

# Tractable Cases of the Extended Global Cardinality Constraint

Marko Samer

Department of Computer Science

TU Darmstadt, Germany

E-mail: `samer@cs.tu-darmstadt.de`

Stefan Szeider

Department of Computer Science

Durham University, UK

E-mail: `stefan@szeider.net`

## Abstract

We study the consistency and domain consistency problem for extended global cardinality (EGC) constraints. An EGC constraint consists of a set  $X$  of variables, a set  $D$  of values, a domain  $D(x) \subseteq D$  for each variable  $x$ , and a “cardinality set”  $K(d)$  of non-negative integers for each value  $d$ . The problem is to instantiate each variable  $x$  with a value in  $D(x)$  such that for each value  $d$ , the number of variables instantiated with  $d$  belongs to the cardinality set  $K(d)$ . It is known that this problem is NP-complete in general, but solvable in polynomial time if all cardinality sets are intervals.

First we pinpoint connections between EGC constraints and general factors in graphs. This allows us to extend the known polynomial-time case to certain non-interval cardinality sets.

Second we consider EGC constraints under restrictions in terms of the treewidth of the value graph (the bipartite graph representing variable-value pairs) and the cardinality-width (the largest integer occurring in the cardinality sets). We show that EGC constraints can be solved in polynomial time for instances of bounded treewidth, where the order of the polynomial depends on the treewidth. We show that (subject to the complexity theoretic assumption  $\text{FPT} \neq \text{W}[1]$ ) this dependency cannot be avoided without imposing additional restrictions. If, however, also the cardinality-width is bounded, this dependency gets removed and EGC constraints can be solved in linear time.

*Keywords:* Global constraints, general factor problem, domain consistency, bounded treewidth, parameterized complexity

## 1 Introduction

Constraint satisfaction is a powerful formalism for encoding a wide range of combinatorial problems and is therefore attractive for both practitioners as well as theorists [28]. Special purpose constraints with non-constant arity, often referred to as *global constraints*, occur frequently in constraint modeling. Efficient propagation algorithms for such constraints are important for the performance of constraint solvers [17]. Currently the Global Constraint Catalog [1] lists 276 global constraints.

In the sequel we focus on *extended global cardinality constraints* (EGC constraints, for short), constraints that occur frequently in constraint modeling and are known as *global\_cardinality* [1], *egcc* [2], *distribution* [7], and *card\_var\_gcc* [27]. An EGC constraint is specified by a set  $D$  of values, a set  $X$  of variables where each variable  $x \in X$  ranges over a set  $D(x) \subseteq D$  of values, and sets  $K(d)$  of non-negative integers associated with values  $d \in D$ ; we refer to the sets  $K(d)$  as *cardinality sets*. The EGC constraint requires that the number of variables in  $X$  instantiating to a value  $d$  must belong to the cardinality set  $K(d)$ . More specifically, an EGC constraint with variables  $X$  and domain  $D$  is *consistent* or *satisfiable* if there is a mapping  $\tau : X \rightarrow D$  such that

1.  $\tau(x) \in D(x)$  holds for all variables  $x \in X$ , and
2.  $|\{x \in X : \tau(x) = d\}| \in K(d)$  holds for all values  $d \in D$ .

Clearly, such a *solution*  $\tau$  uniquely corresponds to a set  $M \subseteq X \times D$ . The *value graph* provides a convenient visualization of the relationship between variables and values present in an EGC constraint; it is the bipartite graph with vertex sets  $X$  and  $D$ , where an edge joins a variable  $x$  with a value  $d$  if and only if  $d \in D(x)$ . Figure 1 shows an EGC constraint and its value graph.

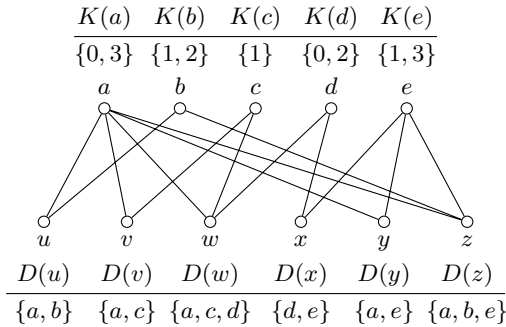


Figure 1: An EGC constraint and its value graph. The constraint is satisfied by  $\tau(u) = b$ ,  $\tau(v) = c$ ,  $\tau(w) = d$ ,  $\tau(x) = d$ ,  $\tau(y) = e$ , and  $\tau(z) = b$ .

We refer to the largest integer occurring in the cardinality sets of an EGC constraint as the *cardinality-width* of the constraint. For example, the constraint in Figure 1 has cardinality-width 3.

Typical applications of the EGC constraint arise in the context of timetable scheduling and assignment problems [8]. For example, the set  $X$  of variables can represent workers and the set  $D$  of values can represent tasks;  $t \in D(w)$  if worker  $w$  is qualified to carry out task  $t$ , and  $k \in K(t)$  if  $k$  workers are able to carry out task  $t$ . In particular, it is not hard to imagine tasks that can be carried out multiple times in parallel for different customers, where a certain number of workers is required at each customer's place. For example, if task  $t$  has to be carried out for at least three and at most seven customers, where the task requires two workers at each customer's place, we have  $K(t) = \{2i : 3 \leq i \leq 7\}$ .

In this paper, we consider the following decision problem:

EGCC-CONSISTENCY

*Instance:* An EGC constraint  $C$ .

*Question:* Is the constraint  $C$  consistent?

Quimper et al. [25] have shown that EGCC-CONSISTENCY is NP-complete. However, as observed by Régim [26], EGCC-CONSISTENCY can be decided in polynomial time by network flow algorithms if all cardinality sets are intervals (such constraints are called *global cardinality constraints*). As we will see in Section 3, both results are special cases of an earlier result of Cornuéjols.

In the following, we explore classes of EGC constraints for which the consistency problem is tractable. We follow two complementary approaches:

1. Tractability due to restrictions on the language.
2. Tractability due to restrictions on the structure.

These are the two main approaches for studying tractable classes of constraint networks, see, e.g., the surveys of Cohen and Jeavons [9] and Dechter [13], respectively.

**Restrictions on the language.** As mentioned above, EGCC-CONSISTENCY is tractable if all cardinality sets are intervals. We consider more general restrictions on the cardinality sets that still assure tractability. Our results rely on the connection between EGCC-CONSISTENCY and the *general factor problem* for graphs, the latter problem was introduced by Lovász [20, 21]. This connection seems not to be known in the constraint satisfaction literature. In view of this connection, a general result of Cornuéjols [10] for the general factor problem allows us to generalize Régin’s polynomial-time result from intervals to cardinality sets with *gaps of length at most 1* (see Section 3.1 for exact definitions). This result is the best possible: allowing a cardinality set with a gap of length 2 or larger renders EGCC-CONSISTENCY NP-hard. We adopt Cornuéjols’s algorithm for general factors to the special case of EGC constraints and show how EGCC-CONSISTENCY can be decided efficiently with graph matching as the main subroutine. This approach can also be used for efficient *domain filtering*, i.e., for removing from the domains of variables those values that do not participate in a solution, an important task in the context of constraint solving [17]. However, if we use the more general variant of domain filtering, where also cardinality sets are considered as domains (see Section 2.1), we may introduce gaps into the intervals of an instance, which may cause domain filtering to become intractable. For that reason, Régin’s flow-based algorithm can only provide bounds consistency [17]. However, by allowing cardinality sets with gaps of length at most 1, we can eliminate significantly more domain values compared to bounds consistency, while still keeping domain filtering tractable.

**Restrictions on the structure.** *Bounded treewidth* is perhaps the most widely used structural restriction on graphs and is a standard technique in the context of constraint satisfaction [13]. Many otherwise NP-hard graph problems are tractable for instances of bounded treewidth; it is generally believed that many practically relevant problems actually do have low treewidth [4, 16]. In the context of our considerations it is natural to consider EGC constraints with value graphs of bounded treewidth. Note that in many applications of the EGC constraint the number of values (size of  $D$ ) is relatively small compared to the number of variables (size of  $X$ ), for example, if many workers are supposed to collaborate on a small number of tasks. In such cases the value graph has small treewidth (see the remark at the end of Section 2.2). We present a dynamic programming algorithm that allows to decide EGCC-CONSISTENCY in polynomial time for instances of bounded treewidth. This algorithm can be easily extended to perform also domain-filtering efficiently.

The polynomial that bounds the runtime of our dynamic programming algorithm depends on the treewidth of the instance. However, if we additionally bound the cardinality-width, this dependency is removed and the algorithm runs in linear time. The question arises whether this dependency can be avoided without bounding the cardinality-width. We answer this question negatively, subject to a reasonable complexity theoretic assumption ( $\text{FPT} \neq \text{W}[1]$ , see Section 6) from the field of *parameterized complexity*, a theoretical framework introduced by Downey and Fellows [14]. As a corollary, we obtain that Lovász’s general factor problem, parameterized by the treewidth of the input graph, is  $\text{W}[1]$ -hard. This result may be of independent interest. Of related interest is the recent work of Bessi ere et al. [3], who study the parameterized complexity of several global constraints.

The remainder of this paper is organized as follows: In Section 2, we give basic definitions and background on constraints and treewidth. In Section 3, we discuss the connection

between EGC constraints and general factors in graphs, and we present tractability results by applying general results of Cornuéjols and Courcelle. In Section 4, we present the dynamic programming algorithm for instances of bounded treewidth; in Section 5, we extend this algorithm to domain filtering. Finally, in Section 6, we prove the W[1]-hardness result.

## 2 Preliminaries

### 2.1 Constraint Satisfaction

A *constraint network* consists of a finite set  $X$  of *variables*, a finite set  $D$  of *values*, and a finite set of *constraints*. Each variable  $x \in X$  ranges over a set  $D(x) \subseteq D$  of values, the *domain* of  $x$ . Each constraint  $C$  specifies the allowed combinations of values for a set  $\text{var}(C) \subseteq X$  of variables, the *scope* of  $C$ ; the *arity* of a constraint is the cardinality of its scope. An *assignment* is a mapping  $\tau$  that assigns to each variable  $x \in X$  a value  $\tau(x) \in D(x)$ . An assignment  $\tau$  *satisfies* a constraint  $C$  if  $\tau$  instantiates the variables in the scope of  $C$  such that an allowed combination of values is formed. An assignment that satisfies simultaneously all constraints is a *solution* of the constraint network. A constraint  $C$  is *consistent* or *satisfiable* if it is satisfied by at least one assignment, and it is *domain consistent* if for every variable  $x \in \text{var}(C)$  and every value  $d \in D(x)$  there exists an assignment  $\tau$  that satisfies  $C$  and instantiates  $x$  with  $d$ . Given a constraint  $C$ , *domain filtering* is the task of removing values  $d$  from domains of variables  $x \in \text{var}(C)$  if there is no assignment that satisfies  $C$  and instantiates  $x$  with  $d$ . What we call domain filtering is sometimes called “complete” domain filtering in order to emphasize that *all* superfluous values are removed from domains, in contrast to weaker forms of domain filtering that only achieve “range consistency” or “bounds consistency” [17].

