Symmetry Breaking in Soft CSPs [⋆]

Stefano Bistarelli[1,2], Jerome Kelleher[3], and Barry O'Sullivan[3]

[1] Istituto di Informatica e Telematica, CNR, Pisa, Italy
`Stefano.Bistarelli@iit.cnr.it`
[2] Dipartimento di Scienze,
Universitá degli Studi "G. D'annunzio" di Chieti-Pescara, Italy
`bista@sci.unich.it`

[3] Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
`{j.kelleher,b.osullivan}@4c.ucc.ie`

Abstract. Exploiting symmetry in constraint satisfaction problems has become a very popular topic of research in recent times. The existence of symmetry in a problem has the effect of artificially increasing the size of the search space that is explored by search algorithms. Another significant topic of research has been approaches to reasoning about preferences. As constraint processing applications are becoming more widespread in areas such as electronic commerce, configuration, etc., it is becoming increasingly important that we can reason about preferences as efficiently as possible. We present an approach to dealing with symmetry in the semiring framework for soft constraints. We demonstrate that breaking symmetries in soft constraint satisfaction problems improves the efficiency of search. The paper contributes to the state-of-the-art in symmetry breaking, as well as in reasoning about preferences.

1 Introduction

Exploiting symmetry in constraint satisfaction problems has become a very popular topic of research in recent times [1, 2, 16, 20, 22, 25]. The existence of symmetry in a problem has the effect of artificially increasing the size of the search space that is explored by search algorithms. Therefore, a typical approach is to break the symmetries in the problem so that only unique solutions are returned (i.e. that only one exemplar of each symmetric equivalence class of solutions is returned). The complete set of solutions can be trivially computed using the symmetry in the problem. The significant advantage is that not only do we return fewer solutions, but we also reduce the search effort required to find these solutions by eliminating symmetric branches of the search tree.

Another significant topic of research in the constraint processing community is the ability to reason about preferences [4, 21]. It has been shown how preferences can be modeled as constraints [6, 14]. As constraint processing applications are becoming

[⋆] This work has received support from Enterprise Ireland under their Basic Research Grant Scheme (Grant Number SC/02/289) and their International Collaboration Programme (Grant Number IC/2003/88).

more widespread in areas such as electronic commerce, configuration, etc., it is becoming increasingly important that we can reason about preferences in as efficient a manner as possible. For example, a typical problem in e-commerce systems requires that we satisfy a set of user-specified preference constraints to a maximal degree. The user would typically wish to see a set of alternative solutions to their preferences, but would like to have diversity amongst the set presented to him. This is a well studied issue in the case-based reasoning community [27], but is less well studied in the constraint processing community. One obvious avenue to be explored here are notions of symmetry in preferences. Diversity in this case might be interpreted as the presentation of a set of solutions which are members of different symmetric equivalence classes. This is a potential application of the work presented in this paper.

The work reported is focused on symmetry breaking in soft constraint satisfaction problems. We present an approach to dealing with symmetry in the semiring framework for soft constraints [6, 8]. We first give definitions of symmetry, extending the work of Benhamou [2], and then relax them using the notion of degradation. The theoretical results are enforced by some empirical tests, showing that breaking symmetries in soft constraint satisfaction problems improves the efficiency of search.

The remainder of the paper is structured as follows. Section 2 presents a review of soft constraints and and overview of the state-of-the-art in the area of symmetry-breaking. We present the theoretical aspects of our approach to symmetry breaking in soft CSPs in Section 3 and give some examples in Section 4. Some empirical results are presented in Section 5. Some concluding remarks are made in Section 6.

2 Background

Before presenting an approach to dealing with symmetry in soft CSPs, we will first present a review of the background to this work. In Section 2.1 a brief state-of-the-art review of symmetry breaking in CSPs will be presented. In Section 2.2 the semiring-based approach to soft CSPs will be recapitulated for the convenience of the reader.

2.1 Symmetry Breaking

There is significant interest within the constraint programming community in exploiting symmetry when solving constraint satisfaction problems. As a consequence, a growing number of techniques are being reported in the literature. Benhamou [2] presented an early analysis of symmetry-breaking and placed it in the context of Freuder's work on interchangeability, a special case of symmetry [18].

