \cap \chi^*(t)$.

Step 4. We simulate in parallel the execution of $M_{\varphi^I(X)}$ on trees $T^I_{S^I}$ for all possible sets $S^I \subseteq V(T^I)$ by guessing the missing bit in the labels of leaves, and by applying the standard power set construction for converting a nondeterministic automaton into a deterministic one (see, e.g., [Arnborg et al. 1991]). The standard conversion assigns each node a set of states. However, we apply the conversion with respect to solution states as considered above and assign each node t of T^I a set $Z(t)$ of solution states.

Step 5. We traverse the nodes t of T^I in a top-down ordering and compute sets $Z'(t) \subseteq Z(t)$ by removing "useless" elements from the sets $Z(t)$ as follows. At the root r we put $Z'(r) = \{\, (q,U) \in Z'(r) \mid q \in Q_{\mathsf{acc}} \,\}$. If t_l, t_r are the children of a node t we put $Z'(t_l) = \{\, P \in Z(t_l) \mid \delta_{\mathsf{sol}}(t, P, P_{t_r}) \in Z'(t) \text{ for some } P_{t_r} \in Z(t_r) \,\}$ and $Z'(t_r) = \{\, P \in Z(t_r) \mid \delta_{\mathsf{sol}}(t, P_{t_l}, P) \in Z'(t) \text{ for some } P_{t_l} \in Z(t_l) \,\}$.

Observe that each set $Z(t)$ (and so each set $Z'(t)$) is of constant size. Since $|V(T)| = O(n)$, all the sets $Z'(t)$ can be computed in time $O(n)$.

Step 6. Finally we compute the labelings σ and λ. For each node t of T we put $\sigma(t) = Z'(t)$. Let t be a node of T with two children t_l and t_r in T (the case where t has only one child is similar). In view of the case distinction we made at the end of Section 2.5, we can assume that the nodes t, t_l, and t_r are embedded in T^I (as black nodes) as follows: t has as its left child a gray node s' and as its right child

t_r; s' has as its left child a gray node s'' and as its right child t_l. Let $P \in Z'(t)$, $P_l \in Z'(t_l)$ and $P_r \in Z'(t_r)$. We add (P_l, P_r) to $\lambda(t, P)$ if and only if there exist $P'' \in Z'(s'')$ and $P' \in Z'(s')$ such that $\delta_{\mathsf{sol}}(s', P'', P_l) = P'$ and $\delta_{\mathsf{sol}}(t, P', P_r) = P$.

Note that for computing $\lambda(t, P)$ we only need to consider the elements of $Z'(t')$ for a constant number of nodes t'. Since the size of each set $Z'(t)$ is constant, Step 6 takes time $O(n)$. □

3.2 Cardinality Constraints

Consider a labeled graph G with cardinality constraints α and $\varphi(X)$ an MSO formula. We say that a solution tree $\mathcal{S}$ for G *characterizes* $(\varphi(X), \alpha)$ if $\mathcal{S}$ accepts a set $S \subseteq V(G) \cup E(G)$ if and only if $(G, \alpha) \models \varphi(S)$.

The next result extends Lemma 1 to graphs with cardinality constraints; Theorem 1 will be an easy consequence.

LEMMA 2. *Let $\varphi(X)$ be a fixed MSO formula on labeled graphs and let k be a fixed positive integer. Assume we are given a labeled graph with cardinality constraints (G, α) and a solution tree $\mathcal{S} = (\mathcal{T}, Q, \sigma, \lambda)$ for G that characterizes $\varphi(X)$, where $\mathcal{T}$ has N nodes and width k. Then we can obtain in time $O(N^{2k+3})$ a solution tree $\mathcal{S}^* = (\mathcal{T}, Q, \sigma^*, \lambda^*)$ of G of width k that characterizes $(\varphi(X), \alpha)$.*

PROOF. Let $\mathcal{S} = (\mathcal{T}, Q, \sigma, \lambda)$, $\mathcal{T} = (T, r, \chi)$ of width k, that characterizes $\varphi(X)$. We may assume that $\chi(r)$ is a singleton (if it is not, then we can simply extend the tree putting at most $k - 2$ forget nodes on top of the root). We assume an arbitrary total ordering of the vertices of G that allows us to associate with each set $\chi(t)$ a vector $\bar{\chi}(t)$ that lists the elements of $\chi(t)$ strictly increasing according to the chosen ordering.

For a node $t \in V(T)$ let $F(t)$ denote the set of vertices of G that are already "forgotten" at t; that is, $F(t) = \bigcup_{t' \leq t} \chi(t') \setminus \chi(t)$ where $t' \leq t$ means that t' belongs to the subtree of T rooted at t.

Let $N = \max \bigcup_{v \in V(G)} \alpha(v)$ and observe that $N = O(n)$ since $|E(G)| = O(n)$ as noted in Section 2.2.

For each node $t \in V(T)$ and each solution state $P \in \sigma(t)$ we define a set $W(t, P) \subseteq \{0, 1 \ldots, N\}^{|\chi(t)|}$ of vectors; later we will show how $W(t, P)$ can be computed. Let $\bar{\chi}(t) = (x_1, \ldots, x_j)$. Then $(n_1, \ldots, n_j) \in W(t, P)$ if and only if there exists a set $S \subseteq V(G) \cup E(G)$ such that

(1) $G \models \varphi(S)$;
(2) $\mathcal{S}$ accepts S with P at t;
(3) $\mathsf{touch}(G, S, v) \in \alpha(v)$ holds for all $v \in F(t)$;
(4) $\mathsf{touch}(G[F(t) \cup \{x_i\}], S, x_i) = n_i$ for $1 \leq i \leq j$.