An EGC constraint is specified by sets  $K(d) \subseteq \{0, 1, \dots, |\{x \in X : d \in D(x)\}|\}$  of integers associated with values  $d \in D$ ; we refer to the sets  $K(d)$  as *cardinality sets*. The EGC constraint requires that the number of variables in  $X$  instantiated with a value  $d$  must belong to the cardinality set  $K(d)$ . The *cardinality-width* of an EGC constraint is the largest integer occurring in the cardinality sets or 0 if all cardinality sets are empty. For an EGC constraint  $C$  with set  $X$  of variables and set  $D$  of values we say that  $C$  is *over*  $(X, D)$ . We also write  $C[x = d]$  and  $C[d = k]$  to denote the constraint obtained from  $C$  by setting  $D(x) = \{d\}$  and  $K(d) = \{k\}$ , respectively. Note that in contrast to extensional constraints, the domains are part of the EGC constraint here.

Now consider an EGC constraint  $C$  over  $(X, D)$  with cardinality sets  $K(d)$  for  $d \in D$ . One can consider the elements  $d \in D$  as variables with domain  $K(d)$ . This gives rise to a more general notion of domain filtering where also elements  $k \in K(d)$  are removed from  $K(d)$  if  $C[d = k]$  is inconsistent. In this paper we shall use this more general notion of domain filtering. The input size of  $C$  over  $(X, D)$  is of order  $\|C\| = \sum_{x \in X} (|D(x)| + 1) + \sum_{d \in D} (|K(d)| + 1)$ . Thus, for the value graph  $G = (V, E)$  of  $C$ , we have  $|V| + |E| \leq \|C\|$ .

Note that we consider the EGC constraint as an *intensional* constraint. Typically, global constraints are considered as intensional constraints since for many global constraints an *extensional* representation (where all combinations of values that satisfy the constraint are explicitly listed) would require exponential space. Domain filtering and consequently testing for domain consistency or consistency is trivial for extensional constraints as these properties can be read off from the constraint relation. For intensional constraints, however, testing for domain consistency or consistency is nontrivial and known to be NP-complete for several classes of constraints [2, 25], in particular for the EGC constraint. Note that whenever domain filtering can be accomplished in polynomial time for a constraint, then its consistency can be checked in polynomial time as well [2], but the converse does not hold in general [30].

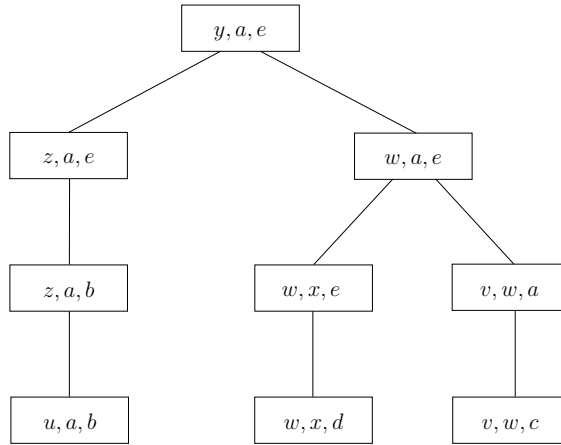


Figure 2: A tree decomposition of width 2 of the value graph in Figure 1.

## 2.2 Tree Decompositions

Treewidth is an important graph invariant which is a measure of “tree-likeness.” For graphs of treewidth bounded by a constant, many otherwise intractable problems become tractable, e.g., 3-colorability, Hamiltonicity, etc. It is generally believed that many practically relevant problems actually do have low treewidth [4].

The treewidth of a graph  $G = (V, E)$  is defined via the following notion of a decomposition: a *tree decomposition* of  $G$  is a pair  $(T, \chi)$ , where  $T$  is a tree and  $\chi$  is a labeling function with  $\chi(t) \subseteq V$  for every tree node  $t$  such that the following conditions hold:

1. Every vertex of  $G$  occurs in  $\chi(t)$  for some tree node  $t$ .
2. For every edge  $uv$  of  $G$  there is a tree node  $t$  such that  $u, v \in \chi(t)$ .
3. For every vertex  $v$  of  $G$ , the tree nodes  $t$  with  $v \in \chi(t)$  induce a connected subtree of  $T$  (“Connectedness Condition”).

The *width* of a tree decomposition  $(T, \chi)$  is the cardinality of a largest set  $\chi(t)$  minus 1 among all nodes  $t$  of  $T$ . A tree decomposition of smallest width is *optimal*. The *treewidth* of a graph  $G$  is the width of an optimal tree decomposition of  $G$ . Figure 2 shows an optimal tree decomposition of width 2 of the value graph in Figure 1.

If we consider graphs of treewidth bounded by an arbitrary but fixed constant  $k$ , we simply speak of graphs of *bounded treewidth* without explicitly mentioning  $k$ ; we will use this convention also for other parameters such as cardinality-width. Note that a graph  $G = (V, E)$  of treewidth  $k$  has at most  $k|V|$  edges [18]. Thus if  $G$  has bounded treewidth, then  $|E| = \mathcal{O}(|V|)$ .

In principle, one can compute in linear time an optimal tree decomposition of graphs with bounded treewidth [5]; however the runtime of the known linear-time algorithm imposes a huge hidden constant. In practice, one often prefers to obtain tree decompositions via heuristics. An important class of tree decomposition heuristics are based on finding an appropriate linear ordering of the vertices [6, 19].

Once a tree decomposition of small width is found, one tries to solve the problem under consideration by dynamic programming via bottom-up traversal of the tree decomposition. For such an approach it is often convenient to consider tree decompositions in the following normal form [18]: A triple  $(T, \chi, r)$  is a *nice tree decomposition* of a graph  $G$  if  $(T, \chi)$  is a tree decomposition of  $G$ , the tree  $T$  is rooted at node  $r$ , and each node of  $T$  is of one of the following four types:

1. a *leaf node*: a node having no children;
2. a *join node*: a node  $t$  having exactly two children  $t_1, t_2$ , and  $\chi(t) = \chi(t_1) = \chi(t_2)$ ;
3. an *introduce node*: a node  $t$  having exactly one child  $t'$ , and  $\chi(t) = \chi(t') \cup \{v\}$  for a vertex  $v$  of  $G$ ;
4. a *forget node*: a node  $t$  having exactly one child  $t'$ , and  $\chi(t) = \chi(t') \setminus \{v\}$  for a vertex  $v$  of  $G$ .

Note that for every vertex  $v$  of  $G$  that does not occur in  $\chi(r)$  there is exactly one node  $t_v$  with parent  $t'_v$  such that  $v \in \chi(t_v)$  and  $v \notin \chi(t'_v)$  ( $t'_v$  is a forget node). If  $v$  occurs in  $\chi(r)$ , we put  $t_v = r$ . In both cases we say that  $t_v$  is the *final node* of  $v$ .

Given a graph  $G$  with  $n$  vertices and a tree decomposition of width  $k$  of  $G$ , one can transform in polynomial time the tree decomposition into a nice tree decomposition of the same width and with at most  $4n$  nodes; if  $k$  is a constant, then this transformation can be carried out in time  $\mathcal{O}(n)$  [18].

It is easy to see (and well known) that the treewidth of a bipartite graph with bipartition  $(X, D)$  is at most the size of  $D$ : take the tree decomposition  $(T, \chi)$  where  $T$  is a star,  $D$  is the bag of the center node and for each  $x \in X$  there is a leaf node with bag  $D \cup \{x\}$ . Thus, if an EGC constraint has a small number of values, then the treewidth of its value graph is small as well.

### 3 EGC Constraints and General Factors

#### 3.1 Applying Cornuéjols's Theorem

Lovász [20, 21] introduced the following problem, known as the *general factor problem*:

GENFACTOR

*Instance:* A graph  $G = (V, E)$  and a mapping  $K$  that assigns to each vertex  $v \in V$  a set  $K(v) \subseteq \{0, \dots, \delta(v)\}$  of integers, where  $\delta(v)$  denotes the degree of  $v$  in  $G$ .

*Question:* Is there a subset  $F \subseteq E$  such that for each vertex  $v \in V$  the number of edges in  $F$  incident with  $v$  is an element of  $K(v)$ ?

Clearly, EGCC-CONSISTENCY is the special case of GENFACTOR where  $G$  is bipartite with bipartition  $(X, D)$  and  $K(v) = \{1\}$  for all  $v \in X$ . Thus, similar to EGC constraints, we call the sets  $K(v)$  *cardinality sets* and we call the largest integer occurring in the cardinality sets of a GENFACTOR instance its *cardinality-width*.

Let  $\mathcal{K}$  be a class of finite sets of non-negative integers. We denote by  $\text{GENFACTOR}(\mathcal{K})$  and  $\text{EGCC-CONSISTENCY}(\mathcal{K})$  the respective problems restricted to instances where for all vertices  $v$  the cardinality sets  $K(v)$  belong to the class  $\{S \cap \{0, 1, \dots, \delta(v)\} : S \in \mathcal{K}\}$ . For simplicity, we will also write  $\mathcal{K}$ -*constraint* to denote an instance of  $\text{EGCC-CONSISTENCY}(\mathcal{K})$ . We call a class  $\mathcal{K}$  *trivial* if each set in  $\mathcal{K}$  contains the number 0.  $\text{GENFACTOR}(\mathcal{K})$  can obviously be solved in polynomial time for each trivial class  $\mathcal{K}$  since  $F = \emptyset$  is then always a solution. A set  $S$  of integers has an  $s$ -*gap* if there exists an integer  $i$  such that  $\min(S) < i < \max(S)$  and  $\{i, \dots, i + s - 1\} \cap S = \emptyset$ . We denote by  $\mathcal{I}_s$  the class of  $(s + 1)$ -gap free sets of non-negative integers. Note that  $\mathcal{I}_0$  is nothing but the class of non-negative intervals. We can state Régin's above mentioned result as follows.

**Theorem 1** (Régin [26]). *EGCC-CONSISTENCY( $\mathcal{I}_0$ ) can be decided in polynomial time.*

**Proposition 2** (Quimper et al. [25]). EGCC-CONSISTENCY( $\mathcal{I}_0$ ) can be decided in time  $\mathcal{O}(m\sqrt{n})$ , where  $m$  and  $n$  are the number of edges and vertices of the value graph, respectively.

The following dichotomy result fully classifies the problem GENFACTOR( $\mathcal{K}$ ).

**Theorem 3** (Cornuéjols [10]). If  $\mathcal{K} \subseteq \mathcal{I}_1$  or if  $\mathcal{K}$  is trivial, then GENFACTOR( $\mathcal{K}$ ) can be decided in polynomial time. Otherwise, GENFACTOR( $\mathcal{K}$ ) is NP-complete.