A common approach to symmetry breaking involves carefully modeling the problem so that symmetries have been removed. For example, Crawford *et al.* [13] have demonstrated how constraints can be added to the model in order to break symmetries. Puget [24] has presented a formal approach to symmetry breaking that involves the addition of ordering constraints to break symmetries. Flener *et al.* [16] adopt a similar approach by adding ordering constraints to break symmetries in matrix models. Flener *et al.* [16] also remind us that symmetry detection is graph-isomorphism complete in the general case, pointing to the work of Crawford [12].

Brown *et al.* [11] have presented a modified backtracking algorithm that breaks symmetry by pruning branches of the search tree dynamically. This is done by ensuring that only one solution from each symmetric equivalence class is computed. Similarly, a general method for eliminating symmetries, known as symmetry breaking during search (SBDS), has been proposed by Gent and Smith [20]. The SBDS approach is based on earlier work by Backofen and Will [1]. Both of these methods can be regarded as examples of a class of approaches to handling symmetries that involve the addition of constraints during search to avoid symmetrical states in the search space. An implementation of SDBS based on the GAP computational abstract algebra system has been presented by Gent *et al.* [19].

Meseguer and Torras [23] have reported the use of search ordering heuristics to avoid symmetries during search. However, the method is less general that SBDS [20].

The notion of partial symmetry breaking has been explored by McDonald and Smith [22]. They show that there is a break-even point to be considered when breaking symmetries during search; there is a point where the benefit in reducing search from adding more symmetries it out-weighed by the extra overhead incurred. By breaking a subset of the possible symmetries in a problem, rather than breaking all of them, significant savings in runtime can be accomplished.

Finally, symmetry breaking based on no-good recording methods have been presented by Fahle *et al.* [15] and Focacci and Milano [17]. The approach presented by the former is known as symmetry-breaking via dominance detection (SBDD) and been shown to compare well with SBDS. The latter approach is known as the global cut framework. Puget has presented an improvement on these approaches by using an auxiliary CSP for performing dominance checks based on no-good recording [25].

It should be noted that all of the approaches to dealing with symmetry presented above are defined for crisp constraints. In this paper we present, for the first time, a theoretical framework for exploiting symmetries in soft CSPs, and demonstrate the utility of the approach empirically using an SBDS-like approach. SBDS is chosen since it is a very flexible approach to breaking symmetries and is readily applicable to symmetry breaking in soft CSPs.

2.2 Soft CSPs

Several formalizations of the concept of *soft constraints* are currently available. In the following, we refer to the one based on c-semirings [3, 5, 6, 8], which can be shown to generalize and express many of the others [4]. A soft constraint may be seen as a constraint where each instantiations of its variables has an associated value from a partially ordered set which can be interpreted as a set of preference values. Combining constraints will then have to take into account such additional values, and thus the formalism has also to provide suitable operations for combination ($\times$) and comparison ($+$) of tuples of values and constraints. This is why this formalization is based on the concept of c-semiring, which is just a set plus two operations.

Semirings. A semiring is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that: 1. A is a set and $\mathbf{0}, \mathbf{1} \in A$; 2. $+$ is commutative, associative and $\mathbf{0}$ is its unit element; 3. $\times$ is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element. A c-semiring is a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that: $+$ is idempotent, $\mathbf{1}$ is its absorbing element and $\times$ is commutative. Let us consider the relation $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$. Then it is possible to prove that (see [6]): 1. $\leq_S$ is a partial order; 2. $+$ and $\times$ are monotone on $\leq_S$; 3. $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum; 4. $\langle A, \leq_S \rangle$ is a complete lattice and, for all $a, b \in A$, $a + b = lub(a, b)$ (where lub is the *least upper bound*). Moreover, if $\times$ is idempotent, then: $+$ distributes over $\times$; $\langle A, \leq_S \rangle$ is a complete distributive lattice and $\times$ its glb (*greatest lower bound*). Informally, the relation $\leq_S$ gives us a way to compare semiring values and constraints. In fact, when we have $a \leq_S b$, we will say that *b is better than a*. In the following, when the semiring will be clear from the context, $a \leq_S b$ will be often indicated by $a \leq b$.

