Breaking Symmetry with Different Orderings

Nina Narodytska and Toby Walsh

NICTA and UNSW,
Sydney, Australia
email: {nina.narodytska,toby.walsh}@nicta.com.au

Abstract. We can break symmetry by eliminating solutions within each symmetry class. For instance, the Lex-Leader method eliminates all but the smallest solution in the lexicographical ordering. Unfortunately, the Lex-Leader method is intractable in general. We prove that, under modest assumptions, we cannot reduce the worst case complexity of breaking symmetry by using other orderings on solutions. We also prove that a common type of symmetry, where rows and columns in a matrix of decision variables are interchangeable, is intractable to break when we use two promising alternatives to the lexicographical ordering: the Gray code ordering (which uses a different ordering on solutions), and the Snake-Lex ordering (which is a variant of the lexicographical ordering that re-orders the variables). Nevertheless, we show experimentally that using other orderings like the Gray code to break symmetry can be beneficial in practice as they may better align with the objective function and branching heuristic.

1 Introduction

Symmetry occurs in many combinatorial problems. For example, when coloring a graph, we can permute the colors in any proper coloring. Symmetry can also be introduced by modelling decisions (e.g. using a set of finite domain variables to model a set of objects will introduce the symmetries that permute these variables). A common method to deal with symmetry is to add constraints which eliminate symmetric solutions (e.g. [1–13]). Unfortunately, breaking symmetry by adding constraints to eliminate symmetric solutions is intractable in general [2]. More specifically, deciding if an assignment is the smallest in its symmetry class for a matrix with row and column symmetries is NP-hard, supposing rows are appended together and compared lexicographically. There is, however, nothing special about appending rows together or comparing solutions lexicographically. We could use *any* total ordering over assignments. For example, we could break symmetry with the Gray code ordering. That is, we add constraints that eliminate symmetric solutions within each symmetry class that are not smallest in the Gray code ordering. This is a total ordering over assignments used in error correcting codes. Such an ordering may pick out different solutions in each symmetry class, reducing the conflict between symmetry breaking, problem constraints, objective function and the branching heuristic. The Gray code ordering has some properties that may be useful for symmetry breaking. In particular, neighbouring assignments in the ordering only differ at one position, and flipping one bit reverses the ordering of the subsequent bits.

As a second example, we can break row and column symmetry with the Snake-Lex ordering [14]. This orders assignments by lexicographically comparing vectors constructed by appending the variables in the matrix in a "snake like" manner. The first row is appended to the reverse of the second row, and this is then appended to the third row, and then the reverse of the fourth row and so on. As a third example, we can break row and column symmetry by ordering the rows lexicographically and the columns with a multiset ordering [15]. This is incomparable to the Lex-Leader method.

We will argue theoretically that breaking symmetry with a different ordering over assignments cannot improve the worst case complexity. However, we also show that other orderings can be useful in practice as they pick out different solutions in each symmetry class. Our argument has two parts. We first argue that, under modest assumptions which are satisfied by the Gray code and Snake-Lex orderings, we cannot reduce the computational complexity from that of breaking symmetry with the lexicographical ordering which considers variables in a matrix row-wise. We then prove that for the particular case of row and column symmetries, breaking symmetry with the Gray code or Snake-Lex ordering is intractable (as it was with the lexicographical ordering). Many dynamic methods for dealing with symmetry are equivalent to posting symmetry breaking constraints "on the fly" (e.g. [16–24]).

Hence, our results have implications for such dynamic methods too.

2 Background

A symmetry of a set of constraints S is a bijection σ on complete assignments that maps solutions of S onto other solutions of S. Many of our results apply to the more restrictive definition of symmetry which considers just those bijections which map individual variable-value pairs [25]. However, this more general definition captures also conditional symmetries [26]. In addition, a few of our results require this more general definition. In particular, Theorem 3 only holds for this more general definition[1]. The set of symmetries form a group under composition. Given a symmetry group Σ, a subset Π generates Σ iff any $\sigma \in \Sigma$ is a composition of elements from Π. A symmetry group Σ partitions the solutions into symmetry classes (or orbits). We write $[A]_\Sigma$ for the symmetry class of solutions symmetric to the solution A. Where Σ is clear from the context, we write $[A]$. A set of symmetry breaking constraints is *sound* iff it leaves at least one solution in each symmetry class, and *complete* iff it leaves at most one solution in each symmetry class.

We will study what happens to symmetries when problems are reformulated onto equivalent problems. For example, we might consider the Boolean form of a problem in which $X_i = j$ maps onto $Z_{ij} = 1$. Two sets of constraints, S and T over possibly different variables are *equivalent* iff there is a bijection between their solutions. Suppose U_i and V_i for $i \in [1, k]$ are partitions of the sets U and V into k subsets. Then the two partitions are *isomorphic* iff there are bijections $\pi : U \mapsto V$ and $\tau : [1, k] \mapsto [1, k]$ such that $\pi(U_i) = V_{\tau(i)}$ for $i \in [1, k]$ where $\pi(U_i) = \{\pi(u) \mid u \in U_i\}$. Two symmetry groups Σ and Π of constraints S and T respectively are *isomorphic* iff S and T are equivalent, and their symmetry classes of solutions are isomorphic.

[1] We thank an anonymous reviewer for pointing this out.

3 Using Other Orderings

The Lex-Leader method [2] picks out the lexicographically smallest solution in each symmetry class. For every symmetry σ, it posts a lexicographical ordering constraint: $\langle X_1, \ldots, X_n \rangle \leq_{\text{lex}} \sigma(\langle X_1, \ldots, X_n \rangle)$ where X_1 to X_n is some ordering on the variables in the problem. Many static symmetry breaking constraints can be derived from such Lex-Leader constraints. For example, DOUBLELEX constraints to break row and column symmetry can be derived from them [27]. As a second example, PRECEDENCE constraints to break the symmetry due to interchangeable values can also be derived from them [5, 8]. Efficient algorithms exist to propagate such lexicographical constraints (e.g. [28–30]).

We could, however, break symmetry by using another ordering on assignments like the Gray code ordering. We define the Gray code ordering on Boolean variables. For each symmetry σ, we could post an ordering constraint:

$$\langle X_1, \ldots, X_n \rangle \leq_{\text{Gray}} \sigma(\langle X_1, \ldots, X_n \rangle)$$

Where the k-bit Gray code ordering is defined recursively as follows: 0 is before 1, and to construct the $k+1$-bit ordering, we append 0 to the front of the k-bit ordering, and concatenate it with the reversed k-bit ordering with 1 appended to the front. For instance, the 4-bit Gray code orders assignments as follows:

$$0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100,$$
$$1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000$$

The Gray code ordering is well founded. Hence, every set of complete assignments will have a smallest member under this ordering. This is the unique complete assignment in each symmetry class selected by posting such Gray code ordering constraints. Thus breaking symmetry with Gray code ordering constraints is sound and complete.