The sets $W(t, P)$ can be computed by dynamic programming along a bottom-up traversal of T; we distinguish between four cases. Let t be a node of T with $\bar{\chi}(t) = (x_1, \ldots, x_j)$ and let $P \in \sigma(t)$.

Case 1: t is a *leaf*. Then $W(t, P) = \{(0, \ldots, 0)\}$ since $F(t) = \emptyset$.

Case 2: t is an *introduce node* with child t'; w.l.o.g., assume $\chi(t) \setminus \chi(t') = \{x_j\}$. Now $W(t, P)$ is the set of all vectors $(n_1, \ldots, n_{j-1}, 0)$ with $(n_1, \ldots, n_{j-1}) \in$

$W(t', P')$ for some $P' \in \lambda(t, P)$, since $F(t) = F(t')$ and x_j is not adjacent to a vertex in $F(t)$ by the first property in the definition of a tree decomposition.

Case 3: t is a *forget node* with child t'; w.l.o.g., assume $\chi(t') \setminus \chi(t) = \{x_j\}$. $W(t, P)$ is the set of all vectors $(n_1, \ldots, n_{j-1})$ with $(n'_1, \ldots, n'_{j-1}, n'_j) \in W(t', P')$ for some $P' = (q', U') \in \lambda(t, P)$, where

$$n'_j + \mathsf{touch}(G[\chi(t')], U', x_j) \in \alpha(x_j), \qquad (*)$$

and

$$n_i = n'_i + \mathsf{touch}(G[\{x_i, x_j\}], U', x_i) \quad (1 \leq i \leq j-1).$$

Case 4: t is a *join node* with children t' and t''. Now $W(t, P)$ is the set of all vectors $(n'_1 + n''_1, \ldots, n'_j + n''_j)$ with $(n'_1, \ldots, n'_j) \in W(t', P')$ and $(n''_1, \ldots, n''_j) \in W(t'', P'')$ for some $(P', P'') \in \lambda(t, P)$. The correctness of the computation of $W(t, P)$ follows from the disjointness of $F(t')$ and $F(t'')$, a consequence of the second property in the definition of a tree decomposition.

If we know the sets $W(r, P)$ for all $P \in \sigma(r)$, then we can decide immediately whether $(G, \alpha) \models \varphi(S)$ for some $S \subseteq V(G) \cup E(G)$: Let $\chi(r) = \{x_1\}$; the answer is *yes* if and only if there is some $(n_1) \in \bigcup_{P \in \sigma(r)} W(r, P)$ such that $n_1 \in \alpha(x_1)$. To find a solution S we proceed as described immediately above Lemma 1, choosing $P_r \in \sigma(r)$ with $(n_1) \in W(r, P_r)$. With a single top-down traversal of T, starting from the root, we can therefore remove from the sets $\sigma(t)$ and $\lambda(t)$ all those elements that do not correspond to a set S with $(G, \alpha) \models \varphi(S)$. Hence we are left with a solution tree that characterizes (φ, α).

It remains to review the running time. The tree has N nodes, and for each node t the set $\sigma(t)$ is of constant size. Also the sets $\lambda(t, P)$ for $P \in \sigma(t)$ are of constant size. Hence we need to compute N many sets $W(t, P)$. For each one we have to process, in the worst case (that is, when t is a join node), $(N+1)^{k+1} \cdot (N+1)^{k+1}$ many pairs of vectors, thus we can perform the computation within the claimed time. The final top-down traversal of the tree takes time $O(N)$ since $\sigma(t)$ and $\lambda(t)$ are of constant size. This completes the proof of the lemma. □

Proof of Theorem 1. Let $\varphi(X)$ be a fixed MSO formula on (labeled) graphs and let $k \geq 1$ be a constant. We show that, given a graph (G, α) with local cardinality constraints, where G is of order n and treewidth k, we can decide in time $n^{O(k)}$ whether there is some $S \subseteq V(G) \cup E(G)$ such that $(G, \alpha) \models \varphi(S)$.

First we use the algorithm described in the proof of Lemma 1 to compute a solution tree $\mathcal{S}$ for G that characterizes $\varphi(X)$; $\mathcal{S}$ is of width k and has $N = O(n)$ nodes.

Second we use the algorithm described in the proof of Lemma 2 to obtain from $\mathcal{S}$ a solution tree $\mathcal{S}^*$ for G that characterizes $(\varphi(X), \alpha)$.

Third, we apply the approach as described above Lemma 1 to $\mathcal{S}^*$ and we decide whether there is some $S \subseteq V(G) \cup E(G)$ such that $(G, \alpha) \models \varphi(S)$, and to find such an S if it exists.

From Lemmas 1 and 2 we get a total running time of $n^{O(k)}$ which is polynomial since k is assumed to be constant. □

Remark. It is well known that the number of states of a smallest Σ-tree automaton that corresponds to an MSO formula is a tower of exponents of a height that corresponds to the number of quantifier alternations in the formula. Therefore, the linear running time as provided by Lemma 1 contains a constant factor that can be non-elementary in the length of the formula. As the algorithm presented in the proof of Theorem 1 uses the algorithm of Lemma 1, the polynomial-time running time $n^{O(k)}$ also contains such a constant factor. It follows from a result of Frick and Grohe [2004] that this non-elementary dependence on the length of the formula is not only due to the tree automata approach but cannot be avoided in general (unless P = NP) even for MSO problems on trees (or strings) without local cardinality constraints.

4. APPLICATIONS

4.1 General Factors

Lovász [1970; 1972] introduced the following problem.