Constraint Problems. Given a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ and an ordered set of variables V over a finite domain D, a *constraint* is a function which, given an assignment $\eta : V \rightarrow D$ of the variables, returns a value of the semiring. By using this notation we define $\mathcal{C} = \eta \rightarrow A$ as the set of all possible constraints that can be built starting from S, D and V.

Note that in this *functional* formulation, each constraint is a function (as defined in [8]) and not a pair (as defined in [5, 6]). Such a function involves all the variables in V, but it depends on the assignment of only a finite subset of them. So, for instance, a binary constraint $c_{x,y}$ over variables x and y, is a function $c_{x,y} : V \rightarrow D \rightarrow A$, but it depends only on the assignment of variables $\{x, y\} \subseteq V$. We call this subset the *support* of the constraint. More formally, consider a constraint $c \in \mathcal{C}$. We define its support as $supp(c) = \{v \in V \mid \exists \eta, d_1, d_2 . c\eta[v := d_1] \neq c\eta[v := d_2]\}$, where

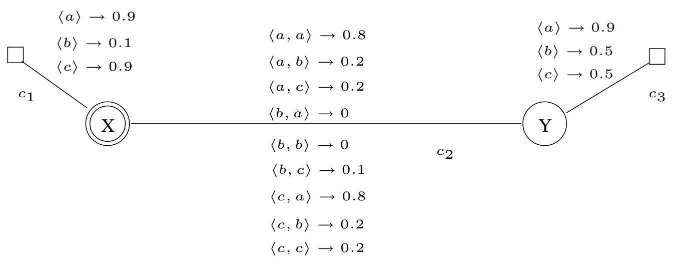
$$\eta[v := d]v' = \begin{cases} d & \text{if } v = v', \\ \eta v' & \text{otherwise.} \end{cases}$$

Note that $c\eta[v := d_1]$ means $c\eta'$ where η' is η modified with the assignment $v := d_1$ (that is the operator $[\;]$ has precedence over application). Note also that $c\eta$ is the application of a constraint function $c : V \rightarrow D \rightarrow A$ to a function $\eta : D \rightarrow A$; what we obtain, is a semiring value $c\eta = a$.

A *soft constraint satisfaction problem* is a pair $\langle C, con \rangle$ where $con \subseteq V$ and C is a set of constraints: con is the set of variables of interest for the constraint set C, which however may concern also variables not in con. Note that a classical CSP is a SCSP where the chosen c-semiring is: $S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true \rangle$. Fuzzy CSPs [10, 26] can instead be modeled in the SCSP framework by choosing the c-semiring $S_{FCSP} = \langle [0, 1], max, min, 0, 1 \rangle$. Many other "soft" CSPs (Probabilistic, weighted, ...) can be modeled by using a suitable semiring structure ($S_{prob} = \langle [0, 1], max, \times, 0, 1 \rangle$, $S_{weight} = \langle \mathcal{R}, min, +, +\infty, 0 \rangle, \ldots$).

Fig. 1 shows the graph representation of a fuzzy CSP. Variables and constraints are represented respectively by nodes and by undirected (unary for c_1 and c_3 and binary for c_2) arcs, and semiring values are written to the right of the corresponding tuples. The variables of interest (that is the set con) are represented with a double circle. Here we assume that the domain D of the variables contains only elements a and b and c.