Actually, the hardness result of Cornuéjols [10] does not directly imply the above dichotomy result, but it can be easily obtained by a reduction from  $k$ -dimensional matching, for  $k \geq 3$ . In fact, the hardness case of Theorem 3 even holds if the graph is bipartite, the vertices on one side have cardinality sets  $\{1\}$ , and the vertices  $v$  on the other side have cardinality sets drawn from the class  $\{S \cap \{0, 1, \dots, \delta(v)\} : S \in \mathcal{K}\}$ . Hence, it follows that the dichotomy stated in Theorem 3 also holds for EGCC-CONSISTENCY( $\mathcal{K}$ ).

**Corollary 4.** If  $\mathcal{K} \subseteq \mathcal{I}_1$ , then EGCC-CONSISTENCY can be decided in polynomial time. Otherwise, EGCC-CONSISTENCY( $\mathcal{K}$ ) is NP-complete.

Thus EGCC-CONSISTENCY( $\mathcal{I}_1$ ) can be decided in polynomial time, a proper generalization of Theorem 1. A further consequence of Corollary 4 is the NP-completeness of EGCC-CONSISTENCY( $\{\{0, 3\}\}$ ), which gives the following.

**Corollary 5.** EGCC-CONSISTENCY is NP-complete for instances of cardinality-width at least 3.

In the remainder of this section, we will show how to solve EGC constraints with cardinality sets containing gaps of length at most 1 by using Cornuéjols's algorithm [10] for the general factor problem. The basic idea is to replace the vertices of the value graph by certain gadgets that allow to reduce the constraint satisfaction problem to a polynomial number of edge-and-triangle packing problems; in fact, it suffices to replace the vertices representing values in  $D$  by gadgets. Cornuéjols, Hartvigsen, and Pulleyblank [11] have shown that the edge-and-triangle packing problem can be solved in polynomial time. We will show that standard matching algorithms actually suffice, which simplifies the original algorithm and decreases its asymptotic runtime.

Key in Cornuéjols's algorithm is the observation that cardinality sets with gaps of length at most 1 can be replaced by cardinality sets that are obtained by the intersection of an interval with a parity condition. Let  $\mathcal{I}_1^* \subseteq \mathcal{I}_1$  denote the class of all sets of integers of the form  $\{p, p+2, p+4, \dots, p+2r\}$  for  $p, r \geq 0$ . Moreover, let  $C$  be an  $\mathcal{I}_1^*$ -constraint over  $(X, D)$ . The *gadget graph* of  $C$  is constructed from the value graph of  $C$  in the following way: (i) we introduce for each edge  $e = (x, d)$  an auxiliary vertex  $u$  and replace  $e$  by the two edges  $(x, u)$  and  $(u, d)$ , and (ii) we replace each vertex  $d \in D$  by a gadget as illustrated in Figure 3, where  $\delta$  is the degree of  $d$ . In this way we obtain the following as a special case of a result of Cornuéjols [10].

**Proposition 6.** An  $\mathcal{I}_1^*$ -constraint  $C$  is consistent if and only if its gadget graph has a perfect matching. Moreover, a solution for  $C$  can be read off from a perfect matching of its gadget graph.

In particular, given a perfect matching  $M$ , an edge  $(u_i, d_i)$  is in  $M$  if and only if  $d$  is assigned to  $x_i$  in the solution corresponding to  $M$ . On the other hand, if  $(u_i, d_i)$  is not in  $M$ , there must be some edge  $(d_i, n_j)$  in  $M$  in order to cover  $d_i$  by  $M$ . So the intuitive role of the vertices  $n_j$  of the gadget graph is to allow vertices  $d_i$  to be covered by  $M$  if  $d$  is not assigned to  $x_i$ . Moreover, the edges  $(n_{2t-1}, n_{2t})$  are necessary to allow vertices  $n_j$  to be covered by  $M$  if there are not enough vertices  $d_i$  left to cover vertices  $n_j$  by edges  $(d_i, n_j)$ .

Note that checking whether a graph  $G = (V, E)$  has a perfect matching and finding one if it exists can be done in time  $\mathcal{O}(|E|\sqrt{|V|})$  [22]. On the other hand, an  $\mathcal{I}_1^*$ -constraint  $C$  whose

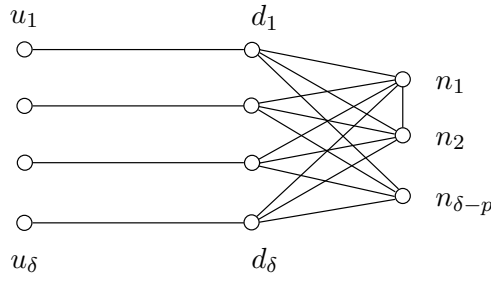


Figure 3: Gadget for a vertex  $d$  of degree  $\delta$  with neighbors  $u_1, \dots, u_\delta$  and cardinality set of the form  $K(d) = \{p, p+2, p+4, \dots, p+2r\}$  for  $r \geq 1$ . All vertices  $d_1, \dots, d_\delta$  are connected with all vertices  $n_1, \dots, n_{\delta-p}$  and there is an edge between each pair  $(n_{2t-1}, n_{2t})$  of vertices for  $1 \leq t \leq r$ .

value graph has  $m$  edges gives rise to a gadget graph with  $\mathcal{O}(m)$  vertices and  $\mathcal{O}(m)$  edges. Hence, we can check the consistency of  $C$  in time  $\mathcal{O}(m^{3/2})$ .

What remains to be shown is how we can reduce an arbitrary EGC constraint to an instance where all cardinality sets are of the above form. To achieve this aim we need the following definitions adapted from [10]: Let  $G = (X \cup D, E)$  be the value graph of an EGC constraint such that the cardinality sets have gaps of length at most 1. For a set  $M \subseteq E$ , we write  $\delta_M(v)$  to denote the *degree* of vertex  $v \in X \cup D$  relative to  $M$ , i.e., the number of edges in  $M$  incident with  $v$ . Let  $\alpha(v)$  and  $\beta(v)$  be the smallest and largest element in  $K(v)$ , respectively. Then we define the *deficiency* of  $M$  at vertex  $v$  relative to  $K$  by

$$def_M(v) = \begin{cases} 0 & \delta_M(v) \in K(v) \\ \alpha(v) - \delta_M(v) & \delta_M(v) < \alpha(v) \\ \delta_M(v) - \beta(v) & \delta_M(v) > \beta(v) \\ 1 & \delta_M(v) \notin K(v) \text{ and } \alpha(v) < \delta_M(v) < \beta(v). \end{cases}$$

Note that the last line follows since there are no gaps of length larger than 1. The *deficiency* of  $M$  relative to  $K$  is defined as  $\sum_{v \in X \cup D} def_M(v)$ . Cornuéjols's algorithmic idea is to successively decrease the deficiency of  $M$  (relative to  $K$ ) until a minimum is reached; if this minimum is 0, we have found a solution. For two integers  $a$  and  $b$ , we write  $[a, b]$  to denote the interval  $[a, b]$  if  $a \leq b$  and the interval  $[b, a]$  if  $a > b$ . Now let  $\alpha_M(v) = \min\{t \in K(v) : \text{all elements of } [t, \delta_M(v)] \cap K(v) \text{ have the same parity}\}$  and  $\beta_M(v) = \max\{t \in K(v) : \text{all elements of } [\delta_M(v), t] \cap K(v) \text{ have the same parity}\}$ . We define

$$K_M(v) = [\alpha_M(v), \beta_M(v)] \cap K(v).$$

Intuitively, the set  $K_M(v)$  consists of the largest subset of  $K(v)$  such that all its elements have the same parity and it is closest to  $\delta_M(v)$ . For example, if  $K(d) = \{2, 4, 6, 7, 9, 10\}$ , then  $K_M(d) = \{2, 4, 6\}$  if  $\delta_M(d) \leq 6$ ,  $K_M(d) = \{7, 9\}$  if  $7 \leq \delta_M(d) \leq 9$ , and  $K_M(d) = \{10\}$  if  $\delta_M(d) \geq 10$ . Finally, we need the following two families for  $d \in D$ :

If  $\alpha_M(d) - 1 \in K(d)$ , let

$$L_M^d(d) = \{\alpha_M(d) - 1\} \quad \text{and} \quad L_M^d(v) = K_M(v) \text{ for all } v \in X \cup (D \setminus \{d\}).$$

If  $\beta_M(d) + 1 \in K(d)$ , let

$$U_M^d(d) = \{\beta_M(d) + 1\} \quad \text{and} \quad U_M^d(v) = K_M(v) \text{ for all } v \in X \cup (D \setminus \{d\}).$$

For example, if again  $K(d) = \{2, 4, 6, 7, 9, 10\}$ , then  $L_M^d(d)$  is undefined and  $U_M^d(d) = \{7\}$  if  $\delta_M(d) \leq 6$ ,  $L_M^d(d) = \{6\}$  and  $U_M^d(d) = \{10\}$  if  $7 \leq \delta_M(d) \leq 9$ , and  $L_M^d(d) = \{9\}$  and  $U_M^d(d)$  is undefined if  $\delta_M(d) \geq 10$ . Now we are able to formulate Cornuéjols's Theorem in terms of value graphs.

**Theorem 7** (Cornuéjols [10]). *Let  $M$  be a set of edges of the value graph of an  $\mathcal{I}_1$ -constraint over  $(X, D)$  with cardinality sets  $K(d)$  for  $d \in D$  and  $K(x) = \{1\}$  for  $x \in X$ . Then  $M$  has minimum deficiency relative to  $K$  if and only if it has minimum deficiency relative to all of the following mappings:*

1.  $K_M$
2.  $L_M^d$  for  $d \in D$  such that  $\alpha_M(d) - 1 \in K(d)$
3.  $U_M^d$  for  $d \in D$  such that  $\beta_M(d) + 1 \in K(d)$ .

*Moreover, if there is a set  $M'$  of edges that has smaller deficiency than  $M$  relative to one of the sets in 1, 2, and 3, then it also has smaller deficiency than  $M$  relative to  $K$ .*

*Proof (Sketch).* The *only if* direction of the theorem holds since the sets  $K_M(v)$ ,  $L_M^d(v)$ , and  $U_M^d(v)$  are subsets of  $K(v)$  for all  $v \in X \cup D$ . For the *if* direction, assume that there is no set  $M'$  of edges that has deficiency relative to  $K_M$ ,  $L_M^d$ , or  $U_M^d$  smaller than the deficiency of  $M$  relative to  $K$ . By using a result of Lovász [21], it can then be shown that  $M$  has minimum deficiency relative to  $K$ .  $\square$

Recall that an EGC constraint is consistent if and only if it has a solution. Moreover, by the definition of the deficiency of EGC constraints with cardinality sets containing gaps of length at most 1, we know that every solution uniquely corresponds to a set  $M \subseteq X \times D$  of edges of the value graph with deficiency 0 (relative to  $K$ ). Consequently, we can solve such EGC constraints by computing a set  $M$  of edges with minimum deficiency. Theorem 7 allows us to reduce the computation of such a set relative to the mapping  $K$  to the computation relative to the simpler mappings  $K_M$ ,  $L_M^d$ , and  $U_M^d$  for  $d \in D$ .