> General Factor
> *Instance:* A graph G and a mapping α that assigns to each vertex $v \in V(G)$ a set $\alpha(v) \subseteq \{0, \ldots, d_G(v)\}$.
> *Question:* Is there a subset $F \subseteq E(G)$ such that for each vertex $v \in V(G)$ the number of edges in F incident with v is an element of $\alpha(v)$?

This problem clearly generalizes the polynomial-time solvable f-Factor problem where the sets $\alpha(v)$ are singletons [Tutte 1952]. However, General Factor is easily seen to be NP-hard (say, by reduction from 3-Dimensional matching). Cornuéjols [1988] established a full classification of the complexity of General Factor when the assigned sets $\alpha(v)$ are restricted to some fixed class. If the number $m = \max \bigcup_{v \in V(G)} \alpha(v)$ is bounded by a constant, then we can use Courcelle's Theorem to show that the problem is linear-time decidable for graphs of bounded treewidth, using a separate predicate for each possible set $\alpha(v)$ [Samer and Szeider 2009]. For unbounded m, we can apply Theorem 1 with the formula $\varphi(X) = \forall x(X(x) \to E(x))$ that just states that X is a set of edges. Hence we have the following result.

Corollary 1. General Factor *can be solved in polynomial time for graphs of bounded treewidth.*

Theorem 1 also applies to MSO graph problems that are already NP-hard without cardinality constraints. For example, $\varphi(X)$ could express that X is a color class of a proper 3-coloring of G. With additional cardinality constraints we can require, say, for each vertex certain numbers of neighbors to be colored with color X. Furthermore, we can restrict the size of X to certain numbers: we add an additional vertex v_0 labeled a to G and connect it to all other vertices (the treewidth increases at most by 1). To $\varphi(X)$ we add the clause that X must not contain vertices labeled a (thus $v_0 \notin X$). With $\alpha(v_0)$ we can now apply restrictions on the size of X. We will use a similar construction in the next subsection.

4.2 Equitable Problems

The following problem was introduced by Meyer [1973] and has received much attention in combinatorics [Lih 1998].

> Equitable r-Coloring
> *Instance:* A graph G.
> *Question:* Is there a proper coloring of G using colors from $\{1,\ldots,r\}$ such that the sizes of any two color classes differ at most by one?

Let G be the given graph. We construct a new graph H from G by adding r new vertices $c_1,\ldots,c_r$ and all edges c_iv for $i \in \{1,\ldots,r\}$ and $v \in V(G)$. Note that $\mathsf{tw}(H) \leq \mathsf{tw}(G) + r$. Let $|V(G)| = n$. We put $\alpha(c_i) = \{\lfloor n/r \rfloor, \lceil n/r \rceil\}$, $1 \leq i \leq r$; for all other vertices v we put $\alpha(v) = \{1\}$. It is easy to construct an MSO formula $\varphi(X)$ that states: "X is a set of edges such that (1) each edge in X is incident with some vertex in $c_1,\ldots,c_r$, and (2) any two adjacent vertices $u, v \notin \{c_1,\ldots,c_r\}$ are not adjacent to the same vertex c_i, $1 \leq i \leq r$, via edges in X." Hence, from Theorem 1 we get the following result.

Corollary 2. Equitable r-Coloring *can be solved in polynomial time for graphs of bounded treewidth.*

One can generalize this construction to (appropriately defined) "equitable MSO problems" where, instead of a proper r-coloring, one asks for other MSO-expressible properties of sets of vertices.

Bodlaender and Fomin [2005] established a result that is stronger than Corollary 2. They showed that Equitable r-Coloring can be solved in polynomial time for graphs of bounded treewidth when the number r of colors is part of the input and not constant. Their algorithm uses a deep combinatorial result of Kostochka, Nakprasit, and Pemmaraju [2005].

4.3 Orientation Problems

An *edge weighting* of a graph G is a mapping w that assigns each edge a positive integer. An *orientation* of G is a mapping $\Lambda : E(G) \to V(G) \times V(G)$ with $\Lambda(uv) \in \{(u,v),(v,u)\}$ for each $uv \in E(G)$. The *weighted outdegree* of a vertex $v \in V(G)$ with respect to an edge weighting w and an orientation Λ is defined as

$$d^+_{G,w,\Lambda}(v) = \sum_{vu \in E(G) \text{ such that } \Lambda(vu)=(v,u)} w(vu).$$

Asahiro, Miyano, and Ono [2008] introduced the following problem and discussed applications and related problems.

> Minimum Maximum Outdegree
> *Instance:* A graph G, an edge weighting w of G given in unary, and a positive integer r.
> *Question:* Is there an orientation Λ of G such that $d^+_{G,w,\Lambda}(v) \leq r$ for all $v \in V(G)$?

We assume that the edge weighting w is given in unary since otherwise the problem is already NP-complete for graphs of treewidth 2 (easily seen by a simple reduction from Partition [Asahiro et al. 2008]). Asahiro et al. showed that Minimum

Maximum Outdegree can be solved in polynomial time for graphs of treewidth 2 but left its complexity open for larger treewidth bounds. We establish the general case by means of a further application of Theorem 1.

Given a graph G with edge weighting w we construct a new graph H with three sorts of vertices (say, red, green, and blue vertices). The vertices of G correspond to the red vertices of H. For each edge $e = uv \in E(G)$ we introduce in H green vertices $g_{e,i}, g'_{e,i}$ for $1 \leq i \leq w(e)$, and blue vertices b_e, b'_e; we add the edges $ug_{e,i}$, $b_e g_{e,i}$, $vg'_{e,i}$, $b'_e g'_{e,i}$ and $g'_{e,i} g_{e,i}$ for $i = 1, \ldots, w(e)$; see Fig. 4.3 for an illustration. Clearly

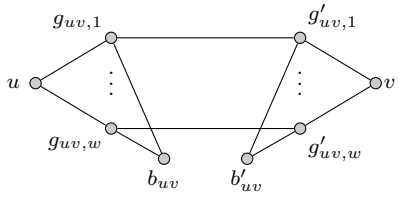

Fig. 2. Gadget representing an edge uv of weight w.

