Constraint Symmetry for the Soft CSP

Barbara M. Smith[1], Stefano Bistarelli[2,3], and Barry O'Sullivan[1]

[1] Cork Constraint Computation Centre, University College Cork, Ireland
{b.smith|b.osullivan}@4c.ucc.ie
[2] Istituto di Informatica e Telematica, CNR, Pisa, Italy
stefano.bistarelli@iit.cnr.it
[3] Dipartimento di Scienze, Universitá degli Studi "G. d'Annunzio", Pescara, Italy
bista@sci.unich.it

Abstract. We introduce a definition of constraint symmetry for soft CSPs, based on the definition of constraint symmetry for classical CSPs. We show that the constraint symmetry group of a soft CSP is a subgroup of that of an associated classical CSP instance. Where it is smaller, we can successfully exploit the additional symmetries using conditional symmetry breaking. We demonstrate, in a case-study of graph colouring, that eliminating the symmetry of the soft CSP combined with conditional symmetry breaking can lead to huge reductions in the search effort to find an optimal solution to the soft CSP.

1 Introduction

The importance of exploiting symmetry in combinatorial search problems is well known. In this paper we focus on symmetries in soft constraint satisfaction problems (CSPs). The paper follows previous work [5, 7], but introduces a definition of constraint symmetry for soft CSPs based on the definition of constraint symmetry for classical CSPs [8]. We show that the constraint symmetry group of a soft CSP is a subgroup of the constraint symmetry group of the associated classical CSP instance, in which all the soft constraints are treated as hard constraints, and that in practice, the soft symmetry group may be smaller. We can successfully exploit the additional symmetries using conditional symmetry breaking [12]. If the optimal solution to the soft CSP is also a solution to the classical CSP, or close to such a solution, All the symmetries of the classical CSP can be expressed as conditional symmetries, and this will reduce search effort if either the optimal solution is also a solution to the classical CSP, or the proof of optimality requires proving that there is no such solution. Furthermore, the additional symmetries can in some cases be expressed as conditional symmetries that will become active earlier in the search; these are similar to the α-symmetries introduced in [5, 7]. We demonstrate the usefulness of these ideas in a case-study of graph colouring.

2 Semiring-based Soft Constraints

In this section, we outline the semiring framework for soft constraints, which is defined and discussed in more detail in [6, 15]. We first define a classical CSP.

Definition 1 (Constraint Satisfaction Problem). *A CSP instance P is a triple $\langle V, D, C\rangle$ where V is a set of variables, D is a universal domain, specifying the possible values for those variables, and C is a set of constraints. Each constraint $c_i \in C$ is a pair $\langle \sigma_i, \rho_i\rangle$ where σ_i, the constraint* scope*, is a list of variables from V and ρ_i is a $|\sigma_i|$-ary relation over D, called the constraint* relation*. A* solution *to P is a mapping from V into D whose restriction to each constraint scope is in the corresponding constraint relation, i.e. is allowed by the constraint.*

Soft constraints associate a qualitative or quantitative value either to the entire constraint or to each assignment of its variables [6]. In this paper, we use the framework of semiring-based constraints [2]. A semiring structure S is a 5-tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$; as with a classical CSP, we also have a set of variables V with domain D. A is a set of values, representing the levels of preference (or importance or cost) associated with constraints or assignments; $\mathbf{0}$ and $\mathbf{1}$ are elements of A corresponding to complete unacceptability and complete satisfaction respectively. A classical CSP fits into this framework: the assignments allowed by a constraint have value $\mathbf{1}$ and those it forbids have value $\mathbf{0}$. In general, in the semiring framework, a constraint c_i is a *function*: given an assignment to the variables in its scope, it returns a value in A.

The operator $+$ is used to compare the levels of preference in A. A partial order $\leq_S$ is defined over A such that $a \leq_S b$ iff $a + b = b$; in that case, b is preferred to a. Within the preference ordering, $\mathbf{1}$ is the best element of A and $\mathbf{0}$ is the worst. The fact that $\leq_S$ can be a partial order allows for different criteria to be used which may not be comparable. Of course, total orders are also allowed within the framework, if that is more appropriate to the problem.

The operator $\times$ is used to combine preference levels, for instance to compute the preference level of a complete assignment from the preference levels of the individual constraints. When combining a completely violated constraint, with semiring value $\mathbf{0}$, with any other preference level, the result is $\mathbf{0}$; on the other hand, combining any preference $a \in A$ with $\mathbf{1}$ gives a.

An optimal solution to a semiring-based soft CSP $\langle V, D, C, S\rangle$ is a complete assignment whose preference level is at least as great as, or incomparable with, any other complete assignment. In the examples in this paper, $\leq_s$ is a total order and incomparable preferences do not occur; however, the results also apply to soft CSPs in which $\leq_s$ is a partial order.

3 Constraint Symmetry

We adopt the approach to defining symmetries in [8]. A constraint symmetry of a CSP instance is defined as an automorphism of a graph that represents the constraints, the microstructure complement; an automorphism of a graph or hypergraph is a bijective mapping of the vertices that preserves the edges.