**Theorem 8.** *Let  $C$  be an  $\mathcal{I}_1$ -constraint over  $(X, D)$  whose value graph has  $m$  edges. In time  $\mathcal{O}(|D|^2 m^{3/2})$  we can decide the consistency of  $C$  and compute a solution if it exists.*

*Proof.* We perform the following steps to compute a set  $M \subseteq X \times D$  of edges of the value graph with minimum deficiency relative to  $K$ :

1. Apply Proposition 2 to find an initial solution  $M$  relative to  $I$ , where  $I(v) = [\alpha(v), \beta(v)]$  for all  $v \in X \cup D$ ; if such an initial solution does not exist, we know that there is no solution relative to  $K$ .
2. Apply Theorem 7 to find a set  $M'$  of edges with smaller deficiency than  $M$  relative to  $K_M$ ,  $L_M^d$ , or  $U_M^d$  for appropriate  $d \in D$ . This can be done by applying Proposition 6 for each of the at most  $2|D| + 1$  cases, since all cardinality sets are in  $\mathcal{I}_1^*$ .
3. Put  $M = M'$  and repeat Step 2 until no such  $M'$  with smaller deficiency can be found. Since the deficiency of  $M$  in our initial solution relative to  $K$  is at most  $|D|$ , there are at most  $|D|$  iterations.

If the deficiency of the resulting set  $M$  relative to  $K$  is 0, we have found a solution; otherwise, we know that there is no solution. Recall that Quimper et al.'s algorithm [25] from Proposition 2 runs in time  $\mathcal{O}(m^{3/2})$ , where  $m$  is the number of edges in the value graph. Thus, the first step can be performed in time  $\mathcal{O}(m^{3/2})$ . Since the size of the gadget graph is linear in  $m$  and since a perfect matching can be found in time  $\mathcal{O}(m^{3/2})$ , we know that the second step can be performed in time  $\mathcal{O}(|D| m^{3/2})$ . Finally, since there are at most  $|D|$  iterations of the second step according to the third step, we obtain our result.  $\square$

Next we consider the domain filtering problem for  $\mathcal{I}_1$ -constraints. Let  $C$  be an  $\mathcal{I}_1$ -constraint over  $(X, D)$  with cardinality sets  $K(d)$  for  $d \in D$  and assume that the value graph of  $C$  has  $m$  edges. We can perform domain filtering in polynomial time by the following two steps:

- (a) Check the consistency of all  $C[x = d]$  for  $x \in X$  and  $d \in D(x)$ .
- (b) Check the consistency of all  $C[d = k]$  for  $d \in D$  and  $k \in K(d)$ .

For each individual check we can use Theorem 8.

Part (a) can be improved as follows: We use Theorem 8 only once to compute a solution  $\tau : X \rightarrow D$  of  $C$  (if no solution exists, then we are done and set all  $D(x)$  to the empty set). Let  $M_\tau = \{(x, \tau(x)) : x \in X\}$  denote the corresponding set of edges of the value graph of  $C$ . By definition,  $M_\tau$  has deficiency 0. Now, for  $x \in X$  and  $d \in D(x) \setminus \{\tau(x)\}$ , take the set  $M_\tau[x = d] = (M_\tau \setminus \{(x, \tau(x))\}) \cup \{(x, d)\}$ .  $M_\tau[x = d]$  is a subset of the edges of the value graph of  $C[x = d]$  and has therefore deficiency at most 2. Thus, at most two applications of Theorem 7 are required to check whether  $C[x = d]$  is consistent, which takes time  $\mathcal{O}(|D| m^{3/2})$ . Hence, part (a) can be carried out in time  $\mathcal{O}(|X| |D|^2 m^{3/2})$ , saving a factor  $|D|$  compared to a repeated application of Theorem 8.

We can use a similar approach to improve part (b): Again, we first take an initial solution  $\tau$ . Consider some  $d \in D$  and let  $k = |\tau^{-1}(d)|$ , i.e., the size of the preimage of  $d$  w.r.t.  $\tau$ . Clearly,  $C[d = k]$  is consistent. Now consider an integer  $k' \in K(d)$  that is next to  $k$ , that is,  $|k - k'| \leq 2$ . To check the consistency of  $C[d = k']$  we can use  $M_\tau$  and apply Theorem 7 at most twice to end up with either a solution  $\tau'$  of  $C[d = k']$  or the result that  $C[d = k']$  is inconsistent. If  $C[d = k']$  has a solution, we can continue similarly using  $M_{\tau'}$ , and so forth, until we have checked all elements in  $K(d)$ . However, if  $C[d = k']$  is inconsistent, then to check the consistency of  $C[d = k'']$  for some  $k'' \in K(d)$  that is next to  $k'$ , we can only use solution  $M_\tau$  and so we need to apply Theorem 7 up to four times. If we proceed in this fashion and get a long series of elements  $k^* \in K(d)$  for which  $C[d = k^*]$  is inconsistent, then there is in the worst case no improvement to a repeated application of Theorem 8.

As discussed earlier, we use a general notion of domain filtering for EGC constraints, where cardinality sets are considered as domains. In this general setting, it is possible that new gaps are introduced or existing gaps are becoming larger (e.g., by domain filtering applied to adjacent constraints in a constraint network). Thus, a tractable instance of an EGC constraint (where all cardinality sets are intervals or have gaps of length at most 1) may become an intractable one. For subsequent domain filtering rounds, we would therefore consider a relaxed version of the EGC constraint where a cardinality set is replaced by a smallest superset having gaps of length at most 1. This keeps the domain filtering polynomial and we still eliminate significantly more domain values compared to the standard domain filtering for such cases based on bounds consistency.

### 3.2 Applying Courcelle's Theorem

Courcelle's Theorem [12] provides a powerful tool for showing that certain graph properties can be checked in linear time for graphs of bounded treewidth. One only needs to define the considered property in terms of a certain formalism (Monadic Second Order Logic, MSO) where one is allowed to quantify over sets of vertices and sets of edges (see, e.g., Downey and Fellows' book [14] for further details and examples). In fact, with Courcelle's Theorem it is easy to establish the following.

**Proposition 9.** *The problems GENFACTOR and EGCC-CONSISTENCY can be decided in linear time for instances having both bounded treewidth and bounded cardinality-width.*

Let us sketch the proof. Note that we only need to consider GENFACTOR since it contains EGCC-CONSISTENCY as a special case. Let  $G = (V, E)$  with cardinality sets  $K(v)$ ,  $v \in V$ , be an instance of GENFACTOR with cardinality-width  $m$ . We may assume, w.l.o.g., that all vertices have degree at least 2, as vertices  $v$  of degree 0 or 1 can be easily eliminated: If  $K(v) = \emptyset$ , the instance is unsatisfiable, and if  $K(v) = \{0\}$ , we can just remove  $v$ . Otherwise,

if  $K(v) = \{1\}$  or  $K(v) = \{0, 1\}$ , we can remove  $v$  and update the cardinality set  $K(w)$  of the single neighbor  $w$  of  $v$  by  $K'(w)$ . In particular, we put  $K'(w) = \{i - 1 : i \in K(w), i > 0\}$  if  $K(v) = \{1\}$  and we put  $K'(w) = K(w) \cup \{i - 1 : i \in K(w), i > 0\}$  if  $K(v) = \{0, 1\}$ .

Now, let  $K_1, \dots, K_{2m+1}$  be an enumeration of all subsets of  $\{0, \dots, m\}$ . We assign to every vertex  $v$  of  $G$  an integer  $i(v)$  such that  $K(v) = K_{i(v)}$ , and we extend  $G$  in the following way: for every vertex  $v$  we introduce new vertices  $v_1, \dots, v_{i(v)}$  and join each of them with  $v$  by an edge, i.e.,  $v$  gets  $i(v)$  new neighbors with degree 1. Let  $G'$  denote the graph obtained from  $G$  in this way. Since the added vertices have degree 1, we can distinguish them from the old vertices. The new vertices allow us to reconstruct the cardinality sets  $K(v)$ . Since  $m$  is a constant, we can define predicates  $P_1, \dots, P_{2m+1}$  such that  $P_i(v)$  is true for a vertex of  $G'$  if and only if  $v$  is of degree at least 2 and has exactly  $i(v)$  neighbors of degree 1 (equivalently,  $v$  belongs to  $G$  and  $K(v) = K_{i(v)}$ ). It is now easy to state an MSO sentence  $\varphi$  that holds on  $G'$  if and only if  $G$  has a general factor that meets the cardinality conditions imposed by the sets  $K(v)$ . Now assume  $G$  has treewidth bounded by a constant  $k$ . Observe that the treewidth of  $G'$  is then also bounded by  $k$ . Hence, by Courcelle's Theorem, we can check in linear time whether  $\varphi$  holds on  $G'$ , and thus the proposition follows.

In the above construction it was essential that the cardinality-width is bounded, since otherwise we could not confine ourselves to a finite number of predicates  $P_i$ . The question arises whether this limitation can be overcome by a different, more sophisticated approach. We will return to this question in Section 6, where we will provide a negative answer. Namely we will show that EGCC-CONSISTENCY, parameterized by the treewidth alone, is complete for the parameterized complexity class W[1]. Hence one cannot expect the existence of an algorithm that solves EGCC-CONSISTENCY (or GENFACTOR) for instances of bounded treewidth within a runtime that is bounded by a polynomial of order independent of the treewidth.

Algorithms obtained via Courcelle's Theorem are impractical as the linear runtime involves a huge hidden factor. Therefore we propose in the next section an efficient combinatorial algorithm based on dynamic programming.

## 4 Efficient Consistency Checking

In the following we consider an EGC constraint  $C$  over  $(X, D)$  together with a nice tree decomposition  $(T, \chi, r)$  of the value graph of  $C$ . Let  $m$  denote the cardinality-width of  $C$ . For every node  $t$  of  $T$  let  $\text{var}(t)$  denote the set of variables in  $\chi(t)$  and let  $\text{val}(t)$  denote the set of values in  $\chi(t)$ . We define  $\text{var}^*(t)$  as the union of  $\text{var}(t')$  for tree nodes  $t'$  that belong to the subtree rooted at  $t$ ;  $\text{val}^*(t)$  is defined similarly. Thus,  $\text{var}^*(t) \setminus \text{var}(t)$  and  $\text{val}^*(t) \setminus \text{val}(t)$  are the sets of variables and values, respectively, that are already ‘‘forgotten’’ at tree node  $t$ ; similarly  $X \setminus \text{var}^*(t)$  and  $D \setminus \text{val}^*(t)$  are the sets of variables and values, respectively, that are ‘‘not yet introduced’’ at tree node  $t$ .

With every tree node  $t$  we associate the set  $A(t)$  of partial assignments  $\tau : X_\tau \rightarrow \text{val}^*(t)$  defined on sets  $X_\tau \subseteq \text{var}^*(t)$  of variables such that

1.  $\tau(x) \in D(x)$  for all  $x \in X_\tau$  ( $\tau$  respects domains),
2.  $\text{var}^*(t) \setminus \text{var}(t) \subseteq X_\tau$  ( $\tau$  is defined for all forgotten variables), and
3.  $|\{x \in \text{var}^*(t) : \tau(x) = d\}| \in K(d)$  for all  $d \in \text{val}^*(t) \setminus \text{val}(t)$  ( $\tau$  respects cardinality sets for forgotten values).