Combining and projecting soft constraints. Given the set $\mathcal{C}$, the combination function $\otimes : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ is defined as $(c_1 \otimes c_2)\eta = c_1\eta \times_S c_2\eta$. Informally, combining two constraints means building a new constraint whose support involves all the variables of the original ones, and which associates with each tuple of domain values for such variables a semiring element which is obtained by multiplying the elements associ-



$\langle a \rangle \to 0.9$
$\langle b \rangle \to 0.1$
$\langle c \rangle \to 0.9$
c_1
X
$\langle a, a \rangle \to 0.8$
$\langle a, b \rangle \to 0.2$
$\langle a, c \rangle \to 0.2$
$\langle b, a \rangle \to 0$
$\langle b, b \rangle \to 0$
$\langle b, c \rangle \to 0.1$
$\langle c, a \rangle \to 0.8$
$\langle c, b \rangle \to 0.2$
$\langle c, c \rangle \to 0.2$
c_2
$\langle a \rangle \to 0.9$
$\langle b \rangle \to 0.5$
$\langle c \rangle \to 0.5$
c_3
Y

Fig. 1. A fuzzy CSP.

ated by the original constraints to the appropriate sub-tuples. It is easy to verify that $supp(c_1 \otimes c_2) \subseteq supp(c_1) \cup supp(c_2)$.

Given a constraint $c \in \mathcal{C}$ and a variable $v \in V$, the *projection* of c over $V - \{v\}$, written $c \Downarrow_{(V-\{v\})}$ is the constraint c' s.t. $c'\eta = \sum_{d \in D} c\eta[v := d]$. Informally, projecting means eliminating some variables from the support. This is done by associating with each tuple over the remaining variables a semiring element which is the sum of the elements associated by the original constraint to all the extensions of this tuple over the eliminated variables. In short, combination is performed via the multiplicative operation of the semiring, and projection via the additive one.

Solutions. A *solution* of an SCSP $P = \langle C, con \rangle$ is the constraint $Sol(P) = (\bigotimes C) \Downarrow_{con}$. That is, we combine all constraints, and then project over the variables in con. In this way we get the constraint with support (not greater than) con which is "induced" by the entire SCSP. Note that when all the variables are of interest we do not need to perform any projection.

For example, the solution of the fuzzy CSP of Fig. 1 associates a semiring element to every domain value of variable x. Such an element is obtained by first combining all the constraints together. For instance, for the tuple $\langle a, a \rangle$ (that is, $x = y = a$), we have to compute the minimum between 0.9 (which is the value assigned to $x = a$ in constraint c_1), 0.8 (which is the value assigned to $\langle x = a, y = a \rangle$ in c_2) and 0.9 (which is the value for $y = a$ in c_3). Hence, the resulting value for this tuple is 0.8. We can do the same work for tuple $\langle a, b \rangle \to 0.2$, $\langle a, c \rangle \to 0.2$, $\langle b, a \rangle \to 0$, $\langle b, b \rangle \to 0$, $\langle b, c \rangle \to 0.1$, $\langle c, a \rangle \to 0.8$, $\langle c, b \rangle \to 0.2$ and $\langle c, c \rangle \to 0.2$. The obtained tuples are then projected over variable x, obtaining the solution $\langle a \rangle \to 0.8$, $\langle b \rangle \to 0.1$ and $\langle c \rangle \to 0.8$.

3 Symmetry in Soft CSPs

Using an approach similar to [2], we can define two notions of *Semantic symmetry*.

Definition 1 (Symmetry for satisfiability). *Consider two domain values b and a for a variable v and the set of constraints C; we say that b and a are* symmetrical for satisfiability ($a \approx b$) *if and only if*

$$\forall \alpha, \exists \eta, \eta' : \bigotimes C\eta[v := a] = \alpha \iff \bigotimes C\eta'[v := b] = \alpha$$

Informally, two domain values a and b are *symmetrical for satisfiability* if whenever the assignment $v := a$ ($v := b$) leads to a solution with value α, then, we can also obtain a solution with the same value α using the assignment $v := b$ ($v := a$).

Definition 2 (Symmetry for all solutions). *Consider two domain values b and a for a variable v and the set of constraints C; we say that b and a are* symmetrical (w.r.t. the constraints C) ($a \simeq b$) *if and only if*

$$\exists \phi, \eta', \eta'' : \forall \eta : \phi(\eta[v := a]) = \eta'[v := b], \textit{ and } \phi(\eta[v := b]) = \eta''[v := a],$$
$$\wedge$$
$$\bigotimes C\eta[v := a] = \bigotimes C\phi(\eta[v := a]) \wedge \bigotimes C\eta[v := b] = \bigotimes C\phi(\eta[v := b]).$$

Informally, two domain value a and b are *symmetrical (w.r.t. the constraints C)* if whenever the assignment $\eta[v := a]$ ($\eta[v := b]$) leads to a solution with value α, then there is also an assignment $\phi(\eta[v := a])$ ($\phi(\eta[v := b])$) leading to the same semiring value α.