For any CSP instance $P = \langle V, D, C\rangle$, the *microstructure complement* of P is a graph with set of vertices $V \times D$. A set of vertices E is a hyperedge of the microstructure complement if it represents an assignment *disallowed* by a constraint, or else consists of a pair of incompatible assignments for the same variable. In other words, a set of vertices $\{\langle v_1, a_1\rangle, \langle v_2, a_2\rangle, \ldots, \langle v_k, a_k\rangle\}$ is a hyperedge if and only if:

– $\{v_1, v_2, \ldots, v_k\}$ is the set of variables in the scope of some constraint, but the constraint disallows the assignment $\{\langle v_1, a_1\rangle, \langle v_2, a_2\rangle, \ldots, \langle v_k, a_k\rangle\}$; *or*
– $k = 2$, $v_1 = v_2$ and $a_1 \neq a_2$.

We can similarly define a graphical representation of the constraints of a semiring-based soft CSP that is analogous to the microstructure complement.

Definition 2 (Microstructure Complement of a Semiring-based CSP Instance). *For any semiring-based CSP instance $P = \langle V, D, C, S\rangle$, the microstructure complement of P is an edge-labelled graph with the set of vertices $V \times D$. For every assignment whose semiring value $\neq \mathbf{1}$, there is a hyperedge joining the vertices of the associated tuple, labelled with the semiring value. There is also an edge, labelled with semiring value $\mathbf{0}$, joining any pair of nodes representing different values for the same variable.*

The difference between this definition of the *microstructure complement* and the definition of the *microstructure* of a soft CSP instance given in [15] is only that we require an edge between each pair of nodes representing two values of the same variable; the edge corresponds to an invalid assignment that cannot be allowed whatever the value given to other assignments by the soft constraints. Because of this difference, it is worthwhile to give a separate definition; moreover, our definition is so close to the definition of the microstructure complement of a classical CSP that is seems sensible to give them the same name.

We can now give a definition of constraint symmetry in soft CSPs that is analogous to the definition of constraint symmetry in classical CSPs in [8].

Definition 3 (Soft Constraint Symmetry). *For any soft CSP instance, a constraint symmetry is an automorphism of its microstructure complement that preserves the edge labels.*

In what follows, when we refer to an automorphism of the microstructure complement of a soft CSP instance, we shall mean one that preserves the edge labels. This is a more restrictive definition of symmetry in soft CSPs than those given in [5, 7]. In those papers, the definitions of symmetry are analogous to solution symmetry as defined in [8]. Although the constraint symmetry group of a classical CSP instance is a subgroup of the solution symmetry group, and can in practice be much smaller, the difficulty of identifying solution symmetries without solving the instance makes constraint symmetry a more practically useful idea; the same applies to soft CSPs.

In constructing the microstructure complement, the semiring operators $\times$ and $+$ are not needed, but only the elements of A and the constraints mapping assignments in $V \times D$ to values in A. In fact, the specific values in A, other than $\mathbf{0}$ and $\mathbf{1}$, do not affect the automorphisms of the microstructure complement: it is the partitions of the hyperedges into sets with the same label that determine the symmetries, not the specific labels. Hence, the symmetries of a soft CSP do not depend on how complete assignments are valued, but only on which sets of assignments are given the same preference values by the constraints.

It is useful to consider the relationship between the constraint symmetry group of a soft CSP instance and an associated classical CSP instance. Suppose that a soft CSP

instance P_s is derived from a classical CSP instance P_h, in which the soft constraints of P_s become hard constraints. P_s might be designed to allow solutions to be produced even if P_h is over-constrained, by associating a semiring value with each nogood in P_h, and thereby allowing solutions to P_s that do not satisfy all the constraints of P_h. In this case, the semiring value represents the 'cost' of violating the constraint. Let G_h, G_s be the microstructure complement of P_h and P_s respectively. Since they have the same vertex set, an automorphism of G_s corresponds to a bijective mapping on the vertices of G_h. Every hyperedge in G_s is a hyperedge in G_h, and any mapping of G_s that maps hyperedges to hyperedges with the same label must translate into a mapping of hyperedges to hyperedges in G_h. Hence, any automorphism of G_s must be an automorphism of G_h and the constraint symmetry group of P_s is a subgroup of that of P_h.

Alternatively, suppose that P_c is a classical CSP instance that has solutions, but we want to differentiate between them by creating a soft CSP instance P_s that expresses preferences. We would keep all the nogoods of P_c, each with a semiring value $\mathbf{0}$, and add further nogoods with a different semiring value, representing solution characteristics that we should prefer to avoid. Every hyperedge in G_c becomes a hyperedge in G_s, with label $\mathbf{0}$, and no other hyperedge in G_s has this label. Any automorphism of G_s must map the hyperedges with label $\mathbf{0}$ to each other, and hence is an automorphism of G_c. In this instance, there is also the classical CSP instance P_h, in which all the soft constraints of P_s must be satisfied. As before, any automorphism of G_s is also an automorphism of G_h. However, the automorphisms of G_c and G_h are incomparable, since the hyperedges of G_h are a superset of those of G_c. Hence, the constraint symmetries of P_s are a subgroup of the constraint symmetries of P_c *and* of the constraint symmetries of P_h, but P_c may have more or less symmetry than P_h .

Given the classical CSP instance P_h derived from a soft CSP instance P_s by setting to $\mathbf{0}$ all the preference levels that are $\neq \mathbf{1}$, we have shown that the symmetries of P_s are a subgroup of the symmetries of P_h. More generally, any abstraction mapping as defined in [3, 4] can be used to derive a new (soft or classical) CSP instance P_a from P_s: the preference levels in A in P_s are mapped into a smaller set A'. In terms of the microstructure complements G_s and G_a respectively, this merges sets of hyperedges with different labels in G_s into a set with a single label in G_a. Hence, by similar arguments to those given earlier, abstraction creates a CSP instance with at least as much symmetry as the original.