A *record* of a tree node  $t$  is a mapping

$$R : \chi(t) \rightarrow \text{val}(t) \cup \{\sqcup, \star\} \cup \{0, 1, 2, \dots, m\}$$

such that the following two conditions are satisfied:

1.  $R(x) \in (\text{val}(t) \cap D(x)) \cup \{\sqcup, \star\}$  for  $x \in \text{var}(t)$ ;
2.  $R(d) \in \{0, 1, 2, \dots, \max(K(d))\}$  for  $d \in \text{val}(t)$ .

We use records to represent the partial assignments  $\tau$  in the set  $A(t)$ :  $R(x) = \sqcup$  means that  $\tau$  is not defined for  $x$ , and  $R(x) = \star$  means that  $\tau$  maps  $x$  to a value that is already forgotten. More specifically, we say that a record  $R$  of a tree node  $t$  *represents* an assignment  $\tau \in A(t)$  if the following two conditions hold:

1. For all  $x \in \text{var}(t)$

$$R(x) = \begin{cases} \tau(x) & \text{if } \tau(x) \in \text{val}(t), \\ \star & \text{if } \tau(x) \in \text{val}(t)^* \setminus \text{val}(t), \\ \sqcup & \text{otherwise (i.e., if } x \in \text{var}(t) \setminus X_\tau); \end{cases}$$

2. for all  $d \in \text{val}(t)$

$$R(d) = |\{x \in \text{var}^*(t) : \tau(x) = d\}|.$$

Note that in general a record can represent an unbounded number of assignments, and every assignment in  $A(t)$  is represented by exactly one record  $R$  of  $t$ . We say that a record  $R$  of  $t$  is *valid* if it represents some assignment  $\tau \in A(t)$ .

**Lemma 10.**  *$C$  is consistent if and only if there is a valid record  $R$  of the root  $r$  such that  $R(x) \neq \sqcup$  for all  $x \in \text{var}(r)$  and  $R(d) \in K(d)$  for all  $d \in \text{val}(r)$ .*

*Proof.* If  $C$  is consistent, there exists an assignment  $\tau : X \rightarrow D$  such that  $\tau(x) \in D(x)$  for all  $x \in X$  and  $|\{x \in X : \tau(x) = d\}| \in K(d)$  for all  $d \in D$ . In particular, this implies that  $\tau \in A(r)$ . Since  $X_\tau = X$ , there must be a record  $R$  of the root  $r$  such that  $R(x) = \tau(x) \in D(x)$  and  $R(x) = \star$  if  $\tau(x) \in \text{val}(r)$  and  $\tau(x) \in \text{val}^*(r) \setminus \text{val}(r)$ , respectively, for all  $x \in \text{var}(r)$ , and  $R(d) = |\{x \in X : \tau(x) = d\}| \in K(d)$  for all  $d \in \text{val}(r)$ . Hence,  $R$  is valid,  $R(x) \neq \sqcup$  for all  $x \in \text{var}(r)$ , and  $R(d) \in K(d)$  for all  $d \in \text{val}(r)$ .

Conversely, let  $R$  be a valid record of the root  $r$  such that  $R(x) \neq \sqcup$  for all  $x \in \text{var}(r)$  and  $R(d) \in K(d)$  for all  $d \in \text{val}(r)$ . Since  $R$  is valid, there must be an  $\tau \in A(r)$  representing  $R$  such that  $|\{x \in X : \tau(x) = d\}| = R(d) \in K(d)$  for all  $d \in \text{val}(r)$ . Moreover, since  $R(x) \neq \sqcup$  for all  $x \in \text{var}(r)$ , we know that  $\text{var}(r) \setminus X_\tau = \emptyset$ , that is,  $X_\tau = \text{var}^*(r) = X$  and thus  $\tau : X \rightarrow D$ . Hence,  $\tau(x) \in D(x)$  for all  $x \in X$  and  $|\{x \in X : \tau(x) = d\}| \in K(d)$  for all  $d \in D$ , that is,  $C$  is consistent.  $\square$

The next five lemmas will allow us to compute the valid records of a tree node from the valid records of its children.

**Lemma 11** (Join nodes). *Let  $t_1, t_2$  be the children of  $t$ . A record  $R$  of  $t$  is valid if and only if there are valid records  $R_1$  and  $R_2$  of  $t_1$  and  $t_2$ , respectively, such that the following holds:*

1. for all  $x \in \text{var}(t)$  one of the following holds

- (a)  $R(x) = R_1(x) = R_2(x) \in D \cup \{\sqcup\}$ ;

- (b)  $R(x) = R_1(x) = \star$  and  $R_2(x) = \sqcup$ ;

- (c)  $R(x) = R_2(x) = \star$  and  $R_1(x) = \sqcup$ ;

2.  $R(d) = R_1(d) + R_2(d) - |\{x \in \text{var}(t) : R(x) = d\}|$  for all  $d \in \text{val}(t)$ .

*Proof.* Let  $R$  be a valid record of  $t$ . By definition,  $R$  represents an assignment  $\tau \in A(t)$ . For  $i = 1, 2$ , let  $\tau_i = \tau|_{\text{var}^*(t_i)}$  be the restriction of  $\tau$  to  $\text{var}^*(t_i)$ . Since  $\tau \in A(t)$ , it can be easily verified that  $\tau_i \in A(t_i)$ ,  $i = 1, 2$ . Let  $R_i$  be the (valid) record of  $t_i$  that represents  $\tau_i$ ,  $i = 1, 2$ . It remains to check that the properties stated in the lemma hold. Let  $x \in \text{var}(t)$ .

If  $R_1(x), R_2(x) \in D(x) \cup \{\perp\}$ , then  $R(x) = R_1(x) = R_2(x)$ , since  $\text{var}(t) = \text{var}(t_1) = \text{var}(t_2)$  and  $X_\tau \cap \text{var}(t) = X_{\tau_1} \cap \text{var}(t_1) = X_{\tau_2} \cap \text{var}(t_2)$ . If  $R_1(x) = \star$  and  $R_2(x) = \perp$ , then  $\tau_1(x) \in \text{val}^*(t_1) \setminus \text{val}(t_1)$  and  $x \notin X_{\tau_2}$ ; hence  $\tau(x) \in \text{val}^*(t) \setminus \text{val}(t)$ , and so  $R(x) = \star$ . Symmetrically, if  $R_1(x) = \perp$  and  $R_2(x) = \star$ , then  $R(x) = \star$ . Thus, the first property of the lemma holds. It is easy to see that  $R(d) = R_1(d) + R_2(d) - |\{x \in \text{var}(t) : R(x) = d\}|$  for all  $d \in \text{val}(t)$ , hence the second property holds as well.

Conversely, let  $R$  be a record of  $t$ , and assume that there are valid records  $R_1$  and  $R_2$  of  $t_1$  and  $t_2$ , respectively, such that the two properties stated in the lemma hold. Let  $\tau_1 \in A(t_1)$  and  $\tau_2 \in A(t_2)$  be assignments that are represented by  $R_1$  and  $R_2$ , respectively. By the connectedness condition of a tree decomposition, it follows that the sets  $\text{var}^*(t_1) \setminus \text{var}(t_1)$  and  $\text{var}^*(t_2) \setminus \text{var}(t_2)$  are disjoint. Hence, we can combine  $\tau_1$  and  $\tau_2$  to an assignment  $\tau : X_\tau \rightarrow D$ , defined for the set  $X_\tau = X_{\tau_1} \cup X_{\tau_2}$ , with  $\tau(x) \in D(x)$  for all  $x \in X_\tau$ . It is easy to check that  $\tau$  corresponds to  $R$ , hence  $R$  is a valid record of  $t$ .  $\square$

**Lemma 12** (Introduce variable). *Let  $t$  be an introduce node with child  $t'$  such that  $\text{var}(t) = \text{var}(t') \cup \{x_0\}$  and  $\text{val}(t) = \text{val}(t')$ . A record  $R$  of  $t$  is valid if and only if there is a valid record  $R'$  of  $t'$  such that  $R'(x) = R(x)$  for all  $x \in \text{var}(t')$ , and one of the following holds:*

1.  $R(x_0) = \perp$  and  $R(d) = R'(d)$  for all  $d \in \text{val}(t)$ ,
2.  $R(x_0) = d_0$  for some  $d_0 \in \text{val}(t) \cap D(x_0)$  such that  $R(d_0) = R'(d_0) + 1$  and  $R(d) = R'(d)$  for all  $d \in \text{val}(t) \setminus \{d_0\}$ ,

*Proof.* Let  $R$  be a valid record of  $t$ . By definition,  $R$  represents an assignment  $\tau \in A(t)$ . Let  $\tau' = \tau|_{\text{var}^*(t')}$  be the restriction of  $\tau$  to  $\text{var}^*(t')$ . Since  $\tau \in A(t)$ , it can be easily verified that  $\tau' \in A(t')$ . Let  $R'$  be the (valid) record of  $t'$  that represents  $\tau'$ . It remains to check that the properties stated in the lemma hold. Let  $x \in \text{var}(t)$ . If  $x \in \text{var}(t')$ , then  $R(x) = R'(x)$ , since  $\text{var}(t') = \text{var}(t) \setminus \{x_0\}$  and  $X_\tau \cap \text{var}(t') = X_{\tau'} \cap \text{var}(t')$ . For  $x_0 \in \text{var}(t) \setminus \text{var}(t')$ , either  $R(x_0) = \perp$  or  $R(x_0) = d_0$  for some  $d_0 \in \text{val}(t) \cap D(x_0)$ . It is thus easy to see that one of the properties of the lemma must hold.

Conversely, let  $R$  be a record of  $t$ , and assume that there is a valid record  $R'$  of  $t'$  such that the properties stated in the lemma hold. Let  $\tau' \in A(t')$  be an assignment represented by  $R'$ . If  $R(x_0) = d_0$  for some  $d_0 \in \text{val}(t) \cap D(x_0)$ , we put  $\tau = \tau' \cup \{(x_0, d_0)\}$ ; otherwise, we put  $\tau = \tau'$ . It is easy to check that  $\tau$  corresponds to  $R$ , hence  $R$  is a valid record of  $t$ .  $\square$

**Lemma 13** (Introduce value). *Let  $t$  be an introduce node with child  $t'$  such that  $\text{val}(t) = \text{val}(t') \cup \{d_0\}$  and  $\text{var}(t) = \text{var}(t')$ . A record  $R$  of  $t$  is valid if and only if there is a valid record  $R'$  of  $t'$  such that the following holds:*

1. for all  $x \in \text{var}(t)$

$$R(x) = \begin{cases} d \in \{\perp\} \cup (\{d_0\} \cap D(x)) & \text{if } R'(x) = \perp; \\ R'(x) & \text{otherwise;} \end{cases}$$

2. for all  $d \in \text{val}(t)$

$$R(d) = \begin{cases} |\{x \in \text{var}(t) : R(x) = d_0\}| & \text{if } d = d_0; \\ R'(d) & \text{otherwise;} \end{cases}$$

*Proof.* Let  $R$  be a valid record of  $t$ . By definition,  $R$  represents an assignment  $\tau \in A(t)$ . Let  $X = \{x \in \text{var}(t) : R(x) = d_0\}$ , and let  $\tau' = \tau|_{\text{var}^*(t) \setminus X}$  be the restriction of  $\tau$  to  $\text{var}^*(t) \setminus X$ . Since  $\tau \in A(t)$ , it can be easily verified that  $\tau' \in A(t')$ . Let  $R'$  be the (valid) record of  $t'$  that represents  $\tau'$ . It remains to check that the properties stated in the lemma hold. If  $x \in \text{var}(t) \setminus X$ , then  $R(x) = R'(x)$ , since  $\text{var}(t) = \text{var}(t')$  and  $X_\tau \cap (\text{var}(t) \setminus X) = X_{\tau'} \cap \text{var}(t')$ . Otherwise, if  $x \in X$ , we know that  $R'(x) = \perp$  and  $R(x) = d_0$ . Thus, the first property of the

lemma holds. It is easy to see that  $R(d_0) = |X|$  and  $R(d) = R'(d)$  for all  $d \in \text{val}(t')$ , hence the second property holds as well.

