

The REGULARGCC Matrix Constraint

Ronald de Haan¹, Nina Narodytska², and Toby Walsh²

¹ Technische Universität Wien

dehaan.ronald@gmail.com

² NICTA and University of NSW, Sydney

{nina.narodytska,toby.walsh}@nicta.com.au

Abstract. We study propagation of a global constraint that ensures that each row of a matrix of decision variables satisfies a REGULAR constraint, and each column satisfies a GCC constraint. On the negative side, we prove that propagation is NP-hard even under some strong restrictions (e.g. just 2 values, just 4 states in the automaton, just 5 columns to the matrix, or restricting to limited classes of automata). We also prove that propagation is W[2]-hard when the problem is parameterized by the number of rows in the matrix. On the positive side, we identify several cases where propagation is fixed parameter tractable.

1 Introduction

Constraints are often applied to the rows and columns of a matrix of decision variables [6–8, 17]. For example, the REGULARGCC constraint posts a REGULAR constraint on each row of such a matrix, and a GCC constraint on each column [1]. The REGULAR constraints ensure that each row is a sequence of values accepted by a deterministic finite automaton (DFA). The GCC constraints enforce upper and lower bound on the occurrences of given values down each column. This modelling pattern occurs in nurse scheduling problems. The row constraints ensure shift rules (e.g. two days off after a night shift) whilst the column constraints ensure that sufficient staff are on duty in each shift. This modelling pattern also occurs in call center rostering and related problems.

In this paper, we investigate the computational complexity of propagating the REGULARGCC constraint. We provide a detailed picture of the conditions under which propagation is (in)tractable [2–4]. The REGULARGCC constraint imposes a special structure on the problem. Such structure in constraints can have a positive impact on the complexity of propagation (e.g. by using the structure of the automaton, the REGULAR constraint can be propagated in linear time [14, 16]). In our investigation of the complexity of propagating REGULARGCC, we will make use of structural properties of the constraint. In particular, we will identify tractable cases by bounding certain structural properties.

2 Background

Constraint Satisfaction Problem A constraint satisfaction problem consists of a set of variables $X = (x_1, \dots, x_n)$, each with a domain of values $D = (D_1, \dots, D_n)$,

and a set of constraints specifying allowed combinations of values for given subsets of variables. A solution is an assignment of values to the variables satisfying the constraints. A common method to find a solution of a constraint satisfaction problem is backtracking search. Constraint solvers typically prune the backtracking search space by enforcing a local consistency property like domain consistency. A constraint is *domain consistent* (*DC*) iff when a variable is assigned any of the values in its domain, there exist compatible values in the domains of all the other variables of the constraint. Such combination of values is called a *support*. A constraint is *bound consistent* (*BC*) iff when a variable is assigned the minimum or maximum value in its domain, there exist compatible values between the minimum and maximum domain value for all the other variables. Such combination of values is called a *bound support*.

Global Constraints. An automaton $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$ consists of an alphabet Σ , a set of states Q , an initial state q_0 , a set of accepting states F , and a transition relation δ defining the possible next states from a state given a symbol. $\mathcal{A}$ is deterministic (DFA) if there is only one possible next state, non-deterministic (NFA) otherwise. A string s is *recognized* by $\mathcal{A}$ iff starting from the state q_0 we can reach an accepting state using the transition relation δ . Both DFAs and NFAs recognize precisely regular languages. The constraint $\text{REGULAR}(\mathcal{A}, (X_1, \dots, X_n))$ is satisfied iff X_1 to X_n is a string accepted by $\mathcal{A}$ [14].

The $\text{STRETCH}((x_1, \dots, x_n), \mathcal{A}, [l_{s_1}, \dots, l_{|\Sigma|}], [u_{s_1}, \dots, u_{|\Sigma|}])$ constraint, where $s_i \in \Sigma$, is a special case of the REGULAR constraint [9]. The automaton $\mathcal{A}$ defines possible combinations of two consecutive symbols. Parameters l_{s_i} and u_{s_i} restrict the length of a stretch of symbol s_i in a sequence from below and from above, respectively. The $\text{GCC}((x_1, \dots, x_n), [l_1, \dots, l_m], [u_1, \dots, u_m])$ constraint ensures that the value i , $i = 1, \dots, m$ occurs between l_i and u_i times in x_1 to x_n . The SEQUENCE constraint is a conjunction of overlapping AMONG constraints. The $\text{AMONG}(l, u, (x_1, \dots, x_k), v)$ constraint holds iff $l \leq |\{i | x_i \in v\}| \leq u$. $\text{SEQUENCE}(l, u, k, (x_1, x_2, \dots, x_n), v)$ holds iff for $1 \leq i \leq n - k + 1$, $\text{AMONG}(l, u, (X_i, X_{i+1}, \dots, X_{i+k-1}), v)$ holds.

We define the XY constraint over a matrix of variables $\mathcal{M}$ with R rows and C columns where X and Y are two global constraints. $\text{XY}(\mathcal{M})$ holds iff X holds on each row of $\mathcal{M}$ and Y holds on each column of $\mathcal{M}$.

