

Tractable Cases of the Extended Global Cardinality Constraint*

Marko Samer

Stefan Szeider

Department of Computer Science
Durham University, UK

Email: {marko.samer, stefan.szeider}@durham.ac.uk

Abstract

We study the 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 $FPT \neq W[1]$) 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, bounded treewidth, parameterized complexity, dynamic programming, domain consistency

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 (Rossi et al. 2006). 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 (van Hoeve & Katriel 2006). Currently the Global Constraint Catalog (Beldiceanu et al. 2005) lists 276 global constraints.

*Research supported by the EPSRC, project EP/E001394/1. Copyright ©2008, Australian Computer Society, Inc. This paper appeared at the Fourteenth Computing: The Australasian Theory Symposium (CATS2008), Wollongong, New South Wales, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 77, James Harland and Prabhu Manyem, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

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* (Beldiceanu et al. 2005), *egcc* (Bessière et al. 2004), *distribution* (Bourdais et al. 2003), and *card_var_gcc* (Régin & Gomes 2004). 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* if there is a mapping $\alpha : X \rightarrow D$ such that

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

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.

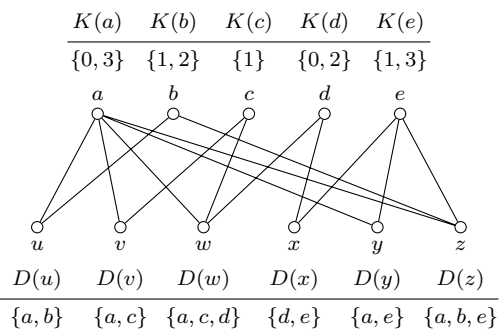


Figure 1: An EGC constraint and its value graph. The constraint is satisfied by $\alpha(u) = b$, $\alpha(v) = c$, $\alpha(w) = d$, $\alpha(x) = d$, $\alpha(y) = e$, and $\alpha(z) = b$.

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.

We consider the following decision problem:

EGCC-CONSISTENCY

Instance: An EGC constraint C .

Question: Is the constraint C consistent?

Quimper et al. (2004) have shown that EGCC-CONSISTENCY is NP-complete. However, as observed

by Régis (1996), 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.

Contributions of this paper

We explore properties of EGC constraints that admit polynomial-time consistency checking even in the presence of non-interval cardinality sets.

We discuss connections between EGCC-CONSISTENCY and the *general factor problem* for graphs as introduced by Lovász (1970, 1972). This connection seems not to be known in the constraint satisfaction literature. In view of this connection, a general result of Cornuéjols (1988) for the general factor problem allows us to generalize Régis’s polynomial-time result from intervals to “2-gap free” cardinality sets.

The main technical contributions of this paper are concerned with the complexity of EGCC-CONSISTENCY under structural restrictions. In particular, we consider instances whose value graphs have bounded treewidth. 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, that is, to remove from the domains of the variables those values that do not participate in a solution of the constraint. Domain filtering is an important task in the context of constraint solving (van Hoeve & Katriel 2006).

The polynomial that bounds the running time 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 the complexity theoretic assumption $FPT \neq W[1]$ (see Section 6).

As a corollary, we obtain that Lovász’s general factor problem, parameterized by the treewidth of the input graph, is $W[1]$ -hard. This result may be of independent interest.

The remainder of the 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 give a tractability result 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. 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 α that assigns to each variable $x \in X$ a value $\alpha(x) \in D(x)$. An assignment α *satisfies* a constraint C if α 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* 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 α 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” (van Hoeve & Katriel 2006).

For an EGC constraint C with set X of variables and set D of values we say that C is *over* (X, D) . The input size of an EGC constraint 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 (Bessière et al. 2004, Quimper et al. 2004), 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 (Bessière et al. 2004), but the converse does not hold in general (Sellmann 2003).

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 (Bodlaender 1993).

The treewidth of a graph $G = (V, E)$ is defined via the following notion of decomposition: a *tree decomposition* of G is a pair (T, χ) , where T is a tree and χ 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, χ) 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 .