Conversely, let  $R$  be a record of  $t$ , and assume that there is a valid record  $R'$  of  $t'$  such that the properties stated in the lemma hold. Let  $\tau' \in A(t')$  be an assignment represented by  $R'$ . We put  $\tau = \tau' \cup \{(x, d_0) : x \in \text{var}(t), R(x) = d_0\}$ . It is easy to check that  $\tau$  corresponds to  $R$ , hence  $R$  is a valid record of  $t$ .  $\square$

**Lemma 14** (Forget variable). *Let  $t$  be a forget node with child  $t'$  such that  $\text{var}(t) = \text{var}(t') \setminus \{x_0\}$  and  $\text{val}(t) = \text{val}(t')$ . A record  $R$  of  $t$  is valid if and only if there is a valid record  $R'$  of  $t'$  such that  $R'(x_0) \neq \sqcup$  and  $R(z) = R'(z)$  for all  $z \in \text{var}(t) \cup \text{val}(t)$ .*

*Proof.* Let  $R$  be a valid record of  $t$ . By definition,  $R$  represents an assignment  $\tau \in A(t)$ . Since  $\text{var}^*(t) = \text{var}^*(t')$ , it can be easily verified that also  $\tau \in A(t')$ . Let  $R'$  be the (valid) record of  $t'$  that represents  $\tau$ . Since  $x_0 \in \text{var}^*(t) \setminus \text{var}(t) \subseteq X_\tau$ , we know that  $R'(x_0) \neq \sqcup$ . It is thus easy to see that the properties stated in the lemma hold.

Conversely, let  $R$  be a record of  $t$ , and assume that there is a valid record  $R'$  of  $t'$  such that the properties stated in the lemma hold. Let  $\tau' \in A(t')$  be an assignment represented by  $R'$ . It is easy to check that  $\tau'$  also corresponds to  $R$ , hence  $R$  is a valid record of  $t$ .  $\square$

**Lemma 15** (Forget value). *Let  $t$  be a forget node with child  $t'$  such that  $\text{val}(t) = \text{val}(t') \setminus \{d_0\}$  and  $\text{var}(t) = \text{var}(t')$ . A record  $R$  of  $t$  is valid if and only if there is a valid record  $R'$  of  $t'$  such that the following holds:*

1.  $R'(d_0) \in K(d_0)$ ;
2. for all  $x \in \text{var}(t)$  we have

$$R(x) = \begin{cases} \star & \text{if } R'(x) = d_0; \\ R'(x) & \text{otherwise;} \end{cases}$$

3.  $R(d) = R'(d)$  for all  $d \in \text{val}(t) = \text{val}(t') \setminus \{d_0\}$ .

*Proof.* Let  $R$  be a valid record of  $t$ . By definition,  $R$  represents an assignment  $\tau \in A(t)$ . Since  $\text{var}^*(t) = \text{var}^*(t')$ , it can be easily verified that also  $\tau \in A(t')$ . Let  $R'$  be the (valid) record of  $t'$  that represents  $\tau$ . It remains to check that the properties stated in the lemma hold. Since  $d_0 \in \text{val}^*(t) \setminus \text{val}(t)$ , we know that  $R'(d_0) = |\{x \in \text{var}^*(t) : \tau(x) = d_0\}| \in K(d_0)$  and  $R(x) = \star$  for  $x \in \text{var}(t)$  with  $R'(x) = d_0$ . It is easy to see that  $R(x) = R'(x)$  for all other  $x \in \text{var}(t)$ , and that  $R(d) = R'(d)$  for all  $d \in \text{val}(t)$ . Thus, the properties of the lemma hold.

Conversely, let  $R$  be a record of  $t$ , and assume that there is a valid record  $R'$  of  $t'$  such that the properties stated in the lemma hold. Let  $\tau' \in A(t')$  be an assignment represented by  $R'$ . It is easy to check that  $\tau'$  also corresponds to  $R$ , hence  $R$  is a valid record of  $t$ .  $\square$

**Theorem 16.** EGCC-CONSISTENCY can be decided in polynomial time for instances having bounded treewidth and in linear time for instances having both bounded treewidth and bounded cardinality-width.

*Proof.* Let  $C$  be an EGC constraint over  $(X, D)$ . Assume that the treewidth of the value graph of  $C$  is bounded by a constant  $k$  and that the cardinality-width of  $C$  is  $m$ . Let  $(T, \chi, r)$  be a nice tree decomposition of the value graph of  $C$  such that the width of the tree decomposition is at most  $k$  and  $T$  has  $\mathcal{O}(n)$  nodes, where  $n = |X| + |D|$  denotes the number of vertices of the value graph of  $C$ . Such a tree decomposition can be found in time  $\mathcal{O}(n)$  (see the discussion in Section 2.2).

With every tree node  $t$  of  $T$  we associate the set  $M(t)$  of valid records of  $t$ . We can compute the sets  $M(t)$  via a bottom-up traversal of  $T$  as follows. For a leaf node  $t$  we can compute  $M(t)$  just by considering all possible rows  $R$  with  $R(x) \in \text{val}(t) \cup \{\sqcup\}$  for  $x \in \text{var}(t)$

and  $R(d) = |\{x \in \text{var}(t) : R(x) = d\}|$  for  $d \in \text{val}(t)$ . Let  $\omega = \max(k+2, m+1)$ ; the number of records at node  $t$  is at most

$$\begin{aligned} |\text{val}(t) \cup \{\sqcup, \star\}|^{|\text{var}(t)|} \cdot (m+1)^{|\text{val}(t)|} &= \\ (k+3 - |\text{var}(t)|)^{|\text{var}(t)|} \cdot (m+1)^{k+1-|\text{var}(t)|} &\leq \\ (\omega+1 - |\text{var}(t)|)^{|\text{var}(t)|} \cdot \omega^{k+1-|\text{var}(t)|} &= \\ ((\omega+1 - |\text{var}(t)|)/\omega)^{|\text{var}(t)|} \cdot \omega^{k+1} &\leq \omega^{k+1}, \end{aligned}$$

i.e., bounded purely in terms of  $k$  and  $m$ . Lemmas 11–15 ensure that for computing  $M(t)$  of a non-leaf node  $t$  we only need to know the sets  $M(t')$  of the children  $t'$  of  $t$ : For a join node  $t$  with children  $t_1$  and  $t_2$  we compute  $M(t)$  by combining all pairs of records  $R_1 \in M(t_1)$ ,  $R_2 \in M(t_2)$ , and by checking the conditions of Lemma 11. The time required for each pair is bounded in terms of  $k$  and  $m$ . Hence, given  $M(t_1)$  and  $M(t_2)$ , we can compute  $M(t)$  in time  $\mathcal{O}(\max(k+2, m+1)^{k+1})$ . Computing the sets  $M(t)$  for introduce and forget nodes  $t$  according to Lemmas 12–15 is even simpler. Hence we can compute the sets  $M(t)$  for all tree nodes  $t$  in time  $\mathcal{O}(n \cdot \max(k+2, m+1)^{k+1})$ . According to Lemma 10 we can decide consistency of  $C$  by examining the records in  $M(r)$  at the root  $r$ . Hence, since  $k$  is assumed to be a constant, we get a polynomial-time algorithm (that is not a fixed-parameter algorithm though, since the degree of the polynomial depends on  $k$ ), and we get a linear-time (and thus fixed-parameter) algorithm in case where also the cardinality-width is bounded.  $\square$

Figure 4 shows a nice tree decomposition of the value graph in Figure 1 together with the sets  $M(t)$  as computed according to the proof of Theorem 16. Records are specified as table rows (the meaning of table rows with gray backgrounds will be discussed in the next section).

## 5 Efficient Domain Filtering

Consider an EGC constraint  $C$  over  $(X, D)$ .  $C$  is domain consistent (recall the definition in Section 2.1) if and only if all constraints  $C[x = d]$  for  $x \in X$  and  $d \in D(x)$  and all constraints  $C[d = k]$  for  $d \in D$  and  $k \in K(d)$  are consistent. Since the treewidth of  $C[x = d]$  and the treewidth of  $C[d = k]$  are both bounded by the treewidth of  $C$ , domain filtering can be carried out by applying Theorem 16  $\mathcal{O}(|X| \cdot |D|)$  times. In the following we show how to extend the dynamic programming approach to accomplish domain filtering more efficiently, even in linear time if treewidth and cardinality-width are bounded.

As in the previous section, let  $C$  be an EGC constraint over  $(X, D)$ . Assume that the treewidth of the value graph of  $C$  is bounded by a constant  $k$  and that the cardinality-width of  $C$  is  $m$ . Let  $(T, \chi, r)$  be a nice tree decomposition of the value graph of  $C$  such that the width of the tree decomposition is at most  $k$  and  $T$  has  $\mathcal{O}(n)$  nodes, where  $n = |X| + |D|$  denotes the number of vertices of the value graph of  $C$ .

We call a record  $R$  of a tree node  $t$  *solution-valid* if  $R$  represents the restriction  $\tau|_{\text{var}^*(t)}$  of a satisfying assignment  $\tau : X \rightarrow D$  of  $C$  to  $\text{var}^*(t)$ . Note that every solution-valid record is valid. The following lemma is a direct consequence of the definitions (recall from the end of Section 2.2 the notion of “final node”).