4 The Symmetries of the Soft n-Queens Problem

As an example of constraint symmetry in soft CSPs, we consider a soft version of the n-queens problem, where we allow solutions that do not satisfy all the constraints. As usual, we are given an $n \times n$ chessboard and want to place n queens on squares of the board in such a way that no two queens are in the same row, column or diagonal. Suppose that the CSP model of the problem is the usual one where the variables represent the rows of the chessboard, and the values represent the columns. The microstructure complement of the 4-queens instance of this CSP is shown in Figure 1. The nodes of the graph correspond to the squares of the chessboard, and Figure 1 is drawn so that each

node is in the position of its corresponding square. The automorphisms of the graph, and so the constraint symmetries of this problem, are the eight symmetries of the chessboard: reflections in the horizontal and vertical axes; rotations through 90°, 180° and 270°; reflections in the two main diagonals; and the identity symmetry. We can denote these by $x, y, r90, r180, r270, d1, d2, i$ respectively.

Suppose that in the soft CSP, we assign to the constraint $v_i \neq v_j$ the value $|i - j|/n$, so that $|i - j|/n$ becomes the label on the edge joining the vertices $\langle v_i, k\rangle$ and $\langle v_j, k\rangle$, for $k = 1, 2, ..., n$. Similarly, when $|v_i - v_j| = |i - j|$, we assign the value $|i - j|/n$. If the constraint is satisfied, the value is **1**. Hence the set of values A, when n = 4, is $\{\mathbf{0}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \mathbf{1}\}$. The microstructure complement of the soft 4-queens problem is shown in Figure 1 (right). Rather than showing the label of each edge, edges with different labels are drawn differently: the solid lines are the edges labelled **0** and the various dashed lines are edges with labels $\frac{1}{4}$, $\frac{1}{2}$, or $\frac{3}{4}$. Note that the rows of the board are now distinguished from the columns, because the only edges labelled **0** join nodes corresponding to squares in the same row.

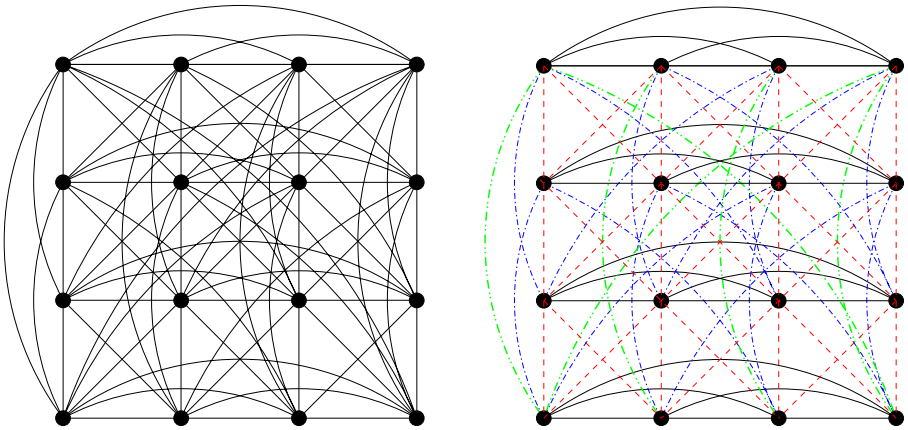

Fig. 1. The microstructure complement of the classical 4-queens CSP (left) and the microstructure complement of a soft CSP based on the same problem (right).

Not all of the eight chessboard symmetries are symmetries of the soft CSP. Suppose that a solution to the soft CSP has two queens in the first column. Rotating this solution through 90° gives a layout with two queens in the first row; since this requires assigning two values to the same variable, it is not a valid assignment. In terms of the microstructure complement, the rotation through 90° maps edges with labels $\frac{1}{4}$, $\frac{1}{2}$ and $\frac{3}{4}$ to edges with label **0**, and so is not an automorphism. However, the chessboard symmetries that map rows into rows translate into automorphisms of the soft microstructure complement, and hence the symmetry group of the soft CSP is the subgroup consisting of $x, y, r180$ and i.

This section demonstrates how the symmetry group of a soft CSP relates to its microstructure complement; however, as in the classical CSP case, constraint symmetries can often be identified by inspection.

5 Symmetries Conditional on Solution Value

In solving a CSP, whether a classical CSP or a soft CSP, it is important to eliminate as much of the symmetry as possible; any remaining symmetry in the problem may cause wasted effort during search, in exploring subtrees that are symmetrically equivalent to subtrees already explored. As in classical CSPs, there is a variety of methods available to eliminate symmetry; we can for instance add further constraints (with semiring value $\mathbf{0}$) to the CSP that ideally can be satisfied by exactly one solution in each symmetry equivalence class [9, 14]; or we can use a dynamic symmetry breaking method, as explored in [5, 7].