Parameterized Complexity. A problem is fixed parameter tractable (FPT) to compute in a parameter k if it can be solved in $O(f(k)n^c)$ time where f is any computable function, c is a constant, and n is the size of the input. Above FPT, there exists a hierarchy of fixed-parameter *intractable* problem classes: $\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \dots$. The clique problem is $\text{W}[1]$ -complete with respect to the size of the clique, whilst the dominating set problem is $\text{W}[2]$ -complete with respect to the size of the dominating set. $\text{W}[t]$ is characterized by the depth t of unbounded fan-in gates in a Boolean circuit specifying the problem. There is considerable evidence to suggest that such problem classes are intractable. For example, the halting problem for non-deterministic Turing machines is $\text{W}[1]$ -complete with respect to the length of the accepting computation.

3 Intractable Cases

We first prove that propagating the REGULARGCC matrix constraint is NP-hard even under strong conditions. For example, it is NP-hard to enforce even a weak level of local consistency like bound consistency with just a few values.

Theorem 1. *Enforcing BC on REGULARGCC is NP-hard with just 3 values, REGULAR constraints given by a DFA of size 4, and GCC constraints specifying only an upper bound on the number of occurrences of one particular value.*

Proof: Reduction from 3-SAT. Let $\varphi = \gamma_1 \wedge \dots \wedge \gamma_C$ be a 3-CNF formula over Boolean variables $p_1, \dots, p_R$. We construct an $R \times C$ matrix $\mathcal{M}$ of decision variables taking their values from $\{-1, 0, 1\}$, where each row $1 \leq r \leq R$ corresponds to a propositional variable p_r and each column $1 \leq c \leq C$ corresponds to a clause γ_c . To initialize the domain of variables in the matrix, we do the following for each clause $\gamma_i = l_1^i \vee l_2^i \vee l_3^i$. We set $\mathcal{M}_{r,i} = 0$ for all propositions p_r not occurring in γ_i . For $j \in \{1, 2, 3\}$ we set $\mathcal{M}_{i,k} \in \{0, 1\}$ if $l_j^i = p_k$ and we set $\mathcal{M}_{i,k} \in \{0, -1\}$ if $l_j^i = \neg p_k$. On each column we put the GCC constraint that states that the value 0 occurs at most $R - 1$ times. On each row we put the REGULAR constraint that states that besides 0's either only 1's or only -1's occur. Figure 1 demonstrates that this REGULAR constraint can be enforced by a DFA of size 4. This instance of REGULARGCC has a solution iff φ is satisfiable.

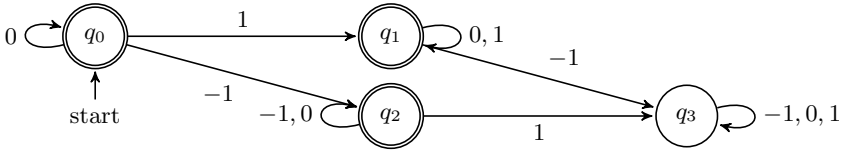


Fig. 1. Deterministic finite automaton used in the proof of Theorem 1

($\Rightarrow$) We create a satisfying assignment I for φ as follows. For each p_r , if in row r occurs at least one 1, we let $I(p_r) = \top$, otherwise we let $I(p_r) = \perp$ (the choice of $I(p_r)$ when only 0's occur in row r is arbitrary). Since in each column c there occur only $R - 1$ many 0's, we know that there exists some p_i for which $\mathcal{M}_{i,c} \neq 0$. By construction of $\mathcal{M}$, this can only happen if $I(l_j^c) = \top$ for some $j \in \{1, 2, 3\}$. Therefore I satisfies γ_c .

($\Leftarrow$) Let I be an assignment satisfying φ . We can instantiate $\mathcal{M}$ as follows. For each clause $\gamma_c = l_1^c \vee l_2^c \vee l_3^c$, for $j \in \{1, 2, 3\}$ we do the following. If $l_j^c = p_k$ and $I(p_k) = \top$, we let $\mathcal{M}_{k,c} = 1$. If $l_j^c = \neg p_k$ and $I(p_k) = \perp$, we let $\mathcal{M}_{k,c} = -1$. Otherwise, we let $\mathcal{M}_{k,c} = 0$. Since I is functional each REGULAR constraint on the rows is satisfied. Also, since at least one literal is satisfied in each clause, each column contains at least one value that is not 0, so the GCC constraints are satisfied. $\square$

In fact, we can strengthen the first condition to show that enforcing BC on REGULARGCC is intractable with a matrix containing just 2 values.