H can be constructed in polynomial time as we assume the edge weights are given in unary. Also it is easy to see that $\mathsf{tw}(H)$ is bounded in terms of $\mathsf{tw}(G)$ since we can form a tree decomposition of H by patching together tree decompositions of G and of the gadget graphs $H[\{u, v, b_{uv}, b'_{uv}\} \cup \bigcup_{i=1}^{w(uv)} \{g_{uv,i}, g'_{uv,i}\}]$, $uv \in E(G)$. The Minimum Maximum Outdegree problem can now be formulated as an MSO-LCC problem for H. For each red vertex $v \in V(H)$ we define $\alpha(v) = \{0, \ldots, r\}$, for each green vertex $v \in V(H)$ we define $\alpha(v) = \{0, 1\}$, and for each blue vertex $v \in V(H)$ we define $\alpha(v) = \{0, d_H(v)\}$. The constraints on blue vertices enforce that any set of vertices that satisfies α contains either all or none of the green vertices $g_{uv,i}$, and either all or none of the green vertices $g'_{uv,i}$. Let $\varphi(X)$ be an MSO formula expressing the property "X is a set of green vertices containing exactly one from any two adjacent green vertices." Sets $S \subseteq V(G)$ with $(H, \alpha) \models \varphi(S)$ and orientations of H with maximum outdegree at most r are in a 1-to-1 correspondence: for each edge $uv \in E(G)$ of weight w we have either $\{g_{uv,1}, \ldots, g_{uv,w}\} \subseteq S$ and $\{g'_{uv,1}, \ldots, g'_{uv,w}\} \cap S = \emptyset$ (representing $\Lambda(uv) = (u, v)$) or $\{g_{uv,1}, \ldots, g_{uv,w}\} \cap S = \emptyset$ and $\{g'_{uv,1}, \ldots, g'_{uv,w}\} \subseteq S$ (representing $\Lambda(uv) = (v, u)$). Since $\alpha(v) = \{0, \ldots, r\}$, the weighted outdegree of v is restricted to r. Hence, from Theorem 1 we get the following result.

Corollary 3. Minimum Maximum Outdegree *can be solved in polynomial time for graphs of bounded treewidth.*

5. EXTENSIONS AND LIMITS

5.1 Basic Extensions

The proof of Theorem 2 easily extends to a more general setting as follows.

Multiple Set Variables. Consider a fixed MSO formula $\varphi(X_1, \ldots, X_m)$ with $m \geq 1$ free set variables and a constant $k \geq 1$. A graph G of treewidth at most k is given together with m cardinality constraints $\alpha_1, \ldots, \alpha_m$. We want to find sets $S_1, \ldots, S_s \subseteq V(G) \cup E(G)$ such that $G \models \varphi(S_1, \ldots, S_m)$ and S_i satisfies α_i, $1 \leq i \leq m$.

It is easy to adjust the definition of solution trees to deal with multiple set variables, considering solution states of the form $(q, U_1, \ldots, U_m)$. The proof of Lemma 1 carries over to this more general setting. For the dynamic programming in the proof of Theorem 1 we need to use longer vectors: $W(t, P) \subseteq \{0, \ldots, N\}^{m|\chi(t)|}$. Thus $W(t, P)$ contains in the worst case $(N+1)^{m(k+1)}$ vectors. However, since m is constant, the total running time is still bounded by $n^{O(k)}$.

Edge and Vertex Weights. Let $\varphi(X)$, G, and α be as above. In addition, we are given a mapping w that assigns to each $x \in V(G) \cup E(G)$ a nonnegative integer weight $w(x)$ given in unary. For a set $S \subseteq V(G) \cup E(G)$, a subgraph H of G, and a vertex $v \in V(H)$, we define

$$\mathsf{touch}_w(H, S, v) = \sum_{u \in N_H(v) \cap S} w(u) + \sum_{e \in I_H(v) \cap S} w(e)$$

We ask if there is a set $S \subseteq V(G) \cup E(G)$ such that $G \models \varphi(S)$ and for each $v \in V(G)$ we have $\mathsf{touch}_w(G, S, v) \in \alpha(v)$. Observe that if all elements have weight 1, then $\mathsf{touch}_w(G, S, v) = \mathsf{touch}(G, S, v)$. It is evident that our proof of Theorem 1 generalizes to the weighted case simply by considering in Case 3 of the dynamic programming part touch_w instead of touch.

The weighted version of Theorem 1 can be used, for example, to simplify the gadget construction of Section 4.3 for the Minimum Maximum Outdegree problem.

Conditional Cardinality Constraints. The following variant of MSO-LCC problems appears to be useful for applications to answer set programming with weight constraints. Consider a fixed MSO formula $\varphi(X)$ and a constant $k \geq 1$. A graph G of treewidth at most k is given together with cardinality constraints α. We want to find a set $S \subseteq V(G) \cup E(G)$ such that $G \models \varphi(S)$ and $\mathsf{touch}(G, S, v) \in \alpha(v)$ holds for all $v \in S$ (thus, for $v \in V(G) \setminus S$ the constraint $\alpha(v)$ is ignored). We proceed as for the unconditional cardinality constraints. We do not need to modify solution trees or Lemma 1, the only difference is Case 3 in the dynamic programming part in the proof of Theorem 1, where we simply ignore condition (*) if $x_j \notin U_j$.