The example of soft n-queens shows that a soft CSP instance P_s may have fewer symmetries than its classical counterpart P_h in which all the soft constraints become hard constraints. In that case, we cannot, of course, eliminate the symmetries of P_h in solving P_s, because this would risk losing solutions (including possibly the optimal solution). However, in solving the soft CSP we may need to search for solutions that are also solutions to P_h, either because the optimal solution to P_s is a solution of P_h, or in proving optimality. In that case, eliminating the symmetries of P_s, but not those of P_h, may lead to wasted search effort.

As an example, we consider a graph colouring problem. Colouring the nodes of a graph so that no adjacent nodes have the same colour can easily be represented as a classical CSP, with variables $v_1, v_2, ..., v_n$ corresponding to the nodes in the graph. The domain of each variable is $\{0, 1, ..., k-1\}$ where k is the number of colours allowed, For each edge (i, j) in the graph there is a constraint in the CSP that $v_i \neq v_j$. The symmetries of the CSP are that the colours are interchangeable; hence, the size of the symmetry group is $k!$, ignoring any symmetry in the graph itself. If k colours are not sufficient to colour the graph, we might want to find the 'best' colouring with the colours available, by attaching a cost to any assignment that does not satisfy the constraints. To illustrate the point, suppose that the value of the assignment $\{v_i = l, v_j = l\}$, where nodes i and j are adjacent, is $\max(2, l)$. Under this valuation, only the colours 0, 1, 2 are interchangeable in the soft CSP.

Suppose that the value of a complete assignment is the sum of the values of individual assignments; an optimal solution minimizes the sum. A solution with value 2 must have just one edge whose vertices have the same colour; if the search finds a solution with value 2, it will then look for a better solution, with *no* edge whose vertices have the same colour. The search will only accept solutions with value 0, corresponding to semiring value $\mathbf{1}$, i.e. it will look for a solution to the classical CSP, P_h. At this point, the symmetries of the classical CSP become valid, and it would be legitimate to eliminate them.

What we want to do ideally is to use the symmetries of P_h where possible, and the symmetries of the soft CSP, P_s, otherwise. The symmetries of P_h that are not also symmetries of P_s can be viewed as *conditional* symmetries [12]. Conditional symmetries

exist only within a subproblem; in this case, within the subproblem where the semiring value is $\mathbf{1}$.

We show how this works out in practice in Section 7.

6 Further Conditional Symmetries

If a soft CSP instance P_s has less symmetry than the classical CSP instance P_h in which all the soft constraints must be satisfied, then in theory it would be useful to treat the additional symmetries as conditional symmetries, which become valid if the search explores a subproblem where solutions must have semiring value $\mathbf{1}$, as described in the last section. However, this will only be useful in practice if the search looks for such solutions; if the optimal solution to the soft CSP has many violated constraints, the conditional symmetries will never become relevant.

However, there are sometimes additional symmetries that can be active even when the optimal solution is not close to a solution to P_h; the graph colouring example illustrates this. Consider the edges of the graph where the constraints of P_h are violated, i.e. the edges whose nodes have the same colour. If the largest colour assigned to the nodes of any of these edges is i, with $i \geq 2$, then the colours $i+1, ..., k-1$ are interchangeable. Hence, there are conditional symmetries whose condition is not the overall value of the solutions within the subproblem, but a specific property of those solutions, namely the largest colour used in a violated constraint. Moreover, there is a hierarchy of such conditional symmetries. Some of them will become active at an early stage of the search; as the search progresses and better solutions are found, more colours can be treated as interchangeable. Again, experiments showing the value of this are presented in Section 7.

Although the graph colouring example is an artificial one constructed for illustration purposes, conditional soft symmetries that do not depend on the value of the solution can occur in more realistic problems. The following example is taken from a competition held in connection with the PATAT conference in 2002-3[4]. It is a simplified version of a university timetabling problem, consisting of classes to be scheduled over 5 days with 9 hours in each day, rooms where classes can take place, students who attend the classes, and room features required by classes. In a feasible timetable, every class is assigned a timeslot and a room so that the following hard constraints are satisfied:

- no student attends more than one class at the same time;
- the room assigned to a class is large enough and has the required features;
- only one class is in each room in any timeslot.

The soft constraints are that:

- a student should not have a class in the last slot of the day;
- a student should not have more than two classes consecutively (within a day);
- a student should not have a single class on a day.

[4] See http://www.idsia.ch/Files/ttcomp2002/.

In the problem of finding a feasible timetable, the 45 timeslots are interchangeable. In adding the soft constraints, some symmetry is lost; however, the days as a whole are still interchangeable. The classical CSP in which all the soft constraints become hard constraints also has more symmetry than the soft CSP; given any timetable that satisfies all the soft constraints, and therefore has no class assigned to the last timeslot of any day, we can reverse the timetable for the first 8 timeslots in any individual day. In any subproblem in which the "no class at the end of the day" soft constraint is satisfied in day i, the reversal symmetry applies to the timetable for that day. Hence this is a conditional symmetry, whose condition is that there is no class assigned to the last timeslot of day i. (Unfortunately, we cannot solve the competition problems and demonstrate the conditional symmetries in this case: the competition was designed to test metaheuristic algorithms and the instances are too large to be amenable to complete search methods.)

7 Experimental Evaluation

Experiments with symmetry breaking, including conditional symmetry breaking, are presented in this section. The experiments were carried out using ILOG Solver 6.0, on a Pentium 1.7GHz laptop running Windows 2000. We consider the soft graph colouring problem described earlier, where the aim is to find the best colouring with k colours.