Note that a graph $G = (V, E)$ of treewidth k has at most $k|V|$ edges (Kloks 1994). Thus if k is bounded by a constant, then $|E| = O(|V|)$.

In principle, one can compute in linear time an optimal tree decomposition of graphs with treewidth bounded by some constant k (Bodlaender 1996); however the running time 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 (Bodlaender 2005, Koster et al. 2001).

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 (Kloks 1994): A triple (T, χ, r) is a *nice tree decomposition* of a graph G if (T, χ) 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 set $t_v = r$. In both cases we say that t_v is the *final node* of v .

For every constant k , given a tree decomposition of a graph G of width k , one can effectively obtain in linear time a nice tree decomposition of G with $O(n)$ nodes and of width at most k (Kloks 1994).

3 EGC Constraints and General Factors in Graphs

3.1 Applying Cornuéjols's Theorem

Lovász (1970, 1972) 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 $K(v) \subseteq \{0, \dots, d(v)\}$ of integers, where $d(v)$ denotes the degree of v in G .

Question: Is there a subset $F \subset E$ 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 GENFACTOR(\mathcal{K}) and EGCC-CONSISTENCY(\mathcal{K}) the respective problems restricted to instances where all cardinality sets belong to the

class \mathcal{K} . A set K of integers has an s -gap if there exists an integer i such that $\min(K) < i < \max(K)$ and $\{i, \dots, i+s-1\} \cap K = \emptyset$. We denote by \mathcal{I}_s the class of s -gap free sets of non-negative integers. Note that \mathcal{I}_1 is nothing but the class of non-negative intervals. We can state Régin's above mentioned result as follows.

Proposition 1 (Régin (1996)). EGCC-CONSISTENCY(\mathcal{I}_1) can be decided in polynomial time.

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

Theorem 2 (Cornuéjols (1988)). If $\mathcal{K} \subseteq \mathcal{I}_2$, then GENFACTOR(\mathcal{K}) can be decided in polynomial time. Otherwise, GENFACTOR(\mathcal{K}) is NP-complete.

Actually, the reduction given by Cornuéjols (1988) shows that the NP-hardness case of Theorem 2 even holds if the graph is bipartite and the vertices on one side have cardinality sets $\{1\}$, the vertices on the other side have cardinality sets drawn from the class \mathcal{K} . Hence, it follows that the dichotomy stated in Theorem 2 also holds for EGCC-CONSISTENCY(\mathcal{K}).

Corollary 3. If $\mathcal{K} \subseteq \mathcal{I}_2$, then EGCC-CONSISTENCY can be decided in polynomial time. Otherwise, EGCC-CONSISTENCY(\mathcal{K}) is NP-complete.

Thus EGCC-CONSISTENCY(\mathcal{I}_2) can be decided in polynomial time, a proper generalization of Proposition 1. A further consequence of Corollary 3 is the NP-completeness of EGCC-CONSISTENCY($\{\{0, 3\}\}$), which gives the following.

Corollary 4. EGCC-CONSISTENCY is NP-complete for instances of cardinality-width at least 3.

3.2 Applying Courcelle's Theorem

Courcelle's Theorem (Courcelle 1987) 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 (Downey & Fellows 1999) for further details and examples). In fact, with Courcelle's Theorem it is easy to establish the following.

Proposition 5. 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 two, as vertices of degree 0 or 1 can be easily eliminated. 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 $K(v) = K_{i(v)}$. Now we construct a graph G' from G by attaching to every vertex v of G new neighbors $v_1, \dots, v_{i(v)}$ of degree 1. Since the added vertices are of 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 neighbors of degree 1 (equivalently, v belongs to G and $K(v) = K_{i(v)}$). It is now easy to state an MSO sentence that is true for G' if and only if G has a general

factor that meets the cardinality conditions imposed by the sets $K(v)$. Hence Proposition 5 follows from Courcelle’s Theorem.

In the above construction it was essential that the cardinality-width is bounded, since otherwise we could not confine us 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 running time that is bounded by a polynomial of order independent of the treewidth.