5.2 Quantified MSO-LCC Problems

The free set variable of an MSO formula defining an MSO-LCC problem is implicitly existentially quantified; this is also the case for multiple set variables as considered above. It would be desirable to extend Theorem 1 to a more general form of MSO-LCC problems where cardinality constraints are applied to second-order variables that are arbitrarily quantified, not just existentially. That is, we consider problems that can be stated in terms of an expression of type

$$(Q) \qquad (G, \alpha_1, \ldots, \alpha_m) \models Q_1 X_1, \ldots, Q_m X_m \; \varphi(X_1, \ldots, X_m)$$

where $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq m$, G is labeled graph and $\alpha_1, \ldots, \alpha_m$ are local cardinality constraints for G, α_i restricting X_i. We will refer to problems expressible in form (Q) where $Q_1X_1, \ldots, Q_mX_m\ \varphi(X_1, \ldots, X_m)$ is fixed and $(G, \alpha_1, \ldots, \alpha_m)$ is part of the instance as *Q-MSO-LCC problems*. If $\alpha_1 = \cdots = \alpha_m$ then we say that the Q-MSO-LCC problem under consideration is *uniform*. We will show that there exist uniform Q-MSO-LCC problems that are already NP-hard for graphs of treewidth 2. Thus it is unlikely that Theorem 1 can be generalized to Q-MSO-LCC problems.

Before we give the hardness proof let us define the semantics of expressions of type (Q) more exactly. For sets $S_1, \ldots, S_{m-1} \subseteq V(G) \cup E(G)$ the expression $(G, \alpha_1, \ldots, \alpha_m) \models \exists X_m \varphi(S_1, \ldots, S_{m-1}, X_m)$ means that S_i satisfies α_i for $1 \leq i \leq m-1$ and there exists a set $S_m \subseteq V(G) \cup E(G)$ satisfying α_m such that $G \models \varphi(S_1, \ldots, S_m)$; similarly, $(G, \alpha_1, \ldots, \alpha_m) \models \forall X_m \varphi(S_1, \ldots, S_{m-1}, X_m)$ means that S_i satisfies α_i for $1 \leq i \leq m-1$ and for all sets $S_m \subseteq V(G) \cup E(G)$ satisfying α_m we have $G \models \varphi(S_1, \ldots, S_m)$. Applying this quantification m times leads to expressions of type (Q).

THEOREM 2. *There are uniform quantified MSO-LCC problems that are already* NP*-hard for graphs of treewidth 2.*

PROOF. To simplify the presentation we will first show the result for non-uniform Q-MSO-LCC problems on labeled graphs, and then explain how constraints can be made uniform and labels avoided. Consider the following problem.

> RED-GREEN FACTOR
> *Instance:* A graph G with some of its edges labeled red, some edges labeled green, and local cardinality constraints $\alpha_R, \alpha_G, \alpha$.
> *Question:* Is there a set E_R of red edges satisfying α_R such that for each set E_G of green edges satisfying α_G there exists a set E' of edges satisfying α and containing exactly one edge from E_R and all the edges from E_G?
> We call a set E_R with this property a *red-green factor* of G.

Evidently RED-GREEN FACTOR is a Q-MSO-LCC problem as it can be expressed as $(G, \alpha_R, \alpha_G, \alpha) \models \exists X_1 \forall X_2 \exists X_3\ \varphi(X_1, X_2, X_3)$ where $\varphi(X_1, X_2, X_3)$ just says that X_1, X_2 are sets of red and green edges, respectively, and X_3 is a set of edges containing exactly one edge from X_1 and all the edges from X_2.

We show that RED-GREEN FACTOR is NP-hard for labeled graphs of treewidth 2, reducing the NP-hard problem 3-SAT [Garey and Johnson 1979]. Let F be an instance of 3-SAT consisting of m clauses $C_1, \ldots, C_m$ over a set $V(F)$ of n variables. Let $\ell_1, \ldots, \ell_{2n}$ be the $2n$ literals over $V(F)$ in such an ordering that $\ell_i \in V(F)$ and $\ell_{i+1} = \neg\ell_i$ holds for odd i. A clause C_i can be written as the disjunction of three distinct literals $\ell_{a_i}, \ell_{b_i}, \ell_{c_i}$ for $1 \leq a_i < b_i < c_i \leq 2n$. We consider a truth assignment τ as a set of literals containing either v or $\neg v$ for each $v \in V(F)$. τ satisfies F if it intersects with each clause of F.

From F we construct a labeled graph G as follows (see Fig. 5.2 for an illustration).

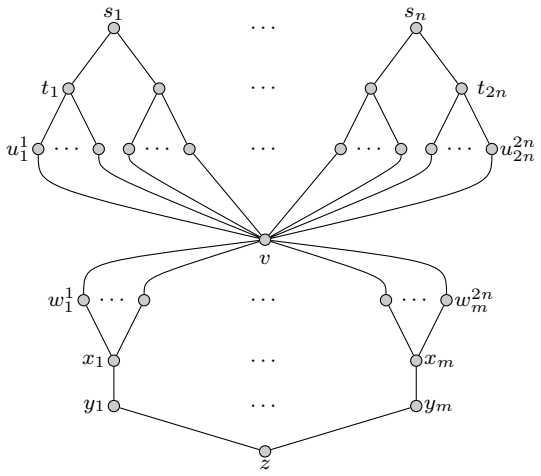

Fig. 3. Illustration for the proof of Theorem 2.