As described earlier, the symmetries of the soft CSP are that the colours 0, 1, 2 are interchangeable. In addition, the soft CSP has conditional symmetries; some are conditional on the value of the solution: if the search is looking for a solution with value 0 (semiring value **1**), then all the colours are interchangeable. Finally, if the largest colour assigned to any edge whose vertices have the same colour is i, the colours $i + 1, ..., k - 1$ are interchangeable.

In the CSP, the colours are represented by the domain values of the variables. Symmetries due to interchangeable values can easily be eliminated using SBDS (Symmetry Breaking During Search) [13]; this is a dynamic symmetry breaking method that adds constraints on backtracking to a choice point, to ensure that choices symmetrically equivalent to the choice already explored will not be explored in future. SBDS requires a function for each symmetry specifying the symmetric equivalent of the assignment of a value to a variable. In the case of interchangeable values, SBDS functions are needed only for transpositions of pairs of values, rather than one function for each element of the symmetry group [1]. For instance, the equivalent of $v_i = j$ under the symmetry σ_{jl} that transposes the values j and l is $v_i = l$. On backtracking to a choice point between $v_i = j$ and $v_i \neq j$, SBDS uses this function both to check whether or not the symmetry remains unbroken on the path through the search tree to this point, and if so, to post the symmetric equivalent of the constraint $v_i \neq j$, that is $v_i \neq l$.

The symmetry functions have access to the variables c, representing the value of the solution, and *maxCol*, representing the largest colour used in any violated constraint. We describe the conditional symmetry function for the symmetries that depend on the value of c; other conditional symmetries are analogous. For the symmetry σ_{jl}, where at least one of $j, l > 2$, the conditional symmetry function returns the constraint $c = 0 \wedge v_i = l$ as the symmetric equivalent of $v_i = j$. If this symmetry is still unbroken, SBDS adds the negation of this constraint, i.e. $c \neq 0 \vee v_i \neq l$.

For our experiments we randomly generated connected graphs with 20 nodes, with the probability of an edge between a pair of nodes set as 0.7. For these parameter values, the chromatic number, γ, i.e. the minimum number of colours required to give a perfect colouring, is between 7 and 9. ILOG Solver's default optimization search was used, with the value of the solution defined as the objective. As each successive solution is found, future solutions are constrained to have a smaller objective value. When it can be proved that no better solution exists, the incumbent solution has been proved optimal. The variable ordering used is smallest domain, breaking ties by the smallest future degree; the value ordering is to use the value that had been least used so far. The value ordering was introduced for the soft case; if the smallest available value is used instead, as is usual, the first solution found has every node coloured 0, and it takes a long time to find a good solution.

We first solved 20 instances with the number of colours allowed equal to γ, so that it is possible to colour the graph exactly. For these instances, we compared no symmetry breaking at all with breaking as much symmetry as possible, i.e. we used SBDS functions for the three unconditional symmetries of the soft CSP and for both types of conditional symmetry. Symmetry breaking makes little or no difference for most instances. Of the 20 instances, 19 are solved with almost the same number of backtracks, with and without symmetry breaking. Since there is some overhead in adding symmetry-breaking constraints on backtracking, solving these instances takes slightly longer with symmetry breaking than without, though very little time in either case (less than 1.5 sec. for any instance). However, the 20th instance takes 1,530 backtracks to solve with symmetry breaking, and nearly 1.5 million without. The time taken to solve this instance without symmetry breaking (192 sec.) vastly outweighs the additional overhead of using symmetry breaking on all the other instances, which amounts to less than 1 second in total. A solution with value 2 can easily be found, both with and without symmetry breaking; from that point on, both types of conditional symmetry become unconditional, and all colours can be treated as interchangeable. If just the symmetries of the soft CSP are eliminated (that the colours 0, 1, 2 are interchangeable), solving this instance takes around 750,000 backtracks and 100 seconds.

Symmetry breaking, including conditional symmetry breaking, is clearly beneficial for this type of instance, even though it makes no difference for most individual instances. It is in line with previous experience with classical CSPs that symmetry breaking often makes little difference in finding a single solution to a CSP instance, if there is one. However, these experiments show that this is not always true, and if the symmetry group is large, it can be well worthwhile to eliminate the symmetry for the sake of the occasional instance where a solution is hard to find.

We then generated 20 instances for which the optimal value is 2, when the number of colours allowed is $\gamma - 1$. (Recall that, given the values of the soft constraints, 2 is the smallest possible non-zero value for a solution.) We solved these using no symmetry breaking and three different symmetry-breaking strategies. In the first, only the symmetries of the soft CSP are broken. In the 2nd strategy, we used the remaining symmetries of the classical CSP, conditional on the value of the solution being 0. Since the optimal value is 2, the proof of optimality requires proving that there is no solution with value 0, and the conditional symmetries come into play at the proof stage. Finally, we made

the symmetries that transpose colours j, l where $2 < j < l$, conditional on *maxCol* $< j$, where *maxCol* is the largest colour used in a violated constraint, rather than on $c = 0$. This allows the symmetry to be eliminated at an earlier stage in the search.