Theorem 2. *Enforcing BC on REGULARGCC is NP-hard with just 2 values.*

Proof: Reduction from the Exact Cover By 3-Set problem. We are given $F = \{S_1, \dots, S_n\}$ with $|S_i| = 3$, $\bigcup_i S_i = U$, $|U| = 3m$. We ask if there exists some subset $C \subseteq F$ with $|C| = m$ and $U = \bigcup_{c \in C} c$. W.l.o.g. we assume U contains the integers 1 to $3m$. We construct a $n \times (n + 3m)$ matrix of decision variables ranging over $\{0, 1\}$. The first n columns will indicate which S_j is included in the cover. The first n columns have a GCC constraint to ensure at most one 1 in each column. The next $3m$ columns represent the elements of the chosen set S_j . The REGULAR constraint on each row reads in the first n values. It only accepts those sequences that contain a single 1 in the first n values. Suppose there is a 1 at the j th position for $j \leq n$. Then this row represents the set S_j . Suppose $S_j = \{p, q, r\}$. Then the automaton only accepts those sequences with 1 at positions $j, n+p, n+q$ and $n+r$ and 0 everywhere else. To ensure that the m sets form a cover, we have a GCC constraint on the final $3m$ columns that ensures that it contains exactly one occurrence of 1. A solution of this REGULARGCC constraint represents an exact cover. $\square$

These NP-hardness results hold for very restricted cases, involving only few values or automata with only a few states.

4 Special Cases

We will consider three special cases of the REGULARGCC constraint which turn up frequently in scheduling and rostering problems. These three special cases restrict the REGULAR constraints to a limited class of automaton. We consider these special cases of the REGULARGCC constraint in an attempt to find tractable cases. As we will see, however, just restricting the REGULAR constraints on the rows to these special cases does not yield tractability.

The first special case is the STRETCHGCC constraint in which the REGULAR constraint on each row encodes a STRETCH constraint [9]. This constrains the length of any sequence of consecutive values (e.g. no more than 3 night shifts in a row), as well as the permitted transitions (e.g. a night shift can only be followed by a rest period). Unfortunately, propagation of STRETCHGCC is intractable even under strong conditions. For example, it is NP-hard to enforce a weak level of local consistency like bound consistency with just 3 values.

Theorem 3. *Enforcing BC on STRETCHGCC is NP-hard with just 3 values.*

Proof: Reduction from the Exact Cover problem. We are given $F = \{S_1, \dots, S_n\}$ with $\bigcup_i S_i = U$. We ask if there is some subset $C \subseteq F$ with $\bigcup_{c \in C} c = U$ and $c \cap c' = \emptyset$ for all distinct $c, c' \in C$. W.l.o.g. we assume that U contains the integers 1 to $|U|$. We construct a $|F| \times |U|$ matrix $\mathcal{M}$, of decision variables taking their value in $\{-1, 0, 1\}$. For each row $1 \leq r \leq |F|$ and each

value $1 \leq i \leq |U|$ we do the following. If $i \in S_r$, we let $\mathcal{M}_{r,i} \in \{0, 1\}$. If $i \notin S_r$, we let $\mathcal{M}_{r,i} \in \{-1, 0\}$. On each column we put the GCC constraint that states that the value 1 occurs exactly once. On each row we put the STRETCH constraint stating that each stretch of 0's must have a length of at least $|U|$. We show that this instance of STRETCHGCC has a solution iff there exists an exact cover.

($\Rightarrow$) Take a solution for our instance. We let C be the set of all U_r for which row r in the solution contains only -1 's and 1 's. Obviously $C \subseteq F$. In order to show that $\bigcup_{c \in C} c = U$, it suffices to show that $U \subseteq \bigcup_{c \in C} c$. Take an arbitrary $i \in U$. Since our solution contains at least one 1 in each column, we know there is some $c \in C$ such that $i \in c$. We also show that all distinct $c, c' \in C$ are disjoint. Take arbitrary $c, c' \in C$ such that $c \neq c'$. Assume $j \in c \cap c'$. This means that column j in the solution would contain two 1's, which contradicts the GCC constraints on the columns.

($\Leftarrow$) Let $C \subseteq F$ be an exact cover. We fill $\mathcal{M}$ as follows. For each row $1 \leq r \leq |F|$, we do the following. If $S_r \in C$, fill the row with -1 's and 1 's (this can be done only in one way). Otherwise, fill row r with only 0's. Obviously, the STRETCH constraints on the rows are satisfied. Also, since C is an exact cover, we know that for each $1 \leq i \leq |U|$ there is exactly one row r such that $\mathcal{M}_{r,i} = 1$. Thus the GCC constraints on the columns are satisfied. $\square$

The second special case we consider is the SLIDEGCC constraint, in which the REGULAR row constraints encode SLIDE meta constraints [2]. This constrains all subwords of a particular length k to instances of a given k -ary constraint (e.g. directly after two night shifts there must be either a small shift or a rest period). Such SLIDE constraints can be encoded using REGULAR constraints with automata whose number of states depends only on k . Unfortunately, SLIDEGCC propagation is again intractable even under strong conditions.