Algorithms obtained via Courcelle’s Theorem are impractical as the linear running time 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, χ, 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 $\alpha : X_\alpha \rightarrow \text{val}^*(t)$ defined on sets of variables $X_\alpha \subseteq \text{var}^*(t)$ such that

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

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

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

such that the following two conditions are satisfied:

1. $R(x) \in (\text{val}(t) \cap D(x)) \cup \{\perp, \star\}$ 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 α in the set $A(t)$: $R(x) = \perp$ means that α is not defined for x , and $R(x) = \star$ means that α 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 $\alpha \in A(t)$ if the following two conditions hold:

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

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

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

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

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 $\alpha \in A(t)$.

The following lemma follows directly from the definitions.

Lemma 6. *C is consistent if and only if there is a valid record R of the root r such that $R(x) \neq \perp$ for all $x \in \text{var}(r)$ and $R(d) \in K(d)$ for all $d \in \text{val}(r)$.*

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

Lemma 7 (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*

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

- (a) $R(x) = R_1(x) = R_2(x) \in D \cup \{\perp\}$;
- (b) $R(x) = R_1(x) = \star$ and $R_2(x) = \perp$;
- (c) $R(x) = R_2(x) = \star$ and $R_1(x) = \perp$;

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 $\alpha \in A(t)$. For $i = 1, 2$ let $\alpha_i = \alpha|_{\text{var}^*(t_i)}$ be the restriction of α to $\text{var}^*(t_i)$. Since $\alpha \in A(t)$ it can be easily verified that also $\alpha_i \in A(t_i)$, $i = 1, 2$. Let R_i be the (valid) record of t_i that represents α_i , $i = 1, 2$. It remains to check that for R, R_1 , and R_2 the two 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 $X_\alpha \cap \text{var}(t) = X_{\alpha_1} \cap \text{var}(t_1) = X_{\alpha_2} \cap \text{var}(t_2)$. If $R_1(x) = \star$ and $R_2(x) = \perp$, then $\alpha_1(x) \in \text{val}^*(t_1) \setminus \text{val}(t_1)$ and $x \notin X_{\alpha_2}$; hence $\alpha(x) \in \text{val}^*(t) \setminus \text{val}(t)$, 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 for R, R_1 , and R_2 . 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 $\alpha_1 \in A(t_1)$ and $\alpha_2 \in A(t_2)$ 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 α_1 and α_2 to an assignment $\alpha : X_\alpha \rightarrow D$, defined for the set $X_\alpha = X_{\alpha_1} \cup X_{\alpha_2}$, with $\alpha(x) \in D(x)$ for all $x \in X_\alpha$. It is easy to check that α corresponds to R , hence R is a valid record of t . \square

The proofs of the following four lemmas are straightforward.