The results for the 20 instances are shown in Table 1, in ascending order of total search effort for the best symmetry breaking strategy. It is useful to separate the

Table 1. Comparison of symmetry-breaking using SBDS for soft graph colouring instances whose optimal value is 2. The symmetries broken are either the soft symmetries (0,1,2 are interchangeable), or that all values are interchangeable when the solution value = 0, or that colours $> l$ are interchangeable when the largest colour involved in a violated constraint is $\leq l$. 'F' is the number of backtracks to find the optimal solution; 'P' is the total number of backtracks; the time is in seconds on a 1.7GHz Pentium PC.

No. of	No symmetry breaking			Soft symmetries		Cond. symmetries: $c = 0$		Cond. symmetries: *maxCol*$\leq l$		
colours	F	P	time	P	time	P	time	F	P	time
8	1,269	1,815,682	412.32	308,242	66.11	1,338	0.26	1,269	1,338	0.26
8	1,232	403,169	85.45	143,009	26.11	1,380	0.22	1,232	1,380	0.23
7	2,283	455,540	11.45	10,668	2.42	2,324	0.34	2,283	2,324	0.36
7	2,913	116,840	23.11	30,032	5.92	3,020	0.43	2,913	3,020	0.43
7	4,187	49,548	13.24	11,892	2.44	4,207	0.61	3,607	3,627	0.54
6	4,090	17,580	2.66	7,190	0.99	4,200	0.56	4,090	4,200	0.55
7	4,740	40,038	9.29	10,782	2.00	4,765	0.68	4,740	4,765	0.61
7	6,353	84,104	17.98	22,592	4.43	6,430	0.75	6,353	6,430	0.69
7	15,068	30,163	9.04	17,731	3.52	15,091	2.25	7,487	7,510	1.06
7	7,636	283,613	46.13	52,757	7.20	7,742	0.98	7,636	7,742	0.89
7	14,406	39,980	9.95	19,124	2.83	14,451	1.51	14,406	14,451	1.51
7	23,388	130,029	27.14	41,973	6.14	23,435	2.36	17,694	17,741	2.11
7	19,184	60,770	11.15	27,314	3.67	19,233	2.04	19,184	19,233	2.02
6	26,061	28,765	3.76	26,761	3.09	26,088	3.12	22,038	22,065	2.54
7	33,294	621,271	108.78	130,015	18.96	33,394	2.87	33,294	33,394	3.04
7	43,751	85,967	17.24	53,495	6.49	43,805	4.52	36,019	36,073	4.70
7	316,784	327,611	42.95	318,755	31.04	316,805	34.81	37,984	38,005	5.56
7	224,022	315,124	46.39	239,668	23.20	224,056	23.23	52,436	52,470	6.49
7	244,249	305,773	44.66	255,205	27.44	244,283	29.16	61,968	62,002	7.67
7	124,966	140,741	21.10	128,285	15.81	124,982	16.95	65,515	65,531	9.01
Ave.	55,993.8	267,615.4	48.19	92,774.5	12.99	56,051.45	6.38	20,107.4	20,165.05	2.51

search into two stages: the columns headed 'F' show the number of backtracks to find a solution with value 2; the columns headed 'P' show the total number of backtracks, including the proof of optimality. We should expect that the symmetry breaking makes most difference at the proof stage; the search then has to show that there is no solution better than the one already found, by an exhaustive search of the remaining possibilities.

For the first three strategies shown in the table, finding the optimal solution takes the same number of backtracks for every instance, and these columns are not repeated.

However, the total search effort, including the proof of optimality, is very different. With no symmetry breaking, the search effort increases with the number of colours, and hence with the number of symmetries. The total search effort is dominated by the proof of optimality: it takes nearly four times as long, on average, to prove that the solution with value 2 is optimal as to find this solution. On the other hand, with symmetries conditional on *maxCol*, the proof of optimality takes little additional effort once the optimal solution is found; furthermore, it takes less than half as much effort to find the optimal solution than with no symmetry breaking. It is also worth noticing that the two instances where the number of colours is 8 are the easiest to solve with this symmetry-breaking strategy, whereas they are amongst the hardest with no symmetry breaking; these instances have the most symmetry. For 11 of the instances, using symmetries conditional on *maxCol* is no better than using symmetries conditional on $c = 0$, even though overall it reduces the average search effort by more than half; the instances where it gives most benefit are those that are most difficult for the other strategies. For these instances, it finds the optimal solution much more quickly, as well as reducing the effort to prove optimality.

We solved a further 20 instances for which the optimal value is 4 when the number of colours is $\gamma - 1$; the optimal solutions have exactly two edges whose vertices are assigned the same colour. These instances are harder to solve than those in Table 1, and we did not attempt to solve them without symmetry breaking. The symmetries conditional on the value of the solution being 0 never become active for these instances, so they cannot affect the number of backtracks, in comparison with breaking just the soft symmetries, and we have not shown the results for this strategy in Table 2. However, the conditional constraints added by SBDS do increase the run-time slightly (by roughly 10%), since there is an overhead involved in adding the constraints. The results for the remaining symmetry-breaking strategies are shown in Table 2; the instances are shown in increasing order of difficulty for the best strategy.

The symmetries that are conditional on the value of *maxCol* are clearly very useful, especially for those instances where the number of colours is 7 rather than 6 (and hence there are more conditional symmetries). For most instances (with one exception) the conditional symmetries have little effect, if any, on the search effort to find the optimal solution, but they speed up the proof of optimality considerably. With a cut-off of 2 million backtracks, the solution could not be proved optimal with just the soft symmetries, in 5 of the 20 instances, although the optimal solution was found in each case. Since these instances are not included in the averages for this strategy, it should be noted that the averages in the table are biased against the conditional symmetries; if 2 million backtracks for each of these instances are included in the calculation of the average number of backtracks, it increases to over 900,000.