Proposition 1 *Breaking symmetry with Gray code ordering constraints is sound and complete.*

In Section 6, we propose a propagator for the Gray code ordering constraint. We cannot enforce the Gray code ordering by ordering variables and values, and using a lexicographical ordering constraint. For example, we cannot map the 2-bit Gray code onto the lexicographical ordering by simply re-ordering variables and values. To put it another way, no reversal and/or inversion of the bits in the 2-bit Gray code will map it onto the lexicographical ordering. The 2-bit Gray code orders 00, 01, 11 and then 10. We can invert the first bit to give: 10, 11, 01 and then 00. Or we can invert the second bit to give: 01, 00, 10, and then 11. Or we can invert both bits to give: 11, 10, 00, and then 01. We can also reverse the bits to give: 00, 10, 11, and then 01. And we can then invert one or both bits to give: 10, 00, 01, and then 10; or 01, 11, 10, and then 00; or 11, 01, 00, and then 10. Note that none of these re-orderings and inversions is the 2-bit lexicographical ordering: 00, 01, 10, and then 11.

4 Complexity of Symmetry Breaking

We will show that, under some modest assumptions, we cannot make breaking symmetry computationally easier by using a new ordering like the Gray code ordering. Our argument breaks into two parts. First, we observe how the symmetry of a problem changes when we reformulate onto an equivalent problem. Second, we argue that we can map onto an equivalent problem on which symmetry breaking is easier.

Proposition 2 *If a set of constraints S has a symmetry group Σ, S and T are equivalent sets of constraints, π is any bijection between solutions of S and T, and $\Pi \subseteq \Sigma$ then:*

(a) $\pi\Sigma\pi^{-1}$ is a symmetry group of T;
(b) Σ and $\pi\Sigma\pi^{-1}$ are isomorphic symmetry groups;
(c) if Π generates Σ then $\pi\Pi\pi^{-1}$ generates $\pi\Sigma\pi^{-1}$.

We will use this proposition to argue that symmetry breaking with any ordering besides the lexicographical ordering is intractable. We consider only *simple* orderings. In a simple ordering, we can compute the position of any assignment in the ordering in polynomial time, and given any position in the ordering we can compute the assignment at this position in polynomial time. We now give our main result.

Proposition 3 *Given any simple ordering $\preceq$, there exists a symmetry group such that deciding if an assignment is smallest in its symmetry class according to $\preceq$ is NP-hard.*

Proof: Deciding if an assignment is smallest in its symmetry class according to $\leq_{\text{lex}}$ is NP-hard [2]. Since $\preceq$ and $\leq_{\text{lex}}$ are both simple orderings, there exist polynomial functions f to map assignments onto positions in the $\leq_{\text{lex}}$ ordering, and g to map positions in the $\preceq$ ordering to assignments. Consider the mapping π defined by $\pi(A) = g(f(A))$. Now π is a permutation that is polynomial to compute which maps the total ordering of assignments of $\leq_{\text{lex}}$ onto that for $\preceq$. Similarly, π^{-1} is a permutation that is polynomial to compute which maps the total ordering of assignments of $\preceq$ onto that for $\leq_{\text{lex}}$. Let Σ_{rc} be the row and column symmetry group. By Theorem 2, the problem of finding the lexicographical least element of each symmetry class for Σ_{rc} is equivalent to the problem of finding the least element of each symmetry class according to $\preceq$ for $\pi\Sigma_{rc}\pi^{-1}$. Thus, for the symmetry group $\pi\Sigma_{rc}\pi^{-1}$ deciding if an assignment is smallest in its symmetry class according to $\preceq$ is NP-hard. □

It follows that there exists an infinite family of symmetry groups such that checking a constraint which is only satisfied by the smallest member of each symmetry class is NP-hard. Note that the Gray code and Snake-Lex orderings are simple. Hence, breaking symmetry with either ordering is NP-hard for some symmetry groups. Note that we are not claiming that deciding if an assignment is smallest in its symmetry class is NP-complete. First, we would need to worry about the size of the input (since we are considering the much larger class of symmetries that act on complete assignments rather than on literals). Second, to decide that an assignment is the smallest, we are also answering a complement problem (there is *no* smaller symmetric assignment). This will take us to DP-completeness or above.

5 Breaking Matrix Symmetry

We next consider a common type of symmetry. In many models, we have a matrix of decision variables in which the rows and columns are interchangeable [31–33]. We will show that breaking row and column symmetry specifically is intractable with the Gray code and the Snake-Lex orderings, as it is with the lexicographical ordering that considers the variables in a row-wise order.

Proposition 4 *Finding the smallest solution up to row and column symmetry for the Snake-Lex ordering is NP-hard.*

Proof: We reduce from the problem of finding the Lex-Leader solution of a matrix B. Let B be an $n \times m$ matrix of Boolean values. W.l.o.g. we assume B does not contain a row of only ones since any such row can be placed at the bottom of the matrix. We embed B in the matrix M such that finding $\sigma(M)$, denoted M', the smallest row and column symmetry of M in the Snake-Lex ordering is equivalent to finding the Lex-Leader of B. We ensure that even rows in the Snake-Lex smallest symmetric solution of M are taken by dummy identical rows. Then in odd rows, where Snake-Lex moves from the left to the right along a row like Lex does, we embed the Lex-Leader solution of B.

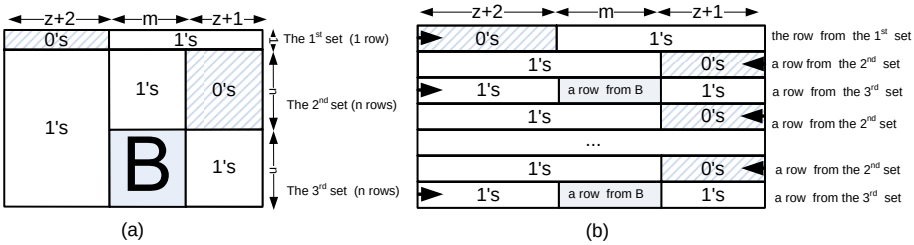


Fig. 1. (a) Construction of M (b) Partial construction of M'. The first and all even rows are fixed.