We define the vertex set as $V(G) = \bigcup_{i=1}^{7} V_i$ where

$$\begin{aligned}
V_1 &= \{v, z\},\\
V_2 &= \{\, s_i \mid 1 \le i \le n \,\},\\
V_3 &= \{\, t_i \mid 1 \le i \le 2n \,\},\\
V_4 &= \{\, u_i^j \mid 1 \le j \le i \le 2n \,\},\\
V_5 &= \{\, w_i^j \mid 1 \le j \le 2n,\ 1 \le i \le m \,\},\\
V_6 &= \{\, x_i \mid 1 \le i \le m \,\},\\
V_7 &= \{\, y_i \mid 1 \le i \le m \,\}.
\end{aligned}$$

We define the edge set as $E(G) = \bigcup_{i=1}^{7} E_i$ where

$$\begin{aligned}
E_1 &= \{\, s_{\lceil i/2 \rceil} t_i \mid 1 \le i \le 2n \,\},\\
E_2 &= \{\, t_i u_i^j \mid 1 \le j \le i \le 2n \,\},\\
E_3 &= \{\, u_i^j v \mid 1 \le j \le i \le 2n \,\},\\
E_4 &= \{\, v w_i^j \mid 1 \le i \le m,\ 1 \le j \le 2n \,\},\\
E_5 &= \{\, w_i^j x_i \mid 1 \le i \le m,\ 1 \le j \le 2n \,\},\\
E_6 &= \{\, x_i y_i \mid 1 \le i \le m \,\},\\
E_7 &= \{\, y_i z \mid 1 \le i \le m \,\}.
\end{aligned}$$

We label the edges in E_1 red and the edges in E_7 green.

It remains to specify the cardinality constraints. We set $\alpha_R(s_i) = \{1\}$ for $1 \le i \le n$; for all other vertices we do not restrict α_R (that is, we set $\alpha_R(u) = \{0, \ldots, d(u)\}$). Similarly, we set $\alpha_G(z) = \{1\}$ and leave α_G unrestricted for all other vertices.

Finally we define α as follows:

$$\begin{aligned}
\alpha(s_i) &= \alpha_R(s_i) = \{1\} \quad \text{for } 1 \le i \le n,\\
\alpha(t_i) &= \{0, i+1\} \quad \text{for } 1 \le i \le 2n,\\
\alpha(u_i^j) &= \{0,2\} \quad \text{for } 1 \le j \le i \le 2n,\\
\alpha(v) &= \{2n\},\\
\alpha(w_i^j) &= \{0,2\} \quad \text{for } 1 \le i \le m,\ 1 \le j \le n,\\
\alpha(x_i) &= \{0, 2n-a_i+1, 2n-b_i+1, 2n-c_i+1\} \quad \text{for } 1 \le i \le m,\\
\alpha(y_i) &= \{0,2\} \quad \text{for } 1 \le i \le m,\\
\alpha(z) &= \alpha_G(z) = \{1\}.
\end{aligned}$$

In fact, F is encoded into the sets $\alpha(x_i)$ for $1 \le i \le m$, all other components of G are generic and depend only on n and m.

Claim 1: F has a satisfying truth assignment if and only if G has a red-green factor.

To prove the claim, let τ be a satisfying truth assignment of F. We show that $E_R = \{\, s_{\lceil i/2 \rceil} t_i \mid \ell_i \in \tau,\ 1 \le i \le 2n \,\}$ is a red-green factor of G. For an arbitrarily chosen set E_G of green edges satisfying α_G we define a set $E' \subseteq E(G)$ as follows. First we observe that $E_G = \{y_r z\}$ for some $r \in \{1, \ldots, m\}$ since $\alpha_G(z) = \{1\}$. Furthermore, since τ satisfies F, there exists an integer $q \in \{a_r, b_r, c_r\}$ such that $\ell_q \in \tau$. We define the set E' as the union of the following sets:

$$\begin{gathered}
\{s_{\lceil q/2 \rceil} t_q\},\\
\{t_q u_q^1, \ldots, t_q u_q^q\},\\
\{u_q^1 v, \ldots, u_i^q v\},\\
\{v w_r^1, \ldots, v w_r^{2n-q}\},\\
\{w_r^1 x_r, \ldots, w_r^{2n-q} x_r\},\\
\{x_r y_r, y_r z\}.
\end{gathered}$$

Observe that by definition we have $E' \cap E_R = \{s_{\lceil q/2 \rceil} t_q\}$ and $\{y_r z\} = E_G \subseteq E_R$. It is easy to check that E' satisfies α. Since E_G was chosen arbitrarily, it follows that E_R is indeed a red-green factor of G.

Conversely, assume that G has a red-green factor E_R. Since E_R satisfies α_R it follows that the set $\tau = \{\, \ell_i \mid 1 \le i \le 2n,\ s_{\lceil i/2 \rceil} t_i \in E_R \,\}$ is a truth assignment of F. To show that τ satisfies F, we pick a clause C_r of F arbitrarily and we let $E_G = \{y_r z\}$. E_G evidently satisfies α_G. Since E_R is a red-green factor, there exists a set $E' \subseteq E(G)$ satisfying α that contains E_G and exactly one edge from E_R. Hence there is an integer $q \in \{1, \ldots, 2n\}$ such that $E_R \cap E' = \{s_{\lceil q/2 \rceil} t_q\}$. It follows that $|E' \cap E_2| = q$ since $\alpha(t_i) = \{0, i+1\}$. In turn, we get $q = |E' \cap E_2| = |E' \cap E_3|$ since $\alpha(u_i^j) = \{0,2\}$.