Corollary 1. *Enforcing BC on SLIDEGCC is NP-hard, already for SLIDE constraints based on constraints of arity 2 and just 3 values.*

Proof: (Sketch.) The reduction from the proof of Theorem 3 can be adapted for SLIDEGCC by putting on each row a REGULAR constraint accepting only words that contain either only 0's or only -1 's and 1 's. This can be expressed using a SLIDE constraint with arity 2. $\square$

The third special case of REGULARGCC we consider is the SEQUENCEGCC constraint in which the REGULAR constraint on each row encodes a SEQUENCE constraint [13]. This constrains the repetition of values within a sliding window (e.g. there are at least 2 days off in any 7 day period, and at most 3 night shifts in any 7 day period). Enforcing domain or bound consistency on the SEQUENCEGCC constraint is NP-hard even if the matrix has just a few columns. For the sake of presentation, we firstly give an intractability proof for domain consistency, and then extend this proof to bound consistency.

Theorem 4. *Enforcing DC on SEQUENCEGCC is NP-hard with just 5 columns.*

Proof: Reduction from the 3D Matching problem. Given three pair-wise disjoint sets W, Z, Y of equal size q and a set $M \subseteq W \times Z \times Y$, $|M| = m$, we ask if there

exists $M' \subseteq M$ such that $|M'| = q$ and no two different elements of M' agree in any coordinate. Assume $M = \{s_1, \dots, s_m\}$. We create a $m \times 5$ matrix $\mathcal{M}$ of decision variables taking their value in $\{0, t, w_1, \dots, w_q, z_1, \dots, z_q, y_1, \dots, y_q\}$. For each $(w_i, z_i, y_i) = s_i$ we let $\mathcal{M}_{i,1} \in \{0, w_i\}$, $\mathcal{M}_{i,2} \in \{0, t\}$, $\mathcal{M}_{i,3} \in \{0, z_i\}$, $\mathcal{M}_{i,4} \in \{0, t\}$, and $\mathcal{M}_{i,5} \in \{0, y_i\}$. We constrain each row i with the constraint $\text{SEQUENCE}(\mathcal{M}_i, 1, 2, 2, \{0\})$, stating that in each sequence of length 2, at least one 0 occurs. On columns 1 (resp. 3 and 5) we put the GCC constraint stating that each value in W (resp. Z and Y) occurs at least once, and that at least $m - q$ many 0's occur. On columns 2 and 4 we put the GCC constraint stating that at least q many t 's occur, and at least $m - q$ many 0's. This instance of SEQUENCEGCC has a solution iff there exists a matching.

($\Rightarrow$) Take a solution for SEQUENCEGCC . We know column 2 contains exactly $m - q$ many t 's, and q many 0's. For each occurrence of a t in column 2 at row i , columns 1 and 3 contain a 0 at row i (by the SEQUENCE constraint). Then, by the GCC constraint, for all rows j where column 2 contains a 0, columns 1 and 3 contain a non-0 at row j , and thus (by SEQUENCE) column 4 contains a 0 at row j . By a similar argument, we know that in the remaining rows column 4 contains t 's. Continuing this argument for column 5, we know that in the solution there are q many rows taking values $(w_i, 0, z_i, 0, y_i)$ and $m - q$ rows taking values $(0, t, 0, t, 0)$. By the GCC constraints, we know that each value $w \in W$ occurs exactly once, as well as each value $z \in Z$ and each $y \in Y$. Since the possible values were chosen by taking elements from M , we know that $M' = \{s_i \mid \mathcal{M}_i \neq (0, t, 0, t, 0)\}$ is a 3D matching.

($\Leftarrow$) Let $M' \subseteq M$ be a 3D matching. We can fill $\mathcal{M}$ as follows. For each $(w_i, z_i, y_i) = s_i \in M'$, we let $\mathcal{M}_i = (w_i, 0, z_i, 0, y_i)$. For each $s_i \in M \setminus M'$ we let $\mathcal{M}_i = (0, t, 0, t, 0)$. Obviously each row satisfies the SEQUENCE constraint. Since $|M'| = q$ and each value $w \in W$ occurs exactly once in the first coordinate of M' (and similarly for values $z \in Z$ and the second coordinate, and $y \in Y$ and the third coordinate), we have that each column satisfies the corresponding GCC constraint. $\square$

Theorem 5. *Enforcing BC on SEQUENCEGCC is NP-hard with just 5 columns.*

Proof: The proof is similar to the proof of Theorem 4. Let cw_i (resp. cz_i , cy_i) be the number of occurrences of the value w_i (resp. z_i , y_i) in M . For each value w_i (resp. z_i , y_i) we create $cw_i - 1$ (resp. $cz_i - 1$, $cy_i - 1$) clones of it. We now define the total order on values as $U = [-w_q^1, \dots, -w_q^{cw_q-1}, \dots, -w_1^1, \dots, -w_1^{cw_1-1}, -z_q^1, \dots, -z_q^{cz_q-1}, \dots, -z_1^1, \dots, -z_1^{cz_1-1}, -y_q^1, \dots, -y_q^{cy_q-1}, \dots, -y_1^1, \dots, -y_1^{cy_1-1}, 0, t, y_1, \dots, y_q, z_1, \dots, z_q, w_1, \dots, w_q]$. We create the matrix $\mathcal{M}$ in a similar fashion as in the proof for Theorem 4, with the difference that for each $(w_i, z_i, y_i) = s_i$ we let $\mathcal{M}_{i,1} \in [-w_i^1, \dots, -w_i^{cw_i-1}, \dots, w_i]$, $\mathcal{M}_{i,3} \in [-z_i^1, \dots, -z_i^{cz_i-1}, \dots, z_i]$, $\mathcal{M}_{i,5} \in [-y_i^1, \dots, -y_i^{cy_i-1}, \dots, y_i]$, and $\mathcal{M}_{i,2} \in [0, t]$ and $\mathcal{M}_{i,4} \in [0, t]$. We adapt the constraint on rows to $\text{SEQUENCE}(\mathcal{M}_i, 1, 2, 2, [-w_q^1, 0])$, stating that in each sequence of length 2, at least one value in $[-w_q^1, 0]$ occurs. On columns 1 (resp. 3 and 5) we replace the GCC constraint with one