Clearly symmetry for all solutions implies symmetry for satisfiability.

Theorem 1. *Symmetry for all solutions implies symmetry for satisfiability.*

Since finding the mapping ϕ is one of the most important steps when looking for symmetry, it could be useful to reformulate the definition of symmetry for all solutions using equivalent propositions.

Proposition 1. *The following propositions describing the notion of symmetry for all solutions ($a \simeq b$) are equivalent:*

$$\begin{gathered}\exists \phi, \eta', \eta'' : \forall \eta : \phi(\eta[v := a]) = \eta'[v := b], \textit{ and } \phi(\eta[v := b]) = \eta''[v := a],\\ \wedge \\ \bigotimes C\eta[v := a] = \bigotimes C\phi(\eta[v := a]) \wedge \bigotimes C\eta[v := b] = \bigotimes C\phi(\eta[v := b]);\end{gathered} \tag{1}$$

$$\begin{gathered}\exists \phi, \forall \eta : \bigotimes C\eta[v := a] = \bigotimes C\phi(\eta[v := a])[v := b]\\ \wedge \\ \bigotimes C\eta[v := b] = \bigotimes C\phi(\eta[v := b])[v := a];\end{gathered} \tag{2}$$

$$\begin{gathered}\exists \phi, \forall \eta : \bigotimes C\eta[v := a] = \bigotimes C\phi(\eta)[v := b]\\ \wedge \\ \bigotimes C\eta[v := b] = \bigotimes C\phi(\eta)[v := a];\end{gathered} \tag{3}$$

It also important to notice that, similar to the crisp case, the notion of Interchangeability for Soft CSPs [9] is a specific type of symmetry for all solutions obtained using as ϕ the identity function.

Theorem 2. *Consider two domain values b and a, for a variable v and the set of constraints C. If b is fully interchangeable with a on v ($FI_v(a/b)$) (that is, for all assignments η, we have $\bigotimes C\eta[v := a] = \bigotimes C\eta[v := b]$ [9]) then a and b are* symmetrical (w.r.t. the constraints C) ($a \simeq b$). *In particular, the symmetry hold using as ϕ the identity function.*

Symmetries in SCSPs are rarer than in classical CSPs. For this reason using a notion of threshold (similar to that defined by Bistarelli *et al.* [9]) is useful.

Definition 3 (Threshold symmetry for satisfiability). *Consider two domain values b and a for a variable v, the set of constraints C and a threshold $\bar{\alpha}$; we say that b and a are* $_{\bar{\alpha}}$symmetrical for satisfiability ($a \approx_{\bar{\alpha}} b$) *if and only if*

$$\forall \alpha \geq \bar{\alpha}, \exists \eta, \eta' : \bigotimes C\eta[v := a] = \alpha \iff \bigotimes C\eta'[v := b] = \alpha$$

Informally, two domain values a and b are $_{\alpha}$symmetrical for satisfiability if whenever the assignment $v := a$ ($v := b$) leads to a solution with value $\alpha \geq \bar{\alpha}$, then, there is also a way to obtain a solution with the same value α using the assignment $v := b$ ($v := a$).