Let z be the maximum number of zeros in any row of B. We construct M with $2n + 1$ rows and $(z + 2) + (z + 1) + m$ columns so that it contains three sets of rows. The first set consists of a single row that contains $z + 2$ zeros followed by $(z + 1) + m$ ones. The second set contains n identical rows with $z + 2 + m$ ones followed by $z + 1$ zeros in each row. The third set of rows contains n rows such that at the ith row the first $(z+2)$ positions are ones, the next m positions are the ith row from B and the last $z+1$ positions are ones again. Schematic representation of M is shown at Figure 1(a).

We determine positions of rows and columns that must be fixed in M' up to permutation of identical rows and columns. The first row of M has to be the first row of M' as no other row contains $z + 2$ zeros. Note that this also fixes the position of columns from 1 to $z + 2$ in M to be the first columns in the M'. Note also that these columns are identical and each of them contains the zero in the first row only.

One of the rows in the second set has to be the second row of M', as none of the rows that embed rows from B contains $z+1$ zeros. As we move from the right to the left on even rows, this also makes sure that last $z+1$ columns from M must be the last columns in M'. We summarise that at this point the first two rows are fixed and the first $z+2$ columns and the last $z+1$ columns in M' must be equal to a permutation of the first $z+2$ identical columns and the last $z+1$ identical columns in M, respectively.

By assumption, B does not contain rows with all ones. Moreover, only rows that embed rows from B can have the value zero at columns from $(z+2)+1$ to $(z+2)+m$ in M'. Hence, a row from the third set that embeds a row from B has to be the third row in M'. We do not specify which row it is at this point. The fourth row has to be again a row from the second set as any of remaining rows from the second set has $z+1$ zeros in the last $z+1$ columns in M' while any row that embeds B has at most z zeros. We can repeat this argument for the remaining rows. A schematic representation of the positions of rows from the first and second sets are shown in Figure 1(b). Note that the first and all even rows in M' are fixed. The only part of M' not yet specified is the ordering of odd rows of m columns from $(z+2)+1$ to $(z+2)+m$. These are exactly all rows from B. Hence, finding M' is reduced to ordering of this set of rows and columns that embed B. Now, all columns from $(z+2)+1$ to $(z+2)+m$ are interchangeable, all odd rows except the first are interchangeable, and all elements of M' except elements of B are fixed by construction. As the Snake-Lex ordering goes from the left to the right on odd rows like the Lex ordering, finding M' is equivalent to finding the Lex-Leader of B $\square$

To show that finding the smallest row and column symmetry in the Gray code ordering is NP-hard, we need a technical lemma about cloning columns in a matrix. We use rowwise ordering in a matrix. Suppose we clone each column in a $n \times m$ Boolean matrix B to give the matrix B^c. Let B^c_{gl} be the smallest row and column symmetry of B^c in the Gray code ordering.

Lemma 1 *Any original column of B is followed by its clone in B^c_{gl} ignoring permutation of identical original columns.*

Proof: By contradiction. Suppose there exists an element $B^c_{gl}[j, i+1]$ such that the original column i and the next column $i+1$ are different at the jth row. We denote by k the $[j, i+1]$ element of B^c_{gl} in its row-wise linearization. We ignore the rows from $j+1$ to n at this point as they are not relevant to this discrepancy.

Each pair of columns coincide on the first j rows for the first $i-1$ columns and on the first $j-1$ rows for the columns from i to m. We conclude that (1) i is odd and $i+1$ is even; (2) the number of ones between the first and the $(k-2)$th positions in the linearization of B^c_{gl} is even as each value is duplicated; (3) the clone of the ith column cannot be among the first $i-1$ columns as each such column is followed by its clone by assumption. Hence, the clone of the ith column is among columns from $i+2$ to m.

Suppose the clone of the ith column is the pth column. Note that the pth column must coincide with the $i+1$th column at the first $j-1$ rows. We consider two cases. In the first case, $B^c_{gl}[j, i] = 1$ and $B^c_{gl}[j, i+1] = 0$. Note that the total number of ones at the positions from 1 to $k-1$ is odd as we have one in the position $k-1$ and the number of ones in the first $k-2$ positions is even. Next we swap the $(i+1)$th and pth

columns in B^c_{gl}. This will not change the first $k-1$ elements in the linearization as the pth column must coincide with the $i+1$th column at the first $j-1$ rows. Moreover, this swap puts 1 in position k. As the number of ones up to the $(k-1)$th position is odd then 1 goes before 0 at position k in the Gray code ordering. Hence, by swapping the $(i+1)$th and pth columns we obtain a matrix that is smaller than B^c_{gl} in the Gray code ordering. This is a contradiction. In the second case, $B^c_{gl}[j,i] = 0$ and $B^c_{gl}[j,i+1] = 1$. Note that the total number of ones at positions 1 to $k-1$ in the linearization is even as we have zero at the position $k-1$ and the number of ones in the first $k-2$ positions is even. Therefore, 0 precedes 1 at position k in the Gray code ordering. By swapping the $(i+1)$th and pth columns we obtain a matrix that is smaller than B^c_{gl} in the Gray code ordering as 0 appears at the position k instead of 1. This is a contradiction. □

Proposition 5 *Finding the smallest solution up to row and column symmetry for the Gray code ordering is NP-hard.*

Proof: We again reduce from the problem of finding the Lex-Leader solution of a matrix B. We clone every column of B and obtain a new matrix B^c. Let B^c_{gl} be the smallest row and column symmetry of B^c in the Gray code ordering. Lemma 1 shows that each original column is followed by its clone in B^c_{gl}. Next we delete all clones by removing every second column. We call the resulting matrix B_l. We prove that B_l is the Lex-Leader of B by contradiction. Suppose there exists a matrix M which is the Lex-Leader of B that is different from B_l. Hence, M is also the Lex-Leader of B_l. We find the first element $M[j,i]$ where $B_l[j,i] \neq M[j,i]$ in the row-wise linearization of M and B_l, so that $B_l[j,i] = 1$ and $M[j,i] = 0$. We denote by k the position of the $[j,i]$ element of M in its row-wise linearization. We clone each column of M once and put each cloned column right after its original column. We obtain a new matrix M^c. We show that M^c is smaller than B^c_{gl} in the Gray code ordering to obtain a contradiction.

As $B_l[j,i] = 1$ and $M[j,i] = 0$ then $B^c_{gl}[j,2i-1] = 1$ and $M^c[j,2i-1] = 0$ because the matrices B^c_{gl} and M^c are obtained from B_l and M by cloning each column and putting each clone right after its original column. As B_l and M coincide on the first $k-1$ positions then B^c_{gl} and M^c coincide in the first $2k-2$ positions. By transforming B_l and M to B^c_{gl} and M^c, we duplicated each value in positions from 1 to $k-1$. Hence, the total number of ones in positions from 1 to $2k-2$ in $B^c_{gl}[j,i]$ and $M^c[j,i]$ is even. Therefore, the value zero precedes the value one at position $2k-1$ in the Gray code ordering. By assumption, the value in the position $2k-1$ in B^c_{gl}, which is $B^c_{gl}[j,2i-1]$, is 1, and the position $2k-1$ in M^c, which is $M^c[j,2i-1]$, is 0. Hence, M^c is smaller than B^c_{gl} in the Gray code ordering. □

We conjecture that row and column symmetry will be intractable to break for other simple orderings. However, each such ordering may require a new proof.