stating that each value in $\{-w_q^1, \dots, -w_q^{cw_q-1}, \dots, -w_1^1, \dots, -w_1^{cw_1-1}\} \cup W$ (resp. $\{-z_q^1, \dots, -z_q^{cz_q-1}, \dots, -z_1^1, \dots, -z_1^{cz_1-1}\} \cup Z$ and $\{-y_q^1, \dots, -y_q^{cy_q-1}, \dots, -y_1^1, \dots, -y_1^{cy_1-1}\} \cup Y$) occurs at least once. We do not change the GCC constraints on columns 2 and 4. We show that this instance of SEQUENCEGCC has a solution iff there exists a 3D matching.

($\Rightarrow$) By reasoning similar to the proof of Theorem 4 (replacing ‘0’ with ‘a value in $[-w_q^1, 0]$ ’ when reasoning about columns 1, 3 and 5), we know that in the solution there are q many rows taking values $(w_i, 0, z_i, 0, y_i)$, possibly containing clones, and $m - q$ rows taking values (n, t, n', t, n'') where n, n', n'' are either 0 or some clone w'_i, z'_i , or y'_i (respectively). Furthermore, by the GCC constraint on the odd columns, we know that each value in $\{-w_q^1, \dots, -w_q^{cw_q-1}, w_q\}$ must occur exactly once. Since these values occur only in the domains of $\mathcal{M}_{i,1}$ for the cw_q many $s_i \in M$ that contain w_q , we know that each of these $\mathcal{M}_{i,1}$ must take a value in $\{-w_q^1, \dots, -w_q^{cw_q-1}, w_q\}$. Then, by the GCC constraint on the odd columns, we know that each value in $\{-w_{q-1}^1, \dots, -w_{q-1}^{cw_{q-1}-1}, w_{q-1}\}$ must occur exactly once. These values occur only in the domains of $\mathcal{M}_{i,1}$ for the $s_i \in M$ that contain w_{q-1} or w_q . However, the $\mathcal{M}_{i,1}$ for the $s_i \in M$ that contain w_q must take values in $\{-w_q^1, \dots, -w_q^{cw_q-1}, w_q\}$. Therefore, the $\mathcal{M}_{i,1}$ for the $s_i \in M$ that contain w_{q-1} must take a value in $\{-w_{q-1}^1, \dots, -w_{q-1}^{cw_{q-1}-1}, w_{q-1}\}$. Repeating this argument recursively until reaching the value t , we can restrict the effective domain of the odd positions of $\mathcal{M}_i$ for $s_i = (w_i, z_i, y_i) \in M$ to $\mathcal{M}_{i,1} \in \{-w_i^1, \dots, -w_i^{cw_i-1}, w_i\}$, $\mathcal{M}_{i,3} \in \{-z_i^1, \dots, -z_i^{cz_i-1}, z_i\}$, and $\mathcal{M}_{i,5} \in \{-y_i^1, \dots, -y_i^{cy_i-1}, y_i\}$. Therefore, we know that each row $\mathcal{M}_i$ for $s_i = (w_i, z_i, y_i) \in M$ either has the values $(w_i, 0, z_i, 0, y_i)$ or the values (w'_i, t, z'_i, t, y'_i) for some clones w'_i, z'_i, y'_i of w_i, z_i, y_i . Now, by the GCC constraints, we know that each value $w \in W$ occurs exactly once, as well as each value $z \in Z$ and each $y \in Y$. Since the possible values were chosen by taking elements from M , we know that $M' = \{s_i \mid \mathcal{M}_i = (w, 0, z, 0, y), w \in W, z \in Z, y \in Y\}$ is a 3D matching.