Lemma 8 (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 prevails:*

1. $R(x_0) = \perp$ and $R(d) = R'(d)$ for all $d \in \text{val}(t)$,

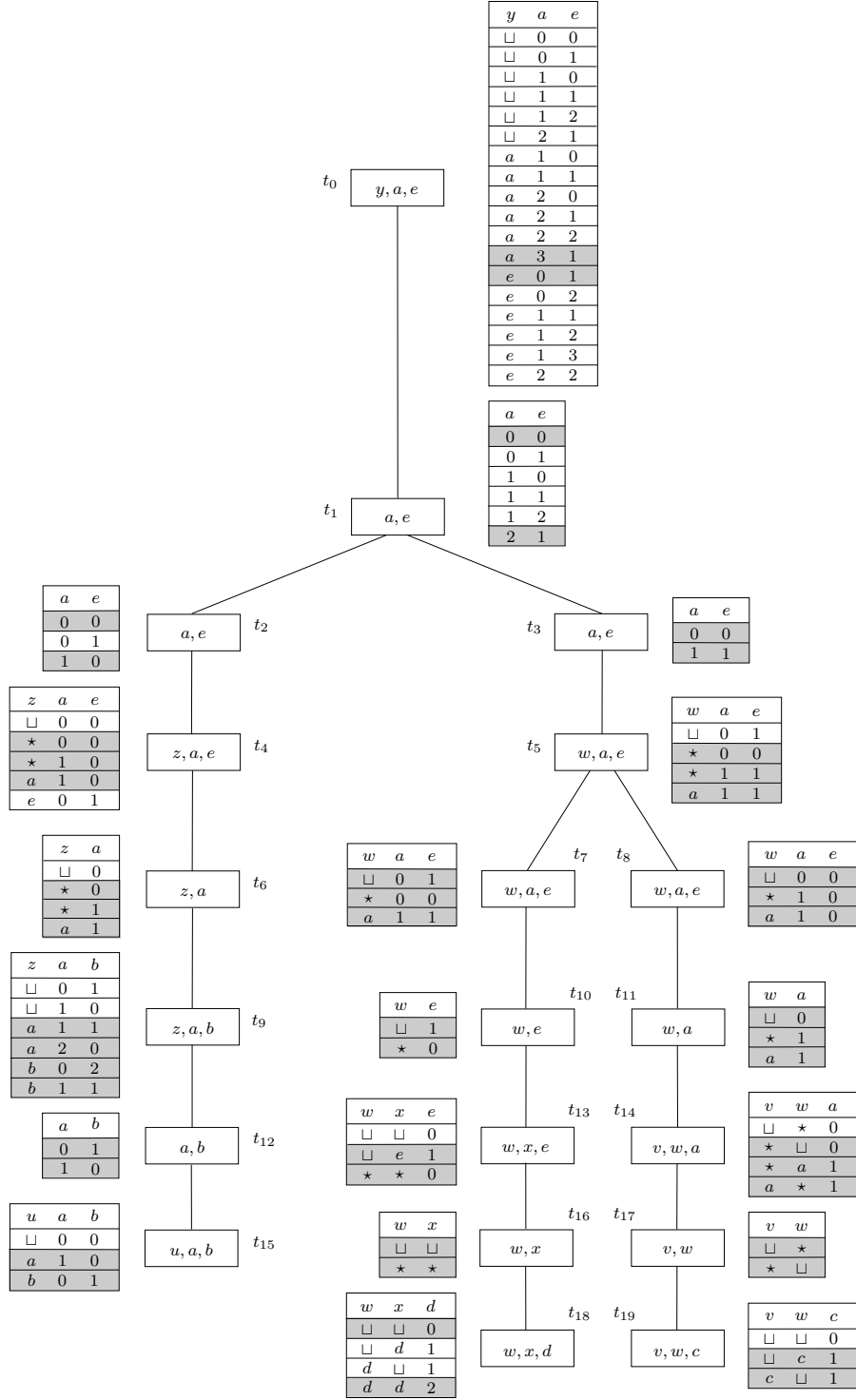


Figure 2: A nice tree decomposition of the value graph of the EGC constraint in Figure 1 with tables representing valid records. Rows with gray background indicate the result after domain filtering.

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 $d \in \text{val}(t) \setminus \{d_0\}$,

Lemma 9 (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 conditions hold:*

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

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

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

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

Lemma 10 (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 $z \in \text{var}(t) \cup \text{val}(t)$.*

Lemma 11 (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 conditions hold:

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

Theorem 12. EGCC-CONSISTENCY can be decided in linear time for instances having both bounded treewidth and bounded cardinality-width.

Proof. Let $k, m \geq 0$ be arbitrary constants. We are given an EGC constraint C over (X, D) with treewidth and cardinality-width bounded by k and m , respectively. Let n denote the number of vertices of the value graph of C . We compute a nice tree decomposition (T, χ, r) of the value graph of C such that the width of the tree decomposition is at most k and T has $O(n)$ nodes. This can be accomplished in time $O(n)$ (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 $d \in \text{val}(t)$. The number of records of a node t is at most

$$|\text{val}(t) \cup \{\sqcup, \star\}|^{|\text{var}(t)|} \cdot m^{|\text{val}(t)|} \leq \max(k+1, m)^{k+1},$$

i.e., bounded purely in terms of the constants k and m . Lemmas 7–11 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 7. The time required for each pair is bounded in terms of the constants k and m . Hence, given $M(t_1)$ and $M(t_2)$, we can compute $M(t)$ in constant time. Computing the sets $M(t)$ for introduce and forget nodes t according to Lemmas 8–11 is even simpler. Hence we can compute the sets $M(t)$ for all $O(n)$ tree nodes t in time $O(n)$. According to Lemma 6 we can decide consistency of C by examining the records in $M(r)$ at the root r . \square