6 Other Symmetry Breaking Constraints

Despite these negative theoretical results, there is still the possibility for other orderings on assignments to be useful when breaking symmetry in practice. It is interesting therefore to develop propagation algorithms for different orderings. Propagation algorithms are used to prune the search space by enforcing properties like domain consistency. A

constraint is *domain consistent* (*DC*) iff when a variable is assigned any value in its domain, there exist compatible values in the domains of the other variables.

6.1 Gray Code Constraint

We give an efficient encoding for the new global constraint $Gray([X_1, \ldots, X_n], [Y_1, \ldots, Y_n])$ that ensures $\langle X_1, \ldots, X_n \rangle$ is before or equal in position to $\langle Y_1, \ldots, Y_n \rangle$ in the Gray code ordering where X_i and Y_j are 0/1 variables. We encode the transition relation of an automaton with 0/1/-1 state variables, Q_1 to Q_{n+1} that reads a sequence $\langle X_1, Y_1, \ldots, X_n, Y_n \rangle$ and ensures that the two sequences are ordered appropriately. We consider the following decomposition where $1 \leq i \leq n$:

$$Q_1 = 1,\ Q_i \neq 1 \vee X_i \leq Y_i,\ Q_i \neq -1 \vee X_i \geq Y_i,$$
$$X_i = Y_i \vee Q_{i+1} = 0,\ X_i = 0 \vee Y_i = 0 \vee Q_{i+1} = -Q_i.$$

We can show that this decomposition not only preserves the semantics of the constraint but also does not hinder propagation.

Proposition 6 *Unit propagation on this decomposition enforces domain consistency on* $Gray([X_1, \ldots, X_n], [Y_1, \ldots, Y_n])$ *in* $O(n)$ *time.*

Proof: (Correctness) $Q_i = 0$ as soon as the two vectors are ordered correctly. $Q_i = 1$ iff X_i and Y_i are ordered in the Gray code ordering with 0 before 1. $Q_i = -1$ iff the ith bits, X_i and Y_i are ordered in the Gray code ordering with 1 before 0. Q_{i+1} stays the same polarity as Q_i iff $X_i = Y_i = 0$ and flips polarity iff $X_i = Y_i = 1$.

(Completeness) This follows from the completeness of CNF encoding of the corresponding automaton [34] and the fact that unit propagation on this set of constraints enforces DC on a table constraint that encodes the transition relation.

(Complexity) There are $O(n)$ disjuncts in the decomposition. Hence unit propagation takes $O(n)$ time. In fact, it is possible to show that the *total* time to enforce DC down a branch of the search tree is $O(n)$. □

Note that this decomposition can be used to break symmetry with the Gray code ordering in a SAT solver.

6.2 Snake-Lex Constraint

For row and column symmetry, we can break symmetry with the DOUBLELEX constraint that lexicographically orders rows and columns, or the SNAKELEX constraint. This is based on the smallest row and column permutation of the matrix according to an ordering on assignments that linearizes the matrix in a snake-like manner [14]. The (columnwise) SNAKELEX constraint can be enforced by a conjunction of $2m - 1$ lexicographical ordering constraints on pairs of columns and $n-1$ lexicographical constraints on pairs of intertwined rows. To obtain the rowwise SNAKELEX constraint, we transpose the matrix and then order as in the columnwise SNAKELEX. Note that DOUBLELEX and SNAKELEX *only break a subset* of the row and colum symmetries.

However, they are very useful in practice. It was shown in [12], that enforcing *DC* on the DOUBLELEX constraint is NP-hard. Hence we typically decompose it into separate row and column constraints. Here, we show that enforcing *DC* on the SNAKELEX constraint is also NP-hard. It is therefore also reasonable to propagate SNAKELEX by decomposition.

Proposition 7 *Enforcing DC on the* SNAKELEX *constraint is NP-hard.*

Proof: (Sketch) A full proof is in the online technical report. Let X be a n by m matrix of Boolean variables. The main idea is to embed X in to a specially constructed matrix in such a way that enforcing *DC* on the DOUBLELEX constraint on X (which we already know is NP-hard) is equivalent to enforcing *DC* on the SNAKELEX constraint on this larger matrix. □

7 Experimental Results

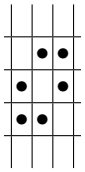
We tested two hypotheses that provide advice to the modeller when breaking symmetry.

1. other orderings besides the lexicographical ordered can be effective when breaking symmetry in practice;
2. symmetry breaking should align with the branching heuristic, and with the objective function.

All our experiments report the time to find an optimal solution *and* prove it optimal. We believe that optimisation is often a more realistic setting in which to illustrate the practical benefits of symmetry breaking, than satisfaction experiments which either find one or all solutions. Breaking symmetry in optimisation problems is important as we must traverse the whole search space when proving optimality. All our experiments used the BProlog 7.7 constraint solver. This solver took second place in the ASP 2011 solver competition. The three sets of experiments took around one CPU month on a MacBook Pro with an Intel Core i5 2 core 2.53 GHz processor, with 4GB of memory. The three domains were chosen as representative of optimisation problems previously studied in symmetry breaking. We observed similar results in these as well as other domains.

7.1 Maximum density still life problem

This is prob032 in CSPLib [35]. This problem arises in Conway's Game of Life, and was popularized by Martin Gardner. Given a n by n submatrix of the infinite plane, we want to find and prove optimal the pattern of maximum density which does not change from generation to generation. For example, an optimal solution for $n = 3$ is:



This is a still life as every live square has between 2 and 3 live neighbours, and every dead square does not have 3 live neighbours. We use the simple 0/1 constraint model from [36]. This problem has the 8 symmetries of the square as we can rotate or reflect any still life to obtain a new one. Bosch and Trick argued that *"... The symmetry embedded in this problem is very strong, leading both to algorithmic insights and algorithmic difficulties..."*.