**Lemma 17.**  *$C$  is domain consistent if and only if*

1. *for every pair  $x, d$  with  $x \in X$  and  $d \in D(x)$ , there exists a solution-valid record  $R$  of some tree node  $t$  with  $R(x) = d$ , and*
2. *for every pair  $d, j$  with  $d \in D$  and  $j \in K(d)$  there exists a solution-valid record  $R$  at the final node of  $d$  such that  $R(d) = j$ .*

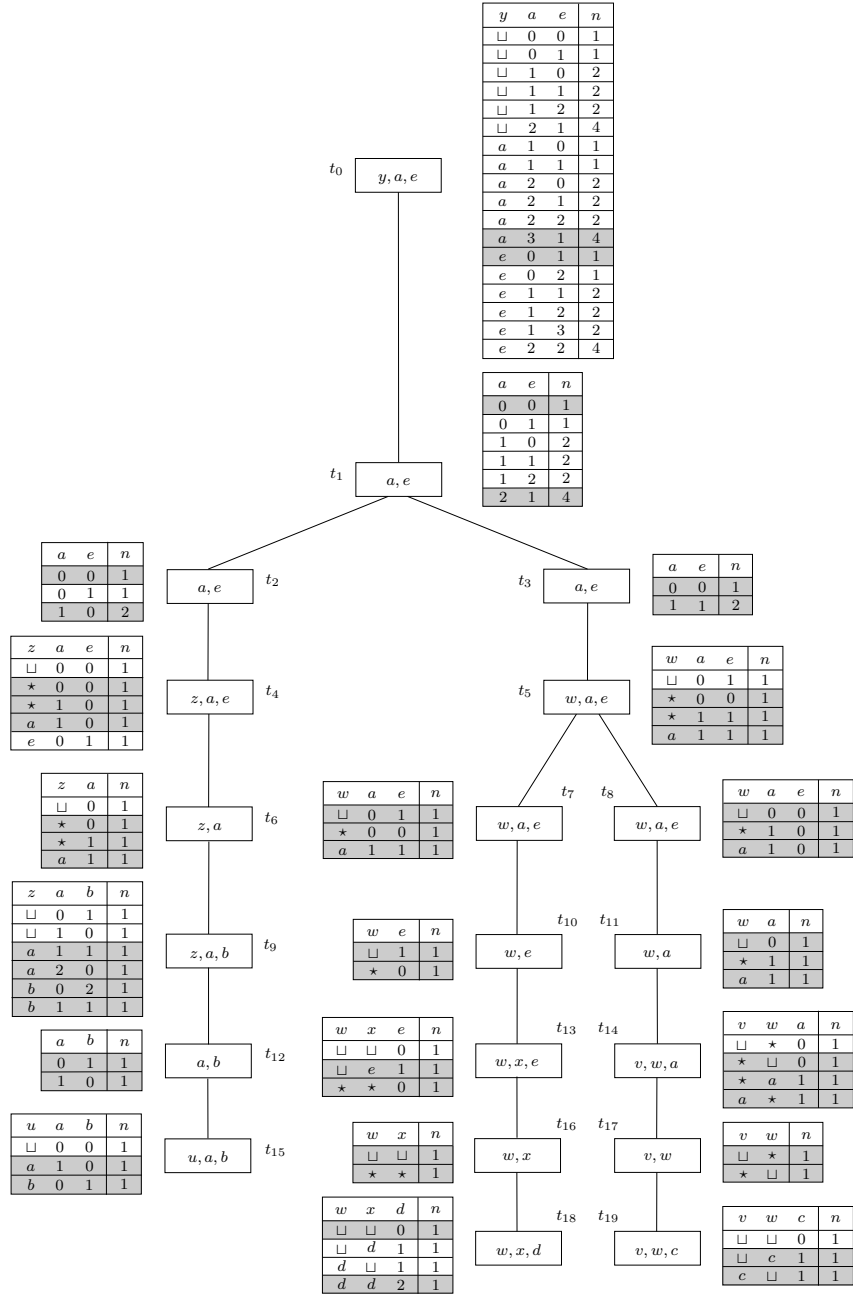


Figure 4: A nice tree decomposition of the value graph in Figure 1 with tables representing valid records. Rows with gray background indicate the result after domain filtering. The columns  $n$  illustrate that our algorithm can be easily extended to even count the number of solutions.

Hence, if we have computed all solution-valid records of all tree nodes, then we have solved the domain filtering task. With every tree node  $t$  of  $T$  we associate the set  $M(t)$  of valid records and the set  $M'(t) \subseteq M(t)$  of solution-valid records of  $t$ . The sets  $M(t)$  are computed in linear time by the algorithm described in the proof of Theorem 16. Next we describe how we can compute the subsets  $M'(t)$  by means of a top-down traversal of  $T$ . This process is illustrated in Figure 4 where solution-valid records are indicated as table rows with gray background.

For the root  $r$  we can easily compute  $M'(r)$  from  $M(r)$  since, according to Lemma 10, a valid record  $R$  at  $r$  is solution-valid if and only if  $R(x) \neq \perp$  for all  $x \in \text{var}(r)$  and  $R(d) \in K(d)$  for all  $d \in \text{val}(r)$ .

Consider a join node  $t$  with children  $t_1, t_2$ , and assume that we have already computed the set  $M'(t)$ . It follows from Lemma 11 that a valid record  $R_1$  of  $t_1$  is solution-valid if and only if there is a solution-valid record  $R$  of  $t$  and a valid record  $R_2$  of  $t_2$  such that for the records  $R, R_1, R_2$  the properties stated in Lemma 11 hold. Hence, we can compute the sets  $M'(t_1)$  and  $M'(t_2)$  from  $M'(t)$  in time that only depends on  $k$  and  $m$ . Similarly, for an introduce or forget node  $t$  with child  $t'$ , a record  $R'$  of  $t'$  is solution-valid if and only if there is a solution-valid record  $R$  of  $t$  such that the properties stated in one of the Lemmas 12–15 hold. Thus if we know  $M'(t)$  we can compute  $M'(t')$  in time that only depends on  $k$  and  $m$ .

Since  $T$  has  $\mathcal{O}(n)$  many nodes, we have shown the following result.

**Theorem 18.** *Domain filtering for extended global cardinality constraints can be carried out in polynomial time for instances having bounded treewidth and in linear time for instances having both bounded treewidth and bounded cardinality-width.*

## 6 W[1]-Hardness for Parameter Treewidth

We return to the question raised at the end of Section 3 of whether bounding the cardinality-width is dispensable in Proposition 9. The framework of parameterized complexity [14] offers concepts and tools for answering this question. Let us briefly review the main concepts of this framework; for an in-depth treatment we refer to other sources [14, 15, 23].

In parameterized complexity one considers problems in two dimensions: one dimension is the usual size  $n$  of the instance and the second dimension is the parameter (usually a positive integer  $k$ ). A parameterized problem is called *fixed-parameter tractable* if it can be solved in time  $\mathcal{O}(f(k) \cdot n^c)$  for some computable function  $f$  and constant  $c$  that is independent of the parameter. FPT denotes the class of fixed-parameter tractable decision problems. The parameterized complexity classes  $\text{W}[1] \subseteq \text{W}[2] \subseteq \dots \subseteq \text{W}[\text{P}]$  contain problems that are believed to be not fixed-parameter tractable [14]; all inclusions are believed to be proper. There are different kinds of evidence for assuming that  $\text{FPT} \neq \text{W}[1]$ . For example,  $\text{FPT} = \text{W}[1]$  would imply that the Exponential Time Hypothesis fails (cf. Flum and Grohe [15]). A parameterized problem  $P$  *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  $\mathcal{O}(f(k) \cdot |x|^c)$  ( $f, g$  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  $Q$ .

The CLIQUE problem asks whether, given a graph  $G$  and an integer  $k$ ,  $G$  contains a complete subgraph on  $k$  vertices. CLIQUE (with parameter  $k$ ) is a W[1]-complete problem [14]. Below we shall use the special case of the problem where the vertex set of the given graph is partitioned into  $k$  independent sets. As observed by Pietrzak [24], CLIQUE remains W[1]-complete under this restriction. This follows by the following reduction. Given  $G = (V, E)$  and  $k$ , take disjoint copies  $V_1, \dots, V_k$  of  $V$ ; let  $v_i$  denote the copy of  $v$  in  $V_i$ . We construct the graph  $G' = (V', E')$  with  $V' = \bigcup_{i=1}^k V_i$  and  $E' = \{u_i v_j : uv \in E, 1 \leq i < j \leq k\}$ . Now it is easy to verify that  $G$  has a clique on  $k$  vertices if and only if  $G'$  has a clique on  $k$  vertices.

**Theorem 19.** EGCC-CONSISTENCY parameterized by the treewidth of the value graph is  $W[1]$ -hard.

*Proof.* We give a reduction from CLIQUE. Consider an instance consisting of a graph  $G = (V, E)$  and an integer  $k$ . As discussed above, we may assume that  $V$  is partitioned into independent sets  $V_1, \dots, V_k$ . Let  $V_i = \{v_i^1, \dots, v_i^{n_i}\}$  and let  $N = \max_{i=1}^k n_i + 1$ .

We will construct an EGC constraint  $C$  such that  $C$  is consistent if and only if  $G$  has a clique on vertices  $v_1^{j[1]}, \dots, v_k^{j[k]}$  with  $j[i] \in \{1, \dots, n_i\}$ . The general idea is that  $C$  consists of  $k$  parts  $P_1, \dots, P_k$  where the  $i$ -th part encodes the selection of  $v_i^{j[i]}$  from  $V_i$ . Any two parts  $P_i$  and  $P_{i'}$  are connected via a value  $d_{\{i, i'\}}$ . Assume w.l.o.g.  $i < i'$ . If  $P_i$  selects vertex  $v_i^{j[i]}$ , it instantiates  $j[i]$  many variables with value  $d_{\{i, i'\}}$ ; if  $P_{i'}$  selects vertex  $v_{i'}^{j[i']}$ , it instantiates  $N \cdot j[i']$  many variables with value  $d_{\{i, i'\}}$ . Now  $K(d_{\{i, i'\}})$  is defined to contain exactly the integers  $j[i] + N \cdot j[i']$  such that  $v_i^{j[i]}$  and  $v_{i'}^{j[i']}$  are adjacent in  $G$ . Since  $j[i] < N$  and  $j[i'] < N$ , each integer in  $K(d_{\{i, i'\}})$  corresponds uniquely to a certain pair  $(j[i], j[i'])$ .

More specifically, the constraint  $C$  is defined as follows. For every  $1 \leq i \leq k$  we introduce a value  $d_i$ . For every  $1 \leq i \leq k$  and  $1 \leq j \leq n_i$  we introduce a variable  $x_i^j$  and a value  $d_i^j$ . For every  $i, i' \in \{1, \dots, k\}$ ,  $i \neq i'$ , and  $1 \leq j \leq n_i$  we introduce a set  $X_{i, i'}^j$  of variables such that

$$|X_{i, i'}^j| = \begin{cases} j & \text{if } i < i'; \\ N \cdot j & \text{otherwise.} \end{cases}$$

Finally, for every  $1 \leq i < i' \leq k$  we introduce a value  $d_{\{i, i'\}}$ . Domains and cardinality sets are defined as follows:

$$\begin{aligned} D(x_i^j) &= \{d_i, d_i^j\} \\ D(x) &= \{d_i^j, d_{\{i, i'\}}\} \quad \text{for } x \in X_{i, i'}^j \\ K(d_i) &= \{1\} \\ K(d_i^j) &= \{0, 1 + \sum_{i' \in \{1, \dots, k\} \setminus \{i\}} |X_{i, i'}^j|\} \\ K(d_{\{i, i'\}}) &= \{|X_{i, i'}^j| + |X_{i', i}^{j'}| : v_i^j v_{i'}^{j'} \in E, \\ &\quad 1 \leq j \leq n_i, 1 \leq j' \leq n_{i'}\}. \end{aligned}$$

This completes the construction of  $C$ ; see Figure 5 for an illustration.