Figure 2 shows a nice tree decomposition of the value graph of the constraint of Figure 1, together with the sets $M(t)$ as computed according to the proof of Theorem 12. 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) . For each pair $x \in X$ and $d \in D(x)$ let $C[x = d]$ denote the EGC constraint obtained from C by instantiating x with d (that is, x gets removed and $K(d)$ gets replaced with $K'(d) = \{j - 1 : j \in K(d) \setminus \{0\}\}$). Evidently, 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 $d \in D(x)$ are consistent. Hence, domain filtering for EGC constraints of bounded treewidth and bounded cardinality-width can be carried out in quadratic time, using the algorithm of Theorem 12 for

each pair $x \in X$ and $d \in D(x)$. However, the following approach allows domain filtering in linear time.

As in the previous section, let C be an EGC constraint over (X, D) . We assume that treewidth and cardinality-width of C are bounded by constants k and m , respectively. Let (T, χ, 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 $O(n)$ nodes, $n = |X| + |D|$.

We call a record R of a tree node t *solution-valid* if R represents the restriction $\alpha|_{\text{var}^*(t)}$ of a solution $\alpha : X \rightarrow D$ 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 13. 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 $R(d) = j$.

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 12. 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 2 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 6, a valid record R at r is solution-valid if and only if $R(x) \neq \sqcup$ 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 7 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 7 hold. Hence, we can compute the sets $M'(t_1)$ and $M'(t_2)$ from $M'(t)$ in time that only depends on the constants 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 8–11 hold. Thus if we know $M'(t)$ we can compute $M'(t')$ in time that only depends on the constants k and m .

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

Theorem 14. Domain filtering for extended global cardinality constraints can be carried out in linear time if both treewidth and cardinality-width are bounded.

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 5. The framework of parameterized complexity (Downey & Fellows 1999) 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 (Downey & Fellows 1999, Flum & Grohe 2006, Niedermeier 2006).

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 $O(f(k) \cdot n^c)$ 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 $W[1] \subseteq W[2] \subseteq \dots \subseteq W[P]$ contain problems that are believed to be not fixed-parameter tractable (Downey & Fellows 1999); all inclusions are believed to be proper. There are different kinds of evidence for assuming that $W[1] \neq \text{FPT}$. For example, $W[1] \neq \text{FPT}$ would imply that the Exponential Time Hypothesis fails (cf. Flum & Grohe (2006)). 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 $O(f(k) \cdot |x|^c)$ (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 (Downey & Fellows 1999). 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 Fellows et al. (2007), 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 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 15. 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 $j[i], j[i'] < N$, 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 3 for an illustration.

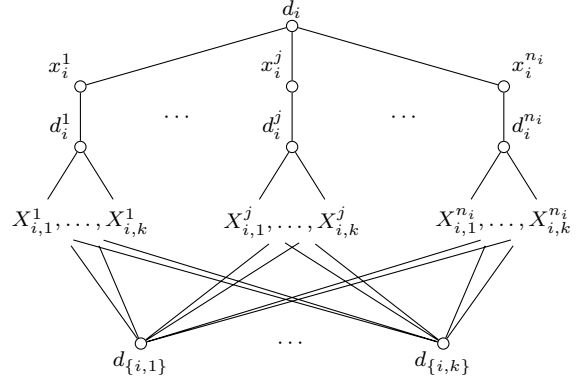


Figure 3: The i -th part of the value graph of the constraint constructed in the proof of Theorem 15.

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 α for C as follows. We put

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

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

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

It can be easily checked that α satisfies C .