Our first experiment used the default search strategy to find and prove optimal the still life of maximum density for a given n. The default strategy instantiates variables row-wise across the matrix. Our goal here is to compare the different symmetry breaking methods with an "out of the box" solver. We then compare the impact of the branching heuristic on symmetry breaking. We broke symmetry with either the lexicographical or Gray code orderings, finding the smallest (lex, gray) or largest (anti-lex, anti-gray) solution in each symmetry class. In addition, we linearized the matrix either row-wise (row), column-wise (col), snake-wise along rows (snake), snake-wise along columns (col-snake), or in a clockwise spiral (spiral). Table 1 gives results for the 20 different symmetry breaking methods constructed by using 1 of the 4 possible solution orderings and the 5 different linearizations, as well with no symmetry breaking (none).

Symmetry breaking	$n = 4$	5	6	7	8
none	176	1,166	12,205	231,408	5,867,694
gray row	91	446	5,702	123,238	2,507,747
anti-lex row	84	424	5,473	120,112	2,416,266
anti-gray col-snake	68	500	5,770	72,691	2,332,085
gray spiral	86	541	6,290	120,051	2,311,854
gray snake	80	477	5,595	120,601	2,264,184
anti-lex col-snake	79	660	4,735	66,371	2,254,325
anti-lex spiral	81	507	6,174	119,262	2,241,660
anti-lex col	74	718	3,980	68,330	2,215,936
anti-lex snake	68	457	5,379	117,479	2,206,189
lex spiral	48	434	4,025	90,289	2,028,624
lex col-snake	77	359	5,502	76,400	2,003,505
lex col	80	560	4,499	83,995	2,017,935
lex row	33	406	2,853	87,781	1,982,698
lex snake	35	407	2,965	86,331	1,980,498
anti-gray col	70	522	5,666	75,930	1,925,613
gray col	65	739	3,907	87,350	1,899,887
gray col-snake	62	693	3,833	82,736	1,880,506
anti-gray row	*26*	269	2,288	38,476	1,073,659
anti-gray spiral	27	279	2,404	40,224	1,081,006
anti-gray snake	28	*262*	*2,203*	*38,383*	*1,059,704*

Table 1. Backtracks required to find and prove optimal the maximum density still life of size n by n using the default branching heuristic. Column winner is in *emphasis*.

We make some observations about these results. First, the Lex-Leader method (lex row) is beaten by many methods. For example, the top three methods all use the anti-Gray code ordering. Second, lex tends to work better than anti-lex, but anti-gray better than gray. We conjecture this is because anti-gray tends to align better with the maximization objective than gray, but anti-lex is too aggressive as the maximum density still life can have more dead cells than alive cells. Third, although we eliminate all 7 non-

identity symmetries, the best method is only about a factor 6 faster than not breaking symmetry at all.

To explore the interaction between symmetry breaking and the branching heuristic, we report results in Table 2 using branching heuristics besides the default row-wise variable ordering. We used the best symmetry breaking method for the default row-wise branching heuristic (anti-gray snake), the worst symmetry breaking method for the default branching heuristic (gray row), a standard method (lex row), as well as no symmetry breaking (none). We compared the default branching heuristic (row heuristic) with branching heuristics that instantiate variables column-wise (col heuristic), snake-wise along rows (snake heuristic), snake-wise along columns (col-snake heuristic), in a clockwise spiral from top left towards the middle (spiral-in heuristic), in an anti-clockwise spiral from the middle out to the top left (spiral-out heuristic), by order of degree (degree heuristic), and by order of the number of attached constraints (constr heuristic). Note that there is no value in reporting results for domain ordering heuristics like fail-first as domains sizes are all binary.

Branching/SymBreak	none	gray row	lex row	anti-gray snake
spiral-out heuristic	196,906,862	24,762,297	194,019,848	222,659,696
spiral-in heuristic	65,034,993	18,787,751	12,662,207	9,292,164
constr heuristic	*5,080,541*	2,816,355	3,952,445	8,590,077
degree heuristic	6,568,195	2,024,955	6,528,018	7,053,908
col-snake heuristic	5,903,851	*1,895,920*	1,849,702	2,127,122
col heuristic	5,867,694	2,212,104	*1,634,016*	1,987,864
snake heuristic	5,903,851	1,868,303	2,043,473	1,371,200
row heuristic	5,867,694	2,507,747	1,982,698	**1,059,704**

Table 2. Backtracks required to find the 8 by 8 still life of maximum density and prove optimality for different branching heuristics and symmetry breaking constraints. Overall winner is in **bold**.

We make some observations about these results. First, the symmetry breaking method with the best overall performance (anti-gray snake + row heuristic) had the worst performance with a different branching heuristic (anti-gray snake + spiral-out heuristic). Second, we observed good performance when the branching heuristic aligned with the symmetry breaking (e.g. anti-gray snake + snake heuristic). Third, a bad combination of branching heuristic and symmetry breaking constraints (e.g. anti-gray snake + spiral-out heuristic) was worse than all of the branching heuristics with no symmetry breaking constraints. Fourth, the default row heuristic was competitive. It was best or not far from best in every column.

7.2 Low autocorrelation binary sequences

This is prob005 in CSPLib [35]. The goal is to find the binary sequence of length n with the lowest autocorrelation. We used a standard model from one of the first studies into symmetry breaking [19]. This model contains a triangular matrix of 0/1 decision variables, in which the sum of the kth row equals the kth autocorrelation. Table 3 reports results to find the sequence of lowest autocorrelation and prove it optimal. We used the default variable ordering heuristic (left2right) that instantiates variables left to right

from the beginning of the sequence to the end. The model has 7 non-identity symmetries which leave the autocorrelation unchanged. We can reverse the sequence, we can invert the bits, we can invert just the even bits, or we can do some combination of these operations. We broke all 7 symmetries by posting the constraints that, within its symmetry class, the sequence is smallest in the lexicographical or Gray code orderings (lex, gray) or largest (anti-lex, anti-gray). In addition, we also considered symmetry breaking constraints that took the variables in reverse order from right to left (rev), alternated the variables from both ends inwards to the middle (outside-in), and from the middle out to both ends (inside-out).