On the other hand, since $E_G \cap E' = \{z y_r\}$ and $\alpha(y_i) = \{0,2\}$ we have $E_6 \cap E' = \{y_r x_r\}$. Since $\alpha(y_i) = \{0, 2n-a_i+1, 2n-b_i+1, 2n-c_i+1\}$ we have $|E' \cap E_5| \in \{2n-a_r, 2n-b_r, 2n-c_r\}$. In turn, since $\alpha(w_i^j) = \{0,2\}$, we have $|E' \cap E_4| \in \{2n-a_r, 2n-b_r, 2n-c_r\}$. And since $\alpha(v) = \{2n\}$, we have $|E' \cap E_3| + |E' \cap E_4| = 2n$, consequently $q \in \{a_r, b_r, c_r\}$. Thus τ contains at least one of the literals $\ell_{a_r}, \ell_{b_r}, \ell_{c_r}$,

that is, τ satisfies C_r. Since C_r was chosen arbitrarily, we conclude that τ satisfies all clauses of F. Hence the claim is established.

The instance $(G, \alpha_R, \alpha_G, \alpha)$ of Red-Green Factor can certainly be constructed in polynomial time from a given 3-Sat instance F, and since 3-Sat is NP-hard, we conclude from Claim 1 that Theorem 2 holds for non-uniform Q-MSO-LCC problems on labeled graphs.

Next we show how the above reduction can be modified to allow uniform cardinality constraints, thus the NP-hardness does not rely on non-uniform cardinality constraints. Let $\varphi_G(S)$ be an MSO formula that is true for G if and only if S is a set of edges such that for any two incident edges $e_1, e_2 \in E(G)$, where one of them is green, either both or none of them belong to S. We will use $\varphi_G(S)$ to replace the cardinality constraint $\alpha(y_i) = \{0, 2\}$. We also define an MSO formula $\varphi_R(S)$ similarly to $\varphi_G(S)$, just requiring that one of the two edges is red instead of green. We will use $\varphi_R(S)$ to replace the cardinality constraint $\alpha(t_i) = \{0, i+1\}$. Accordingly, we define a new cardinality constraint α' by removing from α the restrictions $\alpha(t_i)$ and $\alpha(y_i)$, using the formulas $\varphi_R(S)$ and $\varphi_G(S)$ instead. More specifically, we let $\alpha'(t_i) = \{0, \ldots, d(t_i)\}$ for $1 \leq i \leq 2n$, $\alpha'(y_i) = \{0, \ldots, d(y_i)\}$ for $1 \leq i \leq m$, and $\alpha'(u) = \alpha(u)$ for all other vertices u of G. It is easy to verify that indeed $(G, \alpha_R, \alpha_G, \alpha) \models \exists X_1 \forall X_2 \exists X_3 \ \varphi(X_1, X_2, X_3)$ if and only if $(G, \alpha', \alpha', \alpha') \models \exists X_1 \forall X_2 \exists X_3 \ \varphi(X_1, X_2, X_3) \wedge \varphi_G(X_3) \wedge \varphi_R(X_3)$. Hence the above NP-hardness result carries over to uniform Q-MSO-LCC problems.

It remains to describe how labels can be avoided. Clearly we may assume that the given 3-Sat instance F has at least 2 clauses. This implies that all vertices of the graph G constructed from F are of degree at least 2. Thus we can use new vertices of degree 1 to mark the vertices of G. More specifically, let G' be the graph obtained from G by adding to each vertex s_i a new neighbor s'_i (of degree 1), $1 \leq i \leq n$, and by adding to z two new neighbors z', z'' (of degree 1). Clearly G' has the same treewidth as G. For a fixed integer $j \geq 1$ we can define an MSO formula $\varphi_j(e)$ which is true if and only if e is an edge with both of its ends having degree at least 2 and one of its ends being adjacent to exactly j vertices of degree 1 (in fact, first-order logic suffices to define $\varphi_j(e)$). Using $\varphi_1(e)$ and $\varphi_2(e)$ to distinguish between red and green edges allows us to express Red-Green Factor as a uniform Q-MSO-LCC problem on unlabeled graphs of treewidth at most 2. This completes the proof of Theorem 2. □

5.3 Fixed-Parameter Intractability

The polynomial-time algorithm developed in the proof of Theorem 1 runs in time $O(n^{2k+3})$ for graphs of order n and constant treewidth k. Thus, the order of the polynomial depends on k. The question arises whether this dependency is necessary: *Is there a better algorithm with a running time of, say, $O(n^c)$ where c is a constant independent of k?* We give a negative answer subject to the complexity theoretic assumption W[1] $\neq$ FPT from the area of Parameterized Complexity.

Let us first review some basic concepts of Parameterized Complexity; for more information we refer to other sources [Downey and Fellows 1999; Flum and Grohe 2006; Niedermeier 2006]. An instance of a parameterized problem is a pair (x, k), where x is the *main part* and k (usually a non-negative integer) is the *parameter*.

A parameterized problem is *fixed-parameter tractable* if it can be solved in time $O(f(k)|x|^c)$ where f is a computable function and c is a constant independent of k. FPT denotes the class of all fixed-parameter tractable decision problems. Parameterized Complexity offers a completeness theory similar to the theory of NP-completeness for non-parameterized problems. A parameterized problem P *fpt-reduces* to a parameterized problem Q if we can transform an instance (x, k) of P into an instance $(x', g(k))$ of Q in time $O(f(k)|x|^c)$ (f, g are arbitrary computable functions, c is a constant) such that (x, k) is a yes-instance of P if and only if $(x', g(k))$ is a yes-instance of Q. A parameterized complexity class is the class of parameterized decision problems fpt-reducible to a certain parameterized decision problem. Of particular interest is the class W[1] which is considered as the parameterized analog to NP. For example, the Clique problem (given a graph G and an integer k, decide whether G contains a complete subgraph on k vertices), parameterized by k, is well-known to be W[1]-complete. It is believed that FPT $\neq$ W[1], and there is strong theoretical evidence that supports this belief; for example, FPT = W[1] implies that the Exponential Time Hypothesis fails (see [Flum and Grohe 2006]).