Conversely, let α be an assignment that satisfies C . For every $i \in \{1, \dots, k\}$ there is exactly one $j \in \{1, \dots, n_i\}$ with $\alpha(x_i^j) = d_i$, 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 16. GENFACTOR parameterized by treewidth is $W[1]$ -hard.

7 Conclusion

We have studied extended global cardinality constraints under structural restrictions. 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. Furthermore 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 global 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.

References

- Beldiceanu, N., Carlsson, M. & Rampon, J.-X. (2005), ‘Global constraint catalog’, Technical Report T2005:08. Swedish Institute of Computer Science, Stockholm, Sweden.
- Bessière, C., Hebrard, E., Hnich, B. & Walsh, T. (2004), 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*, Springer-Verlag, pp. 716–720.
- Bodlaender, H. L. (1993), ‘A tourist guide through treewidth’, *Acta Cybernetica* **11**(1-2), 1–22.
- Bodlaender, H. L. (1996), ‘A linear time algorithm for finding tree-decompositions of small treewidth’, *SIAM Journal on Computing* **25**(6), 1305–1317.
- Bodlaender, H. L. (2005), Discovering treewidth, in ‘Proceedings of the 31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM’05)’, Vol. 3381 of *LNCS*, Springer-Verlag, pp. 1–16.
- Bourdais, S., Galinier, P. & Pesant, G. (2003), 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*, Springer-Verlag, pp. 153–167.
- Cornuéjols, G. (1988), ‘General factors of graphs’, *Journal of Combinatorial Theory, Series B* **45**(2), 185–198.
- Courcelle, B. (1987), ‘Recognizability and second-order definability for sets of finite graphs’, Technical Report I-8634. Université de Bordeaux, Bordeaux, France.
- Downey, R. G. & Fellows, M. R. (1999), *Parameterized Complexity*, Springer-Verlag.
- Fellows, M. R., Hermelin, D. & Rosamond, F. (2007), ‘On the fixed-parameter intractability and tractability of multiple-interval graph problems’, Manuscript.
- Flum, J. & Grohe, M. (2006), *Parameterized Complexity Theory*, Springer-Verlag.
- van Hoeve, W.-J. & Katriel, I. (2006), Global constraints, in F. Rossi, P. van Beek & T. Walsh, eds, ‘Handbook of Constraint Programming’, Elsevier, chapter 6, pp. 169–208.
- Kloks, T. (1994), *Treewidth: Computations and approximations*, Springer-Verlag.
- Koster, A. M. C. A., Bodlaender, H. L. & van Hoesel, S. P. M. (2001), ‘Treewidth: Computational experiments’, *Electronic Notes in Discrete Mathematics* **8**.
- Lovász, L. (1970), The factorization of graphs, in ‘Combinatorial Structures and their Applications’, Gordon and Breach, pp. 243–246.
- Lovász, L. (1972), ‘The factorization of graphs II’, *Acta Mathematica Academiae Scientiarum Hungaricae* **23**, 223–246.
- Niedermeier, R. (2006), *Invitation to Fixed-Parameter Algorithms*, Oxford University Press.
- Quimper, C.-G., López-Ortiz, A., van Beek, P. & Golynski, A. (2004), 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*, Springer-Verlag, pp. 542–556.
- Régin, J.-C. (1996), Generalized arc consistency for global cardinality constraint, in ‘Proceedings of the 13th National Conference of the American Association for Artificial Intelligence (AAAI’96)’, AAAI Press, pp. 209–215.
- Régin, J.-C. & Gomes, C. P. (2004), The cardinality matrix constraint, in ‘Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP’04)’, Vol. 3258 of *LNCS*, Springer-Verlag, pp. 572–587.
- Rossi, F., van Beek, P. & Walsh, T., eds (2006), *Handbook of Constraint Programming*, Elsevier.
- Sellmann, M. (2003), Cost-based filtering for shorter path constraints, in ‘Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP’03)’, Vol. 2833 of *LNCS*, Springer-Verlag, pp. 694–708.