Symmetry breaking	$n = 12$	14	16	18	20	22	24
none	2,434	9,487	36,248	126,057	474,915	1,725,076	7,447,186
anti-gray outside-in	2,209	6,177	18,881	92,239	310,473	1,223,155	4,966,068
gray outside-in	1,351	5,040	19,152	68,272	350,790	903,441	4,526,114
lex outside-in	869	3,057	11,838	43,669	262,935	557,790	3,330,931
gray	704	2,400	10,158	36,854	158,080	468,317	3,048,723
lex	707	2,408	10,178	36,885	158,132	468,390	3,047,241
gray rev	699	1,790	9,892	25,551	147,911	329,897	2,706,466
anti-lex outside-in	1,262	2,704	14,059	67,848	179,219	544,116	2,579,981
anti-gray	1,036	2,226	9,889	45,375	167,916	606,977	2,436,236
anti-lex	1,522	3,087	10,380	51,162	281,789	920,543	2,415,736
lex rev	634	1,751	7,601	23,218	127,438	299,877	2,160,463
anti-lex rev	549	1,707	9,398	32,638	117,367	398,822	2,092,787
gray inside-out	662	1,582	6,557	25,237	89,365	248,135	1,667,262
lex inside-out	640	1,549	6,478	25,049	88,978	247,558	1,665,054
anti-gray rev	1,007	1,661	6,894	29,689	86,198	312,038	1,422,693
anti-gray inside-out	*412*	1,412	5,934	22,942	*82,673*	*245,259*	1,271,986
anti-lex inside-out	629	*1,320*	*4,558*	*19,811*	138,337	291,050	*927,321*

Table 3. Backtracks required to find the n bit binary sequence of lowest autocorrelation and prove optimality with the default branching heuristic.

We make some observations about these results. First, the best two symmetry breaking methods both look at variables starting from the middle and moving outwards to both ends (inside-out). By comparison, symmetry breaking constraints that reverse this ordering of variables (outside-in) perform poorly. We conjecture this is because the middle bits in the sequence are more constrained, appearing in more autocorrelations, and so are more important to decide early in search. Second, although we only eliminate 7 symmetries, the best method offers a factor of 8 improvement in search over not breaking symmetry.

To explore the interaction between symmetry breaking and branching heuristics, we report results in Table 4 to find the optimal solution and prove optimality using different branching heuristics. We used the best two symmetry breaking methods for the default left to right branching heuristic (anti-gray inside-out, and anti-lex inside-out), the worst symmetry breaking method for the default branching heuristic (anti-gray outside-in), a standard symmetry breaking method (lex), the Gray code alternative (gray), as well as no symmetry breaking (none). We compared the default branching heuristic (left2right heuristic) with branching heuristics that instantiate variables right to left (right2left heuristic), alternating from both ends inwards to the middle (outside-in heuristic), from the middle alternating outwards to both ends (inside-out heuristic), by order of degree (degree heuristic), and by order of the number of attached constraints

(constr heuristic). Note that all domains are binary so there is again no value for a heuristic like ff that considers domain size.

Branching/SymBreak	none	anti-gray outside-in	gray	lex	anti-gray inside-out	anti-lex inside-out
left2right heuristic	*1,725,076*	1,223,155	468,317	468,390	245,259	291,050
right2left heuristic	*1,725,076*	*322,291*	329,897	299,877	**224,540**	269,628
degree heuristic	2,024,484	603,857	329,897	400,228	500,415	*268,173*
constr heuristic	2,024,484	1,624,765	349,025	313,817	1,097,303	297,616
inside-out heuristic	1,786,741	2,787,164	1,406,831	1,055,918	326,938	268,206
outside-in heuristic	2,053,179	364,469	*284,417*	*284,526*	2,044,042	2,767,059

Table 4. Backtracks required to find the 22 bit sequence of lowest autocorrelation and prove optimality with different branching heuristics and symmetry breaking constraints.

We make some observations about these results. First, the best overall performance is observed when we break symmetry with the anti-Gray code ordering (anti-gray inside-out + right2left heuristic). Second, we observe better performance when the symmetry breaking constraint aligns with the branching heuristic than when it goes against it (e.g. anti-gray outside-in + outside-in heuristic is much better than anti-gray outside-in + inside-out heuristic). Third, the default heuristic (left2right) is again competitive.

7.3 Peaceable armies of queens

The goal of this optimisation problem is to place the largest possible equal-sized armies of white and black queens on a chess board so that no white queen attacks a black queen or vice versa [37]. We used a simple model from an earlier study of symmetry breaking [38]. The model has a matrix of 0/1/2 decision variables, in which $X_{ij} = 2$ iff a black queen goes on square (i, j), $X_{ij} = 1$ iff a white queen goes on square (i, j), and 0 otherwise. Note that our model is now ternary, unlike the binary models considered in the two previous examples. However, the Gray code ordering extends from binary to ternary codes in a straight forward. Similarly, we can extend the decomposition to propagate Gray code ordering constraints on ternary codes.

Table 5 reports results to find the optimal solution and prove optimality for peaceable armies of queens. This model has 15 non-identity symmetries, consisting of any combination of the symmetries of the square and the symmetry that swaps white queens for black queens. We broke all 15 symmetries by posting constraints to ensure that we only find the smallest solution in each symmetry class according to the lexicographical or Gray code orderings (lex, gray), or the largest solution in each symmetry class according to the two orders (anti-lex, anti-gray). We also considered symmetry breaking constraints that take the variables in row-wise order (row), in column-wise order (col), in a snake order along the rows (snake), in a snake order along the columns (col-snake), or in a clockwise spiral (spiral). We again used the default variable ordering that instantiates variables in the lexicographical row-wise order.

We make some observations about these results. First, finding the largest solution in each symmetry class (anti-gray and anti-lex) is always better than finding the smallest (gray and lex). We conjecture that this is because symmetry breaking lines up better with the objective of maximizing the number of queens on the board. Second, symmetry breaking in a "conventional" way (lex, row) is beaten by half of the symmetry

Symmetry breaking	$n = 3$	4	5	6	7	8
none	19	194	2,588	37,434	679,771	19,597,858
lex col-snake	13	98	1,014	8,638	199,964	5,299,787
lex col	23	87	1,042	10,792	198,032	5,197,013
gray col	26	101	1,118	9,763	214,391	5,008,279
gray col-snake	13	100	1,059	8,973	205,453	4,877,014
gray spiral	18	104	913	10,795	169,725	4,690,071
lex spiral	18	93	887	10,694	169,293	4,674,458
gray row	19	73	680	6,975	116,725	3,705,591
gray snake	19	81	685	7,070	117,489	3,683,558
lex snake	19	80	661	7,043	117,590	3,682,438
lex row	19	73	679	6,880	115,999	3,652,269
anti-gray spiral	8	43	466	4,381	108,214	2,402,049
anti-gray snake	8	47	472	4,333	106,317	2,367,290
anti-gray row	8	44	452	4,326	105,837	2,357,024
anti-lex col-snake	18	59	560	4,513	70,950	2,346,875
anti-lex col	18	57	485	4,373	69,484	2,291,512
anti-lex row	9	29	315	3,417	101,530	2,037,336
anti-lex snake	9	34	314	3,366	100,472	2,010,354
anti-lex spiral	9	30	326	3,432	105,717	2,007,586
anti-gray col-snake	19	40	471	4,061	71,079	1,709,744
anti-gray col	19	40	385	4,317	70,632	1,698,492

Table 5. Backtracks required to solve the n by n peaceable armies of queens problem to optimality with the default branching heuristic.

breaking methods. In particular, all 10 methods which find the largest solution up to symmetry in the Gray order (anti-gray) or lexicographical ordering (anti-lex) beat the "conventional" method (lex row). Third, ordering the variables row-wise in the symmetry breaking constraint is best for lex, but for every other ordering (anti-lex, gray, anti-gray) ordering variables row-wise is never best. In particular, anti-lex spiral beats anti-lex row and all other anti-lex methods, gray snake beats gray row and all other gray methods, and anti-gray col beats anti-gray row and all other anti-gray methods. Fourth, a good symmetry breaking method (e.g. anti-gray col) offers up to a 12-fold improvement over not breaking the 15 non-identity symmetries.