Definition 4 (Threshold symmetry for all solutions). *Consider two domain values b and a for a variable v, the set of constraints C and a threshold $\bar{\alpha}$; we say that b and a are* $_{\bar{\alpha}}$symmetrical for all solutions ($a \simeq_{\bar{\alpha}} b$) *if and only if*

$$\exists \phi, \eta', \eta'', \alpha, \alpha' \geq \bar{\alpha} : \forall \eta : \phi(\eta[v := a]) = \eta'[v := b], \text{ and } \phi(\eta[v := b]) = \eta''[v := a],$$
$$\wedge$$
$$\bigotimes C\eta[v := a] = \alpha \wedge \bigotimes C\phi(\eta[v := a]) = \alpha$$
$$\wedge$$
$$\bigotimes C\eta[v := b] = \alpha' \wedge \bigotimes C\phi(\eta[v := b]) = \alpha'.$$

Informally, two domain values a and b are $_{\bar{\alpha}}$*symmetrical for all solutions* if whenever the assignment $\eta[v := a]$ ($\eta[v := b]$) leads to a solution whose semiring value is $\alpha \geq \bar{\alpha}$ ($\alpha' \geq \bar{\alpha}$), then there is also a solution $\eta' = \phi(\eta[v := a])$ ($\eta'' = \phi(\eta[v := b])$) that has the same semiring value α (α').

We can prove that the number of symmetries increases when we increase the threshold level.

Theorem 3. *Given two domain element a and b and two thresholds $\alpha_1 \leq \alpha_2$. Then,*

- *if $a \approx_{\alpha_1} b$, then $a \approx_{\alpha_2} b$;*
- *if $a \simeq_{\alpha_1} b$, then $a \simeq_{\alpha_2} b$.*

4 Examples

The example problem that will be studied here is based on the soft n-queens problem [7]. The example is a generalization of the usual n-queens problem, which can be found in [28]. The classical formulation requires that n queens are placed on a $n \times n$ chess-board in such a way that they do not attack each other. In this formulation, we may allow attacking queens, but we give a higher preference to solutions where queens attacking each other are farther apart. In order to formulate this problem as a constraint satisfaction problem, the location of the queens can be given by variables, and the *"do not attack each other"* requirement can be expressed in terms of a number of constraints. A simple way to do this is to assign a variable to each queen[1].

As the n queens must be placed in n different columns, we can identify each queen by its column, and represent its position by a variable which indicates the row of the queen in question. Let v_i stand for the row of the queen in the i-th column. The domain of each of the variables $v_1, \ldots, v_n$ is $\{1, 2, \ldots, n\}$. For any two different variables the following two constraints must hold, expressing that the queens should be in different rows and on different diagonals:

$$v_i \neq v_j$$
$$v_i - v_j \neq i - j$$

If we want to use soft constraints, the previous crisp constraints must to be assigned an element of the semiring. So, whenever $v_i = v_j$, instead of giving the boolean value $false$ we assign the fuzzy value i–j/n. This means that the farther apart the two queens are, the higher this value will be. The same reasoning also holds for the two diagonals; in this case, when v_i–$v_j = i$–j we assign the value i–j/n.

Notice that each solution of this generalized n-queens problem has a semiring value which is obtained by minimizing the semiring values of all its constraints. This comes from the choice of the fuzzy semiring, where the multiplicative operation is the min. Therefore, if a solution contains three pairs of attacking queens, each of the pairs will have a semiring value given by one of the constraints, and the value of this solution will be the minimum of these three values. Different solutions are then ordered using the other semiring operation, which in this case is the max. Note that this same problem can be solved also with a different semiring, obtaining a different way to compute a solution and a different ordering. For example, we could have chosen the semiring $\{\mathcal{R} \cup +\infty, min, +, +\infty, 0\}$, where the value of each solution would have been obtained by summing the values of each attacking pair, and solutions would have been compared using the min operator.

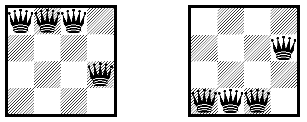
Let's now fix $n = 4$ and illustrate the definitions of the previous section using this example. Clearly we can see that, as in the crisp case, for any v_i with $i = 1, \ldots, 4$, we have $1 \simeq 4$ and $2 \simeq 3$. We have, in fact, for any configuration η,

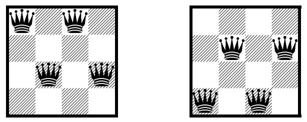
$$\bigotimes C\eta[v_1 := 1] = \bigotimes C\phi^v(\eta)[v_1 := 4]$$
$$\wedge$$
$$\bigotimes C\eta[v_1 := 4] = \bigotimes C\phi^v(\eta)[v_1 := 1];$$

where $\phi^v(\{v_1 := a_1, v_2 := a_2, v_3 := a_3, v_4 := a_4\}) = \{v_1 := 4 - a_1 + 1, v_2 := 4 - a_2 + 1, v_3 := 4 - a_3 + 1, v_4 := 4 - a_4 + 1\}$[2]. The same happens for the variables $v_2, \ldots, v_4$ and for the pair $2 \simeq 3$. In Fig. 2 some mappings are presented.