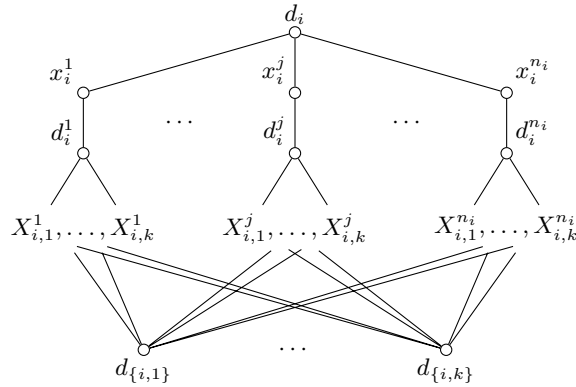


Figure 5: The  $i$ -th part of the value graph of the constraint constructed in the proof of Theorem 19.

*Claim 1:*  $C$  is consistent if and only if  $G$  has a clique of size  $k$ .

Assume that  $S = \{v_1^{j[1]}, \dots, v_k^{j[k]}\}$  induces a clique in  $G$ . We define an assignment  $\tau$  for  $C$  as follows. We put

$$\tau(x_i^j) = \begin{cases} d_i & \text{if } j[i] = j; \\ d_i^j & \text{otherwise} \end{cases}$$

and for  $x \in X_{i,i'}^j$  we put

$$\tau(x) = \begin{cases} d_{\{i,i'\}} & \text{if } j[i] = j; \\ d_i^j & \text{otherwise.} \end{cases}$$

It can be easily checked that  $\tau$  satisfies  $C$ .

Conversely, let  $\tau$  be an assignment that satisfies  $C$ . For every  $i \in \{1, \dots, k\}$  there is exactly one  $j \in \{1, \dots, n_i\}$  with  $\tau(x_i^j) = d_i$ , since  $K(d_i) = \{1\}$ . Let  $S = \{v_1^{j[1]}, \dots, v_k^{j[k]}\}$ . We show that  $S$  induces a clique in  $G$ . To this aim, choose  $1 \leq i < i' \leq k$  arbitrarily. It follows from the definition of  $C$  that the variables mapped to  $d_{\{i,i'\}}$  are exactly the variables in the sets  $X_{i,i'}^{j[i]}$  and  $X_{i,i'}^{j[i']}$ . We have  $|X_{i,i'}^{j[i]}| = j[i]$ ,  $|X_{i,i'}^{j[i']}$  =  $N \cdot j[i']$ , and  $j[i] + N \cdot j[i'] \in K(d_{\{i,i'\}})$ . Since  $j[i], j[i'] < N$ ,  $j[i] + N \cdot j[i'] \in K(d_{\{i,i'\}})$  implies that  $v_i^{j[i]}$  and  $v_{i'}^{j[i']}$  are adjacent in  $G$ . Since  $i$  and  $i'$  were chosen arbitrarily, it follows that all vertices in  $S$  are adjacent to each other, i.e.,  $S$  induces a clique in  $G$ . Hence Claim 1 is shown.

*Claim 2: The treewidth of the value graph of  $C$  is at most  $\binom{k}{2} + 1$ .*

Let  $W = \{d_{\{i,i'\}} : 1 \leq i < i' \leq k\}$ . If we delete all vertices in  $W$  from the value graph of  $C$ , then we are left with a collection of  $k$  disjoint trees. Hence without the vertices in  $W$ , the value graph admits a tree decomposition of width 1. Now adding  $W$  to all the bags of this tree decomposition yields a tree decomposition of the full value graph of  $G$ . The width of this tree decomposition is  $|W| + 1 = \binom{k}{2} + 1$ . Hence Claim 2 is shown.

Since the construction of  $C$  from  $G$  can certainly be carried out in polynomial time (polynomial in  $G$  and  $k$ ), we have a reduction from the W[1]-complete CLIQUE problem to EGCC-CONSISTENCY with parameter treewidth. Hence the latter problem is W[1]-hard.  $\square$

**Corollary 20.** GENFACTOR parameterized by treewidth is W[1]-hard.

## 7 Conclusion

We have studied extended global cardinality constraints under restrictions of the language and under restrictions of the structure. We have shown that (complete) domain filtering and consistency checking for these constraints can be carried out in linear time if the parameters treewidth and cardinality-width are both bounded by arbitrary constants. In addition, we have shown that consistency checking is NP-hard if the cardinality-width is bounded alone and W[1]-hard if the treewidth is bounded alone. Furthermore, we have pointed out the connection between extended global cardinality constraints and general factors of graphs. By means of this connection we could identify the largest class of cardinality sets that admits polynomial-time consistency checking. An empirical evaluation of our theoretical results is left for future research. We hope that our work stimulates further research on global constraints under structural restrictions as well as the development of fixed-parameter algorithms for other global constraints.

**Acknowledgments.** We thank Jácint Szabó for discussions on Cornuéjols's algorithm and the anonymous referees for their helpful comments. This research was supported by the EP-SRC, project EP/E001394/1, and was carried out during the first author's postdoc position at the University of Durham. Some parts of this work appeared in preliminary form [29] in the Proceedings of the 14th Computing: The Australasian Theory Symposium (CATS'08).

## References

- [1] N. Beldiceanu, M. Carlsson, and J.-X. Rampon. Global Constraint Catalog. Technical Report T2005:08, Swedish Institute of Computer Science, Stockholm, Sweden, 2005.
- [2] C. Bessière, E. Hebrard, B. Hnich, and T. Walsh. The Tractability of Global Constraints. In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP'04)*, vol. 3258 of *LNCS*, pages 716–720. Springer-Verlag, 2004.
- [3] C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, C.-G. Quimper, and T. Walsh. The Parameterized Complexity of Global Constraints. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, pages 235–240. AAAI Press, 2008.
- [4] H. L. Bodlaender. *A Tourist Guide Through Treewidth*. Acta Cybernetica **11**(1-2):1–22, 1993.
- [5] H. L. Bodlaender. *A Linear Time Algorithm for Finding Tree-Decompositions of Small Treewidth*. SIAM Journal on Computing **25**(6):1305–1317, 1996.
- [6] H. L. Bodlaender. Discovering Treewidth. In *Proceedings of the 31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'05)*, vol. 3381 of *LNCS*, pages 1–16. Springer-Verlag, 2005.
- [7] S. Bourdais, P. Galinier, and G. Pesant. HIBISCUS: A Constraint Programming Application to Staff Scheduling in Health Care. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP'03)*, vol. 2833 of *LNCS*, pages 153–167. Springer-Verlag, 2003.
- [8] Y. Caseau, P.-Y. Guillo, and E. Levenez. *A Deductive and Object-Oriented Approach to a Complex Scheduling Problem*. Journal of Intelligent Information Systems **4**(2):149–166, 1995.
- [9] D. Cohen and P. Jeavons. *The Complexity of Constraint Languages*. In F. Rossi, P. van Beek, and T. Walsh, eds, Handbook of Constraint Programming, chapter 8, pages 245–280. Elsevier, 2006.
- [10] G. Cornuéjols. *General Factors of Graphs*. Journal of Combinatorial Theory, Series B, **45**(2):185–198, 1988.
- [11] G. Cornuéjols, D. Hartvigsen, and W. Pulleyblank. *Packing Subgraphs in a Graph*. Operations Research Letters **1**:139–143, 1982.
- [12] B. Courcelle. *Graph Rewriting: An Algebraic and Logic Approach*. In J. van Leeuwen, editor, Handbook of Theoretical Computer Science, volume B: Formal Models and Semantics, chapter 5, pages 193–242. Elsevier, 1990.
- [13] R. Dechter. *Tractable Structures for Constraint Satisfaction Problems*. In F. Rossi, P. van Beek, and T. Walsh, eds, Handbook of Constraint Programming, chapter 7, pages 209–244. Elsevier, 2006.
- [14] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
- [15] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.
- [16] G. Gottlob and R. Pichler and F. Wei. Bounded Treewidth as a Key to Tractability of Knowledge Representation and Reasoning. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence (AAAI'06)*, pages 250–256. AAAI Press, 2006.

- [17] W.-J. van Hoeve and I. Katriel. *Global Constraints*. In F. Rossi, P. van Beek, and T. Walsh, eds, *Handbook of Constraint Programming*, chapter 6, pages 169–208. Elsevier, 2006.
- [18] T. Kloks. *Treewidth: Computations and Approximations*. Springer-Verlag, 1994.
- [19] A. M. C. A. Koster, H. L. Bodlaender, and S. P. M. van Hoesel. *Treewidth: Computational Experiments*. *Electronic Notes in Discrete Mathematics* **8**:54–57, 2001.
- [20] L. Lovász. The Factorization of Graphs. In *Combinatorial Structures and Their Applications*, Gordon and Breach, pages 243–246, 1970.
- [21] L. Lovász. *The Factorization of Graphs II*. *Acta Mathematica Academiae Scientiarum Hungaricae* **23**:223–246, 1972.
- [22] S. Micali and V. V. Vazirani. An  $O(\sqrt{|V|} \cdot |E|)$  Algorithm for Finding a Maximum Matching in General Graphs. In *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (SFCS'80)*, pages 17–27. IEEE Computer Society, 1980.
- [23] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
- [24] K. Pietrzak. *On the Parameterized Complexity of the Fixed Alphabet Shortest Common Supersequence and Longest Common Subsequence Problems*. *Journal of Computer and System Sciences* **67**(4):757–771, 2003.
- [25] C.-G. Quimper, A. López-Ortiz, P. van Beek, and A. Golynski. Improved Algorithms for the Global Cardinality Constraint. In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP'04)*, vol. 3258 of *LNCS*, pages 542–556. Springer-Verlag, 2004.
- [26] J.-C. Régin. Generalized Arc Consistency for Global Cardinality Constraint. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence (AAAI'96)*, pages 209–215. AAAI Press, 1996.
- [27] J.-C. Régin and C. P. Gomes. The Cardinality Matrix Constraint. In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP'04)*, vol. 3258 of *LNCS*, pages 572–587. Springer-Verlag, 2004.
- [28] F. Rossi, P. van Beek, and T. Walsh, eds., *Handbook of Constraint Programming*. Elsevier, 2006.
- [29] M. Samer and S. Szeider. Tractable Cases of the Extended Global Cardinality Constraint. In *Proceedings of the 14th Computing: The Australasian Theory Symposium (CATS'08), Theory of Computing 2008*, vol. 77 of *CRPIT*, pages 67–74. Australian Computer Society, 2008.
- [30] M. Sellmann, T. Gellermann, and R. Wright. *Cost-Based Filtering for Shorter Path Constraints*. *Constraints* **12**(2):207–238, 2007.