To explore the interaction between symmetry breaking and branching heuristics, we report results in Table 6 using different branching heuristics. We used the best symmetry breaking method for the default row-wise branching heuristic (anti-gray col), the worst symmetry breaking method for the default branching heuristic (lex col-snake), a standard method (lex row), the Gray code alternative (gray row), as well as no symmetry breaking (none). We compared the same branching heuristics as with the maximum density still life problem. As domains are now not necessarily binary, we also included the ff heuristic that order variables by their domain size tie-breaking with the row heuristic (ff heuristic). Given the good performance of the spiral and ff heuristics individually, we also tried a novel heuristic that combines them together, branching on variables by order of the domain size and tie-breaking with the spiral-in heuristic (ff-spiral heuristic).

We make some observations about these results. First, the best symmetry breaking constraint with the default branching heuristic (anti-gray col + row heuristic) was either very good or very bad with the other branching heuristics. It offers the best overall performance in this experiment (viz. anti-gray col + spiral-in heuristic), and is the best of all the symmetry breaking methods for 5 other heuristics. However, it also the worst of all the symmetry breaking methods with 4 other heuristics. Second, aligning the branching heuristic with the symmetry breaking constraint at best offers middle of the road performance (e.g. lex row + row heuristic) but can also be counter-productive

Branching/SymBreak	none	lex col-snake	gray row	lex row	anti-gray col
col-snake heuristic	20,209,357	4,270,637	6,372,404	5,836,975	7,363,488
col heuristic	19,597,858	4,384,086	6,338,413	5,775,781	6,811,345
spiral-out heuristic	8,196,693	4,894,264	5,099,899	5,126,074	6,478,506
degree heuristic	19,597,858	3,129,599	4,216,463	4,343,792	6,351,547
snake heuristic	20,209,357	5,261,095	4,258,903	4,221,336	1,946,556
constr heuristic	*7,305,061*	2,757,360	2,650,590	2,645,054	1,789,444
row heuristic	19,597,858	5,299,787	3,705,591	3,652,269	1,698,492
ff heuristic	12,826,856	3,371,419	2,495,788	2,521,351	1,309,529
ff-spiral heuristic	13,400,485	2,447,867	3,147,237	*2,162,657*	1,222,607
spiral-in heuristic	15,577,982	*1,787,653*	*2,387,067*	2,430,499	**1,193,988**

Table 6. Backtracks required to solve the 8 by 8 peaceable armies of queens problem to optimality for different branching heuristics and symmetry breaking constraints.

(e.g. anti-gray col + col heuristic). Third, the spiral-in heuristic offer some of the best performance. This heuristic provided the best overall result, and was always in the top 2 for every symmetry breaking method. Recall that the spiral-in heuristic was one of the worst heuristics on the maximum density still life problem. We conjecture that this is because it delays constraint propagation on the still life problem constraints but not on the constraints in the peaceable armies of queens problem. Fourth, a bad combination of branching heuristic and symmetry breaking constraints is worse than not breaking symmetry if we have a good branching heuristic (e.g. none + constr heuristic beats anti-gray col + col-snake heuristic).

These results support both our hypotheses. Other orderings besides the simple lexicographical ordering can be effective for breaking symmetry, and symmetry breaking should align with both the branching heuristic and the objective function. Unfortunately, as the last example demonstrated, the interaction between problem constraints, symmetry breaking and branching heuristic can be complex and difficult to predict. Overall, the Gray code ordering appears useful. Whilst it is conceptually similar to the lexicographical ordering, it looks at more than one bit at a time. This is reflected in the automaton for the Gray code ordering which has more states than that required for the lexicographical ordering.

8 Conclusions

We have argued that in general breaking symmetry with a different ordering over assignments than the usual lexicographical ordering does not improve the computational complexity of breaking with symmetry. Our argument had two parts. First, we argued that under modest assumptions we cannot reduce the worst case complexity from that of breaking symmetry with a lexicographical ordering. These assumptions are satisfied by the Gray code and Snake-Lex orderings. Second, we proved that for the particular case of row and column symmetries, breaking symmetry with the Gray code or Snake-Lex ordering is intractable (as it was with the lexicographical ordering). We then explored algorithms to break symmetry with other orderings. In particular, we gave a linear time propagator for the Gray code ordering constraint, and proved that enforcing domain consistency on the SNAKELEX constraint, like on the DOUBLELEX constraint, is NP-hard. Finally, we demonstrated that other orderings have promise in practice. We ran experiments on three standard benchmark domains where breaking symmetry with the Gray code ordering was often better than with the Lex-Leader or Snake-Lex methods.