The results presented in this section demonstrate that if the symmetry group of the classical CSP is larger than that of the soft CSP, being able to eliminate the resulting conditional symmetries can lead to huge reductions in search effort. Making these symmetries conditional on the value of the solution being the semiring value $\mathbf{1}$ is most useful when the search has to prove that there is no such solution, as in Table 1. If the optimal solution is worse, so that subproblems in which the value of the solution is constrained to be $\mathbf{1}$ never arise, these conditional symmetries never become active,

Table 2. Comparison of symmetry-breaking using SBDS for soft graph colouring instances whose optimal value is 4. The symmetries broken are either the soft symmetries (0,1,2 are interchangeable), or that colours $> l$ are interchangeable when the largest colour involved in a violated constraint is $\leq l$. 'F' is the number of backtracks to find the optimal solution; 'P' is the total number of backtracks; the time is in seconds on a 1.7GHz Pentium PC. Averages exclude instances that reach the cutoff of 2 million backtracks.

No. of colours	Soft symmetries			Cond. symmetries: *maxCol* $\leq l$		
	F	P	time	F	P	time
6	14,962	110,231	18.97	14,962	61,200	10.29
6	28,263	133,530	15.02	28,262	80,439	13.10
6	24,937	139,168	18.19	24,937	81,703	13.04
6	4,517	248,960	31.74	4,517	125,577	22.19
6	5,000	278,943	33.01	5,000	141,921	23.59
6	34,414	334,160	48.32	34,414	182,874	27.02
6	49,234	345,038	54.08	49,234	195,950	30.91
6	6,503	390,822	59.55	6,503	196,757	30.94
7	17,560	1,146,070	155.14	17,531	202,487	36.57
6	65,008	422,944	48.18	64,927	244,408	36.77
6	171,462	326,490	39.97	171,011	248,000	42.78
6	8,515	577,918	85.28	8,515	292,529	45.45
7	3,301	1,794,580	214.93	3,301	298,872	52.93
6	168,687	599,754	89.90	168,687	381,969	56.33
7	61,563	$>$ 2m.		61,563	463,088	80.81
7	423,299	1,968,826	323.76	212,149	463,780	74.35
7	201,208	$>$ 2m.		201,208	489,981	80.64
7	5,073	$>$ 2m.		5,073	1,062,627	169.67
7	58,619	$>$ 2m.		58,619	1,128,539	170.87
7	38,077	$>$ 2m.		38,077	1,326,228	212.23
Ave.	69,510.1	587,828.9	82.40	58,924.5	383,446.45	61.52

and offer no benefit. However, some of the additional symmetries may still be relevant during the search, conditional on some other aspect of the solution. Table 2 shows that eliminating the symmetries as early in the search as possible can give large reductions in search effort, both in finding the optimal solution and in proving optimality.

8 Discussion and Conclusions

We have enriched the notions of *symmetry for satisfiability* and *symmetry for all solutions* for soft CSPs [5, 7] with the notion of *constraint symmetry*, defined analogously to constraint symmetry for classical CSPs [8]. It was previously noted that symmetries in soft CSPs are rarer than in classical CSPs [5, 7]; we have shown that the constraint symmetry group of a soft CSP instance P_s is a subgroup of that of the corresponding classical CSP instance P_h in which the soft constraints become hard constraints, and in practice may be smaller. The additional symmetries can be successfully exploited by

using *conditional* symmetry breaking. All the additional symmetries can be eliminated in any subproblem in which the solution to P_s is constrained to be a solution to P_h, i.e. to have semiring value $\mathbf{1}$; this happens either if the optimal solution to P_s is also a solution to P_h, or if the proof of optimality requires proving that there is no such solution.

The graph colouring example shows that the additional symmetries may become applicable at an earlier stage in the search, conditional on some property of the solution, not necessarily its value. If the conditional symmetries depend on the value of the solution, they can be seen as similar to α-symmetry [5, 7]; α-symmetries preserve solutions whose semiring value is better than a threshold α, but can also map solutions to solutions with a different semiring value, if both values are worse than α. However, the graph colouring example shows that the symmetries of P_h that are not symmetries of P_s can be conditional on other properties of the solution (in that case, the largest colour involved in a violated constraint). Hence, the conditional symmetries that arise in soft CSPs are more general than α-symmetries.

Soft CSPs provide a context in which conditional symmetries can come into their own; whenever a soft CSP instance P_s has fewer symmetries than its classical CSP counterpart P_h, the additional symmetries will hold in the subproblem in which the solution value is $\mathbf{1}$, if not sooner. In general, identifying conditional symmetries in a classical CSP instance is difficult; there is no practicable way to identify the symmetries of all the subproblems that could arise during search, and conditional symmetry breaking has hitherto relied on insight to identify useful conditional symmetries, as in the examples given in [12]. However, the conditional symmetries that arise in soft CSPs are well-defined: the constraint symmetry groups of P_h and P_s can be found, in principle automatically, and the conditional symmetries, if any, are those that are in the first group but not in the second.