($\Leftarrow$) Let $M' \subseteq M$ be a 3D matching. We can fill $\mathcal{M}$ as follows. For each $(w_i, z_i, y_i) = s_i \in M'$, we let $\mathcal{M}_i = (w_i, 0, z_i, 0, y_i)$. For each $(w_i, z_i, y_i) = s_i \in M \setminus M'$ we let $\mathcal{M}_i = (w'_i, t, z'_i, t, y'_i)$ for some clones w'_i, z'_i, y'_i of w_i, z_i, y_i that have not been used before in the process of filling $\mathcal{M}$. We know there are enough different clones for this procedure. It is easy to verify that this instantiation of $\mathcal{M}$ satisfies all the constraints. $\square$

5 Fixed Parameter Tractable Cases

We have seen that propagating the REGULARGCC matrix constraint is NP-hard even under the strong restriction that either the number of values or the number of columns is bounded. We now identify three cases where we obtain tractability by bounding *two* parameters simultaneously. The three cases bound either the number of rows and the size of the automaton, or the number of columns and the product of the upper or lower bounds on the occurrences of values on a column.

Theorem 6. *Enforcing DC on REGULARGCC is fixed parameter tractable in $k = R + |Q|$, where R is the number of rows and $|Q|$ the maximum number of states in any row automaton.*

Proof: This follows observing that GCC over a sequence of a fixed size can be encoded in a DFA with polynomially many states, then swapping the rows with the columns, and appealing to Lemma 1 in the Appendix. $\square$

Theorem 7. *Enforcing DC on REGULARGCC is fixed parameter tractable in $k = C + B$, where C is the number of columns, and B is the maximum product of upper bounds specified on one column.*

Proof: Let $u_1, \dots, u_k$ be the upper bounds enforced in a given column on values $v_1, \dots, v_k$. Note that in the column constraints, for each value either both a lower and upper bound is specified, or neither. We can enforce the cardinality constraints consisting of the lower and upper bounds with a REGULAR constraint of size $u_1 \cdot \dots \cdot u_k = \mathcal{O}(B)$. The result now follows directly from Lemma 1 in the Appendix. $\square$

Theorem 8. *Enforcing DC on REGULARGCC is fixed parameter tractable in $k = C + B$, where C is the number of columns, and B is the maximum product of lower bounds specified on one column, and the GCC constraints in REGULARGCC only have lower bounds.*

Proof: Let $l_1, \dots, l_k$ be the lower bounds enforced in a given column on values $v_1, \dots, v_k$. We can enforce the cardinality constraints consisting of only these lower bounds with a REGULAR constraint of size $l_1 \cdot \dots \cdot l_k = \mathcal{O}(B)$. The result now follows directly from Lemma 1 in the Appendix. $\square$

We have shown that propagation of REGULARGCC is tractable if certain combinations of parameters of the problem are bounded like the number of rows and the size of the automaton. On the other hand, just bounding the number of rows on its own does not ensure tractability.

Theorem 9. *Enforcing BC on REGULARGCC is W[2]-hard in $k = R$ the number of rows with just 2 values.*

Proof: We reduce from p -HITTING-SET. Let $\mathcal{H} = (V, E)$ a hypergraph, where $V = \{v_1, \dots, v_{|V|}\}$ and $E = \{e_1, \dots, e_{|E|}\}$. We ask if there is a hitting set $S \subseteq V$ in $\mathcal{H}$ of cardinality k . We construct an instance $\mathcal{M}$ of REGULARGCC with $|V| + |E|$ columns and k rows on the alphabet $\{0, 1\}$. The REGULAR constraint accepts $|V|$ different words $w^1, \dots, w^{|V|}$ of length $|V| + |E|$. For any word w^v , the v th value is 1, and the remainder of the first $|V|$ values is 0. Also, for any word w^v and any $1 \leq j \leq |E|$, the j th value of w^v is 1 if $v \in e_j$, and is 0 otherwise. The GCC constraints we put on the columns are as follows. In the first $|V|$ columns we require exactly one 1. In the remaining columns, we require at least one 1. In this reduction, each row corresponds to one vertex that is being chosen for inclusion in the hitting set, and each column after the first $|V|$ to one hyperedge. The GCC constraints on the first $|V|$ columns ensure that no vertex is chosen in

two rows. The constraints on the last $|E|$ columns ensure that each hyperedge contains a vertex chosen for the hitting set. If there is a hyperedge all vertices of which are not included in the hitting set, the column corresponding to this hyperedge will contain 0's only, violating the GCC constraint of that column. We show that there exists a hitting set S in $\mathcal{H}$ of cardinality k iff the REGULARGCC matrix constraint has a solution.

($\Rightarrow$) Assume there exists a hitting set S in $\mathcal{H}$ of size k . We construct an assignment to REGULARGCC by matching one vertex $v \in S$ with each row (in any manner). If row i is matched to vertex v , we assign word w^v to row i . The fact that S contains k different vertices v ensures that k different words w^v are used. Also, since S is a hitting set, we know that for each hyperedge e_j there is at least one $v \in e_j \cap S$, and so each of the last $|E|$ columns contains at least one 1. Thus the GCC constraints are satisfied.