References

1. Puget, J.F.: On the satisfiability of symmetrical constrained satisfaction problems. In Komorowski, J., Ras, Z., eds.: Proceedings of ISMIS'93. LNAI 689, Springer-Verlag (1993) 350–361
2. Crawford, J., Luks, G., Ginsberg, M., Roy, A.: Symmetry breaking predicates for search problems. In: Proceedings of the 5th International Conference on Knowledge Representation and Reasoning, (KR '96). (1996) 148–159
3. Shlyakhter, I.: Generating effective symmetry-breaking predicates for search problems. In: Proceedings of LICS workshop on Theory and Applications of Satisfiability Testing (SAT 2001). (2001)
4. Aloul, F., Sakallah, K., Markov, I.: Efficient symmetry breaking for Boolean satisfiability. In: Proceedings of the 18th International Joint Conference on AI, International Joint Conference on Artificial Intelligence (2003) 271–276
5. Law, Y., Lee, J.: Global constraints for integer and set value precedence. In: Proceedings of 10th International Conference on Principles and Practice of Constraint Programming (CP2004), Springer (2004) 362–376
6. Puget, J.F.: Breaking symmetries in all different problems. In: Proceedings of 19th IJCAI, International Joint Conference on Artificial Intelligence (2005) 272–277
7. Law, Y., Lee, J.: Symmetry Breaking Constraints for Value Symmetries in Constraint Satisfaction. Constraints **11**(2–3) (2006) 221–267
8. Walsh, T.: Symmetry breaking using value precedence. In: Proc. of the 17th European Conference on Artificial Intelligence (ECAI-2006), European Conference on Artificial Intelligence, IOS Press (2006)
9. Walsh, T.: General symmetry breaking constraints. In: 12th International Conference on Principles and Practices of Constraint Programming (CP-2006), Springer-Verlag (2006)
10. Law, Y.C., Lee, J., Walsh, T., Yip, J.: Breaking symmetry of interchangeable variables and values. In: 13th International Conference on Principles and Practices of Constraint Programming (CP-2007), Springer-Verlag (2007)
11. Walsh, T.: Breaking value symmetry. In Fox, D., Gomes, C., eds.: Proceedings of the 23rd National Conference on AI, Association for Advancement of Artificial Intelligence (2008) 1585–1588
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.: Proceedings of the 16th International Conference on the Principles and Practice of Constraint Programming (CP 2010). Volume 6308 of Lecture Notes in Computer Science., Springer (2010) 305–320
13. Rossi, F., van Beek, P., Walsh, T., eds.: Handbook of Constraint Programming. Foundations of Artificial Intelligence. Elsevier (2006)
14. Grayland, A., Miguel, I., Roney-Dougal, C.: Snake lex: An alternative to double lex. In Gent, I.P., ed.: Proceedings of 15th International Conference on Principles and Practice of Constraint Programming. Volume 5732 of Lecture Notes in Computer Science., Springer (2009) 391–399
15. Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Multiset ordering constraints. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003), International Joint Conference on Artificial Intelligence (2003)
16. Benhamou, B., Sais, L.: Theoretical study of symmetries in propositional calculus and applications. In: Proceedings of 11th International Conference on Automated Deduction. Volume 607 of Lecture Notes in Computer Science., Springer (1992) 281–294
17. Benhamou, B., Sais, L.: Tractability through symmetries in propositional calculus. Journal of Automated Reasoning **12**(1) (1994) 89–102

18. Backofen, R., Will, S.: Excluding symmetries in constraint-based search. In Jaffar, J., ed.: Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming. Number 1713 in Lecture Notes in Computer Science, Springer-Verlag (1999) 73–87
19. Gent, I., Smith, B.: Symmetry breaking in constraint programming. In Horn, W., ed.: Proceedings of ECAI-2000, IOS Press (2000) 599–603
20. Fahle, T., Schamberger, S., Sellmann, M.: Symmetry breaking. In Walsh, T., ed.: Proceedings of 7th International Conference on Principles and Practice of Constraint Programming (CP2001), Springer (2001) 93–107
21. Sellmann, M., Hentenryck, P.V.: Structural symmetry breaking. In: Proceedings of 19th Internatinal Joint Conference on AI (IJCAI 2005), International Joint Conference on Artificial Intelligence (2005) 298–303
22. Puget, J.F.: Dynamic lex constraints. In Benhamou, F., ed.: 12th International Conference on the Principles and Practice of Constraint Programming (CP 2006). Volume 4204 of Lecture Notes in Computer Science., Springer (2006) 453–467
23. Katsirelos, G., Walsh, T.: Symmetries of symmetry breaking constraints. In: Proc. of the 19th European Conference on Artificial Intelligence (ECAI-2010), European Conference on Artificial Intelligence, IOS Press (2010)
24. Narodytska, N., Walsh, T.: An adaptive model restarts heuristic. In: Proceedings of 10th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2013). Volume 7874 of Lecture Notes in Computer Science., Springer (2013) 369–377
25. Cohen, D., Jeavons, P., Jefferson, C., Petrie, K., Smith, B.: Symmetry definitions for constraint satisfaction problems. Constraints **11**(2–3) (2006) 115–137
26. Gent, I., Kelsey, T., Linton, S., McDonald, I., Miguel, I., Smith, B.: Conditional symmetry breaking. In van Beek, P., ed.: Proceedings of 11 International Conference on Principles and Practice of Constraint Programming (CP 2005). Volume 3709 of Lecture Notes in Computer Science., Springer (2005) 256–270
27. Flener, P., Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Pearson, J., Walsh, T.: Symmetry in matrix models. In: Proceedings of the CP-01 Workshop on Symmetry in Constraints (SymCon'01). (2001) Held alongside CP 2001. Also APES-30-2001 technical report.
28. Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Global constraints for lexicographic orderings. In: 8th International Conference on Principles and Practices of Constraint Programming (CP-2002), Springer (2002)
29. Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Propagation algorithms for lexicographic ordering constraints. Artificial Intelligence **170**(10) (2006) 803–908
30. Katsirelos, G., Narodytska, N., Walsh, T.: Combining symmetry breaking and global constraints. In Oddi, A., Fages, F., Rossi, F., eds.: Recent Advances in Constraints, 13th Annual ERCIM International Workshop on Constraint Solving and Constraint Logic Programming (CSCLP 2008). Volume 5655 of Lecture Notes in Computer Science., Springer (2009) 84–98
31. Flener, P., Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Matrix modelling. In: Proceedings of the CP-01 Workshop on Modelling and Problem Formulation. (2001) Held alongside CP 2001.
32. Flener, P., Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Matrix modelling: Exploiting common patterns in constraint programming. In: Proceedings of the International Workshop on Reformulating Constraint Satisfaction Problems. (2002) Held alongside CP-2002.
33. Flener, P., Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Pearson, J., Walsh, T.: Breaking row and column symmetry in matrix models. In: 8th International Conference on Principles and Practices of Constraint Programming (CP-2002), Springer (2002)

34. Quimper, C.G., Walsh, T.: Decomposing global grammar constraints. In: 13th International Conference on Principles and Practices of Constraint Programming (CP-2007), Springer-Verlag (2007)
35. Gent, I., Walsh, T.: CSPLib: a benchmark library for constraints. Technical report, Technical report APES-09-1999 (1999) A shorter version appears in the Proceedings of the 5th International Conference on Principles and Practices of Constraint Programming (CP-99).
36. Bosch, R., Trick, M.: Constraint programming and hybrid formulations for three life designs. Annals OR **130**(1-4) (2004) 41–56
37. Bosch, R.: Peaceably coexisting armies of Queens. Optima (Newsletter of the Mathematical Programming Society) **62** (1999) 6–9
38. Smith, B., Petrie, K., Gent, I.: Models and symmetry breaking for peacable armies of queens. In: Proceedings of the First International Conference on Integration of AI and OR Techniques in Constraint Programming (CP-AI-OR), Springer-Verlag (2004) 271–286 LNCS 3011.