Samer and Szeider [2009] have shown that General Factor is W[1]-hard when parameterized by treewidth, using an fpt-reduction from Clique. W[1]-hardness even holds if the instance graph is bipartite and all vertices of one side are assigned the set $\{1\}$. Since General Factor can be expressed as an MSO-LCC problem, we can answer the above question negatively.

6. CONCLUDING REMARKS

We have established an algorithmic meta-theorem that provides for each MSO-LCC problem an algorithm that solves the problem in polynomial-time if the input graph has bounded treewidth. In way of contrast we have shown that the algorithmic meta-theorem cannot be extended to the class of Q-MSO-LCC problems unless P = NP.

It would be interesting to determine classes of Q-MSO-LCC problems for which the meta-theorem still holds. More specifically, for quantifiers $Q_1, \ldots, Q_m \in \{\exists, \forall\}$ let $Q_1 \ldots Q_m$-MSO-LCC denote the class of all Q-MSO-LCC problems that can be stated in terms of an expression of type $(G, \alpha_1, \ldots, \alpha_m) \models Q_1 X_1, \ldots, Q_m X_m\ \varphi(X_1, \ldots, X_m)$ (the quantifiers within φ can be arbitrary). For example, the Red-Green Factor problem is an $\exists\forall\exists$-MSO-LCC problem. It would be interesting to determine for all strings $\mathcal{S} \in \{\exists, \forall\}^*$ the complexity of $\mathcal{S}$-MSO-LCC problems on graphs of bounded treewidth

REFERENCES

Arnborg, S., Corneil, D. G., and Proskurowski, A. 1987. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods 8,* 2, 277–284.

Arnborg, S., Lagergren, J., and Seese, D. 1991. Easy problems for tree-decomposable graphs. *J. Algorithms 12,* 2, 308–340.

Asahiro, Y., Miyano, E., and Ono, H. 2008. Graph classes and the complexity of the graph orientation minimizing the maximum weighted outdegree. In *Proceedings of CATS 2008, Computing: The Australasian Theory Symposium, University of Wollongong, New South Wales, Australia, January 22-25, 2008, part of the Australasian Computer Society Week (ACSW 2008)*, J. Harland and P. Manyem, Eds. Conferences in Research and Practice in Information Technology, vol. 77. Australian Computer Society, Sydney, 97–106.

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

Bodlaender, H. L. 1998. A partial k-arboretum of graphs with bounded treewidth. *Theoret. Comput. Sci. 209,* 1-2, 1–45.

Bodlaender, H. L. and Fomin, F. V. 2005. Equitable colorings of bounded treewidth graphs. *Theoret. Comput. Sci. 349,* 1, 22–30.

Cornuéjols, G. 1988. General factors of graphs. *J. Combin. Theory Ser. B 45,* 2, 185–198.

Courcelle, B. 1987. Recognizability and second-order definability for sets of finite graphs. Tech. Rep. I-8634, Université de Bordeaux.

Courcelle, B. 1990. Graph rewriting: an algebraic and logic approach. In *Handbook of theoretical computer science, Vol. B*. Elsevier Science Publishers, North-Holland, Amsterdam, 193–242.

Downey, R. G. and Fellows, M. R. 1999. *Parameterized Complexity*. Monographs in Computer Science. Springer Verlag, New York.

Flum, J. and Grohe, M. 2006. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series, vol. XIV. Springer Verlag, Berlin.

Frick, M. and Grohe, M. 2004. The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic 130,* 1-3, 3–31.

Garey, M. R. and Johnson, D. R. 1979. *Computers and Intractability*. W. H. Freeman and Company, New York, San Francisco.

Grohe, M. 2007. Logic, graphs, and algorithms. In *Logic and Automata: History and Perspectives*, J. Flum, E. Grädel, and T.Wilke, Eds. Texts in Logic and Games, vol. 2. Amsterdam University Press, 357–422.

Kloks, T. 1994. *Treewidth: Computations and Approximations*. Springer Verlag, Berlin.

Kostochka, A. V., Nakprasit, K., and Pemmaraju, S. V. 2005. On equitable coloring of d-degenerate graphs. *SIAM J. Discrete Math. 19,* 1, 83–95.

Lih, K.-W. 1998. The equitable coloring of graphs. In *Handbook of combinatorial optimization, Vol. 3*. Kluwer Academic Publishers, Dortrecht, Boston, MA, 543–566.

Lovász, L. 1970. The factorization of graphs. In *Combinatorial Structures and their Applications (Proc. Calgary Internat. Conf., Calgary, Alta., 1969)*. Gordon and Breach, New York, 243–246.

Lovász, L. 1972. The factorization of graphs. II. *Acta Math. Acad. Sci. Hungar. 23*, 223–246.

Meyer, W. 1973. Equitable coloring. *Amer. Math. Monthly 80*, 920–922.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, Oxford.

Rose, D. J. 1974. On simple characterizations of k-trees. *Discrete Math. 7*, 317–322.

Samer, M. and Szeider, S. 2009. Tractable cases of the extended global cardinality constraint. *Constraints*. Springer Online First (print edition to appear).

Thatcher, J. W. and Wright, J. B. 1968. Generalized finite automata theory with an application to a decision problem of second-order logic. *Math. Systems Theory 2*, 57–81.

Tutte, W. T. 1952. The factors of graphs. *Canadian J. Math. 4*, 314–328.

Received April 2009; revised January 2010; accepted April 2010