($\Leftarrow$) Suppose the REGULARGCC matrix constraint has a solution. We construct a hitting set S by taking all v such that w^v is a row in the solution. By the GCC constraints we know that the solution contains k different words w^v , so S is of size k . Now, to derive the contrary, assume there exists a hyperedge $e_j \in E$ such that $S \cap e_j = \emptyset$. Then the column corresponding to e_j (column $|V| + j$) contains only 0's. This violates GCC on this column, which is a contradiction. $\square$

6 A Practical Application

We illustrate the usefulness of the tractable cases identified here. In order to do so, we will describe a nurse scheduling problem that can be modelled (partially) using one of the identified tractable cases of REGULARGCC constraints.

Often, medical staff in a hospital are divided into different departments of some bounded size. The schedules of individual departments are mainly constrained by factors independent of other departments (e.g. the minimum number of staff on duty during different time shifts). Of course, there might be constraints expressing dependencies between schedules of different departments. Such constraints, however, are often more complicated than simple number restrictions on the staff on duty (e.g. for certain interdepartmental tasks, staff of different departments with particular skills needs to be on duty at the same time). As a result, these additional constraints need to be expressed by means of constraints separate to the REGULARGCC constraints anyway.

Concretely, we describe an example that corresponds to the result in Theorem 6, where the number of rows and the size of the row automata are bounded. Consider a hospital that can grow arbitrarily large in size, but where the different departments have a fixed maximum size. Since the departments typically have independent requirements on the number of staff on duty, the nurse scheduling problem can be divided into subproblems for every department, expressible by separate REGULARGCC constraints with a bounded number of rows. Also, shift rules typically express constraints on a bounded time window of n days (e.g. weekly or monthly requirements). Then, with row automata of size $\mathcal{O}(n \cdot s)$, for s the number of different shifts, we can express any requirement on (fixed or

moving) time windows of n (or less) days, (e.g. minimum and maximum number of working shifts in each calendar month, a day off after two night shifts in a row, never more than 5 working days in a row, for every 3 consecutive weekends at least one should be off duty).

Because the maximum time window n and the number of shifts s are typically fixed, we can use instances of the REGULARGCC constraint with bounded row automaton size and a bounded number of rows to solve rostering problems like the example sketched above. Theorem 6 then tells us that propagation for such instances of the REGULARGCC constraint is tractable.

7 Symmetry Breaking

A technique often used to reduce the number of isomorphic solutions and to increase propagation for a given constraint satisfaction problem is the introduction of symmetry breaking constraints [5, 15, 18]. Such constraints eliminate symmetrical solutions. One symmetry breaking constraint that is useful in the case of REGULARGCC requires the rows of the matrix to be lexicographically ordered (with respect to a given ordering on the domain values). Since the column constraints only impose bounds on the number of occurrences of values, swapping rows in any solution results in another solution, symmetric to the first solution. Additional symmetric solutions are often irrelevant, for instance when modelling nurse scheduling problems. Requiring the rows to be ordered eliminates such symmetric solutions, and the constraints that impose this requirement can increase propagation.

The complexity analysis of propagation of the REGULARGCC constraint presented above can be extended to the case where one constraint enforces simultaneously the REGULARGCC constraint and such symmetry breaking constraints [11]. The general NP-hardness and the $W[2]$ -hardness results above can directly be extended to hold for this combined setting as well. The fixed-parameter tractability results can also be extended to the symmetry breaking setting. For the result of Theorem 6, this can be done without modifying the parameter, by constructing an automaton, enforcing lexicographic row ordering, running over the matrix column per column. For the other fixed-parameter tractability results, this can be done when the parameter also includes the maximum domain size, by constructing such an automaton running over the matrix row per row. Further research is needed to extend the complexity analysis of propagating REGULARGCC for combinations with other symmetry breaking constraints.

8 Conclusions

The REGULARGCC constraint is useful to model a range of scheduling and rostering problems. We have provided a detailed picture of the conditions under which propagation of the REGULARGCC constraint on a matrix of decision variables is tractable. On the negative side, we proved that it is NP-hard to enforce even a weak level of consistency like bound consistency with just 2 values, just

4 states in the automaton, or just 5 columns to the matrix. On the positive side, we proved that it is fixed parameter tractable to enforce domain consistency when we bound the number of rows and the size of the automaton, or the number of columns and the product of the upper or lower bounds on the occurrences of values on a column. We note that such FPT results have found practical application recently. For example, Katsirelos, Narodytska and Walsh recently used such a FPT result to break all row and column symmetry [12]. Even more recently, Yip and Van Hentenryck have converted such a FPT result into an effective constraint propagator [19]. In future work, we therefore intend to convert the FPT results identified here into practical algorithms for solving rostering and scheduling problems.

Appendix: Regular² Constraint

Some of our fixed parameter tractability results exploit a mapping the REGULARGCC constraint onto a closely related global constraint. The REGULAR² constraint posts REGULAR constraints on every row and column of a matrix model. When we bound the number of columns and the number of states in the column automaton of such a constraint, propagation is polynomial.