Having identified the conditional symmetries, we need an efficient way to deal with them. It is pointed out in [12] that, in general, SBDS is not a satisfactory way to deal with conditional symmetry because it requires a different symmetry function for every *possible* conditional symmetry, and there might be very many of them. Again, this difficulty is mitigated in soft CSPs, because we need to deal only with the constraint symmetry group of P_h. Furthermore, SBDS adds conditional constraints on backtracking to a choice point; this could slow down the search significantly if there are many SBDS functions and no subproblem in which the conditions are satisfied ever arises. In the graph colouring example, the overhead is manageable even when the conditional constraints never becomes active, because the number of SBDS functions is small; in other problems, this might become a difficulty. One possibility is to add the SBDS constraints only when the condition holds; in general, this is not a valid approach, as pointed out in [12], because on backtracking, the condition might become false, and this could result in symmetrically equivalent subtrees being explored. In the case of symmetries that are conditional on the value of the solution, however, once the condition is true, it will never become false on backtracking; when a solution with a given value has been found, the search never again considers worse solutions. Hence, this might be a valid approach for these symmetries.

A remaining drawback of SBDS is that in most cases a symmetry function is required for every element of the symmetry group, for complete symmetry breaking, and this is impracticable if the symmetry group is large. The same difficulty applies to adding constraints to the CSP. SBDD (Symmetry breaking by Dominance Detection) [10, 11], as explored in [7], may then be a better option.

Our empirical results on graph colouring problems demonstrate that eliminating both the symmetry of the soft CSP and the conditional symmetry can lead to huge reductions in the search effort to find an optimal solution. Where the soft CSP has less symmetry than its classical counterpart, the additional symmetries will apply whenever the solutions is constrained to have semiring value **1**; however, it is important to consider whether the additional symmetries apply earlier. The more symmetry is eliminated, and the earlier in search, the better the results.

Acknowledgments

This material is based in part on works supported by the Science Foundation Ireland under Grant Nos. 00/PI.1/C075 and 05/IN/I886.

References

1. Rolf Backofen and Sebastian Will. Excluding Symmetries in Constraint-Based Search. In Joxan Jaffar, editor, *Principles and Practice of Constraint Programming - CP'99*, LNCS 1713, pages 73–87. Springer, 1999.
2. Stefano Bistarelli. *Semirings for Soft Costraint Solving and Programming*. LNCS 2962. Springer, 2004.
3. Stefano Bistarelli, Philippe Codognet, Yan Georget, and Francesca Rossi. Abstracting Soft Constraints. In *New Trends in Constraints: Proc. Joint ERCIM/Compulog Net Workshop, 1999. Selected Papers*, volume 1865 of *LNAI*, pages 108–133. Springer, 2000.
4. Stefano Bistarelli, Philippe Codognet, and Francesca Rossi. Abstracting Soft Constraints: Framework, Properties, Examples. *Artificial Intelligence*, 139(2):175–211, 2002.
5. Stefano Bistarelli, Jerome Kelleher, and Barry O'Sullivan. Symmetry Breaking in Soft CSPs. In *Proceedings of AI-2003, the Twenty-third SGAI International Conference on Knowledge-Based Systems and Applied Artificial Intelligence*, BCS Conference Series "Research and Development in Intelligent Systems XX", pages 199–212. Springer, 2004.
6. Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint solving and optimization. *Journal of ACM*, 44(2):201–236, 1997.
7. Stefano Bistarelli and Barry O'Sullivan. Combining branch & bound and SBDD to solve soft CSPs. In *Proceedings of SymCon Workshop*, 2004.
8. David Cohen, Peter Jeavons, Chris Jefferson, Karen E. Petrie, and Barbara M. Smith. Symmetry Definitions for Constraint Programming. *Constraints*, 11:115–137, 2006.
9. James Crawford, Matthew Ginsberg, Eugene Luks, and Amitabha Roy. Symmetry-Breaking Predicates for Search Problems. In *Proceedings KR'96*, pages 149–159, 1996.
10. Torsten Fahle, Stefan Schamberger, and Meinolf Sellmann. Symmetry Breaking. In Toby Walsh, editor, *Principles and Practice of Constraint Programming - CP 2001*, LNCS 2239, pages 225–239. Springer, 2001.
11. Filippo Focacci and Michaela Milano. Global Cut Framework for Removing Symmetries. In Toby Walsh, editor, *Principles and Practice of Constraint Programming - CP 2001*, LNCS 2239, pages 77–92. Springer, 2001.

12. Ian P. Gent, Tom Kelsey, Steve A. Linton, Iain McDonald, Ian Miguel, and Barbara M. Smith. Conditional Symmetry Breaking. In P. van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005*, LNCS 3709, pages 256–270. Springer, 2005.
13. Ian P. Gent and Barbara M. Smith. Symmetry Breaking During Search in Constraint Programming. In W. Horn, editor, *Proceedings ECAI'2000, the European Conference on Artificial Intelligence*, pages 599–603, 2000.
14. Pedro Meseguer, Francesca Rossi, and Thomas Schiex. Soft constraints. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 9, pages 281–328. Elsevier, 2006.
15. Jean-François Puget. On the Satisfiability of Symmetrical Constrained Satisfaction Problems. In J. Komorowski and Z. W. Ras, editors, *Methodologies for Intelligent Systems (Proceedings of ISMIS'93)*, LNAI 689, pages 350–361. Springer-Verlag, 1993.