Lemma 1. *Enforcing DC on REGULAR² is fixed parameter tractable in $k = C + |Q'|$, where C is the number of columns, and $|Q'|$ is the (maximum) size of the column automata.*

Proof: This proof is similar to Observation 2 in [10]. W.l.o.g. we assume that all row constraints are the same, and all column constraints are the same. Let $|Q|$ be the size of the row automaton. We can encode the matrix constraint on an $R \times C$ matrix $\mathcal{M}$ in a single DFA on the matrix stretched out to a single sequence of variables $\mathcal{M}_{1,1}, \dots, \mathcal{M}_{1,C}, \mathcal{M}_{2,1}, \dots, \mathcal{M}_{R,C}$. The state set of the automaton is $\{1, \dots, C\} \times Q \times (Q')^C$. In each state, the automaton keeps track of the column it is in, the current state $q' \in Q'$ for each column c , and the current state $q \in Q$ in the current row. The size of the automaton is $\mathcal{O}(C \cdot |Q| \cdot 2^{C \cdot \log |Q'|})$. Enforcing DC on a REGULAR constraint takes time polynomial in the size of the automaton [14, 16], so our algorithm runs in fixed parameter tractable time. $\square$

References

1. Beldiceanu, N., Carlsson, M., Flener, P., Pearson, J.: On Matrices, Automata, and Double Counting. In: Lodi, A., Milano, M., Toth, P. (eds.) CPAIOR 2010. LNCS, vol. 6140, pp. 10–24. Springer, Heidelberg (2010)
2. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: SLIDE: a useful special case of the CardPath constraint. In: Proc. of the 18th European Conf. on Artificial Intelligence (ECAI 2008). IOS Press (2008)
3. Bessiere, C., Hebrard, E., Hnich, B., Walsh, T.: The complexity of global constraints. In: Proc. of 19th National Conf. on AI. Association for Advancement of Artificial Intelligence (2004)

4. Bessière, C., Hebrard, E., Hnich, B., Walsh, T.: The Tractability of Global Constraints. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 716–720. Springer, Heidelberg (2004)
5. Crawford, J., Luks, G., Ginsberg, M., Roy, A.: Symmetry breaking predicates for search problems. In: Proc. of the 5th Int. Conf. on Knowledge Representation and Reasoning (KR 1996), pp. 148–159 (1996)
6. Flener, P., Frisch, A.M., Hnich, B., Kiziltan, Z., Miguel, I., Pearson, J., Walsh, T.: Breaking Row and Column Symmetries in Matrix Models. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 462–477. Springer, Heidelberg (2002)
7. Flener, P., Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Matrix modelling. Tech. Rep. APES-36-2001, APES group (2001); presented at Formul 2001 Workshop on Modelling and Problem Formulation, CP 2001
8. Flener, P., Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Matrix modelling: Exploiting common patterns in constraint programming. In: Proc. of the Int. Workshop on Reformulating Constraint Satisfaction Problems (2002); held alongside CP 2002
9. Hellsten, L., Pesant, G., van Beek, P.: A Domain Consistency Algorithm for the Stretch Constraint. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 290–304. Springer, Heidelberg (2004)
10. Katsirelos, G., Narodytska, N., Quimper, C., Walsh, T.: Global matrix constraints. In: Proc. of the Int. Workshop on Constraint Modelling and Reformulation, pp. 27–41 (2011), held alongside CP 2011
11. Katsirelos, G., Narodytska, N., Walsh, T.: Combining Symmetry Breaking and Global Constraints. In: Oddi, A., Fages, F., Rossi, F. (eds.) CSCLP 2008. LNCS, vol. 5655, pp. 84–98. Springer, Heidelberg (2009)
12. Katsirelos, G., Narodytska, N., Walsh, T.: On the Complexity and Completeness of Static Constraints for Breaking Row and Column Symmetry. In: Cohen, D. (ed.) CP 2010. LNCS, vol. 6308, pp. 305–320. Springer, Heidelberg (2010)
13. Maher, M.J., Narodytska, N., Quimper, C.-G., Walsh, T.: Flow-Based Propagators for the SEQUENCE and Related Global Constraints. In: Stuckey, P.J. (ed.) CP 2008. LNCS, vol. 5202, pp. 159–174. Springer, Heidelberg (2008)
14. Pesant, G.: A Regular Language Membership Constraint for Finite Sequences of Variables. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 482–495. Springer, Heidelberg (2004)
15. Puget, J.F.: On the Satisfiability of Symmetrical Constrained Satisfaction Problems. In: Komorowski, J., Raś, Z.W. (eds.) ISMIS 1993. LNCS, vol. 689, pp. 350–361. Springer, Heidelberg (1993)
16. Quimper, C.-G., Walsh, T.: Global Grammar Constraints. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 751–755. Springer, Heidelberg (2006)
17. Walsh, T.: Constraint Patterns. In: Rossi, F. (ed.) CP 2003. LNCS, vol. 2833, pp. 53–64. Springer, Heidelberg (2003)
18. Walsh, T.: General Symmetry Breaking Constraints. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 650–664. Springer, Heidelberg (2006)
19. Yip, J., Hentenryck, P.V.: Symmetry breaking via lexleader feasibility checkers. In: Walsh, T. (ed.) Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011). pp. 687–692. IJCAI/AAAI (2011)