[1] Note that this choice already eliminates some possible symmetries.

[2] Notice also that this holds for any of the formulas in Proposition 1.


(a) A solution η with level $\frac{1}{4}$ and its symmetric equivalent ($1 \simeq 4$ for v_1).


(b) A solution η with level $\frac{1}{2}$ and its symmetric equivalent ($1 \simeq 4$ for v_1).

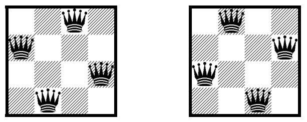

(c) A solution η with level 1 and its symmetric equivalent ($2 \simeq 3$ for v_1).

Fig. 2. Some mappings showing $1 \simeq 4$ and $2 \simeq 3$ for v_1.

Let's now consider the notion of ${}_{\alpha}$symmetry. For $n = 4$ we have many configurations with semiring value $\frac{1}{4}$. We only present in Fig. 4 the configurations whose semiring value is greater or equal than $\frac{1}{2}$. All the configurations with levels $\frac{1}{2}$ and 1 can be obtained by applying the two geometric symmetries:

1. vertical symmetry $\phi^v(\{v_1 := a_1, \ldots, v_i := a_i, \ldots, v_n := a_n\}) = \{v_1 := n - a_1 + 1, \ldots, v_i := n - a_i + 1, \ldots, v_n := n - a_n + 1\}$
2. horizontal symmetry $\phi^h(\{v_1 := a_1, \ldots, v_i := a_i, \ldots, v_n := a_n\}) = \{v_1 := a_n, \ldots, v_i := a_{n-i+1}, \ldots, v_n := a_1\}$

Notice also that in the soft 4-queen problem there are solutions with semiring value $\frac{3}{4}$ (see Fig. 3) but these are excluded from our model (since we represent one queen in each column in our model).

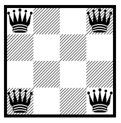
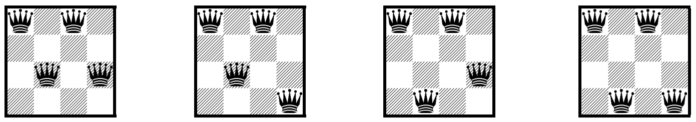


Fig. 3. A solution with level $\frac{3}{4}$ not permitted in our model.

We want to show that when using a threshold $\bar{\alpha}$, more symmetries can be found. Suppose we fix $\bar{\alpha} = \frac{1}{2}$. By using Definition 4, looking for ${}_{\bar{\alpha}}$symmetry means that we need to check if there exists a mapping ϕ s.t. $a \simeq b$, but only when a and b are involved in a solution greater than the threshold $\bar{\alpha}$. Since in our example we want look for ${}_{\frac{1}{2}}$symmetries, it is enough to focus attention to the configurations depicted in Fig. 4.

We claim that $1 \simeq_{\frac{1}{2}} 3$ and $2 \simeq_{\frac{1}{2}} 4$ for variable v_2. To prove this, we need to give a mapping ϕ s.t. for any configuration η s.t. $\bigotimes C\eta[v_1 := 1] \geq \frac{1}{2}$ we have $\bigotimes C\eta[v_1 := 1] = \bigotimes C\phi(\eta)[v_1 := 3]$ and the opposite one (the same happens for $2 \simeq_{\frac{1}{2}} 4$). Let's consider the mapping ϕ similar to ϕ^v, s.t.


(a) Solutions with level $\frac{1}{2}$.

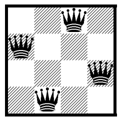
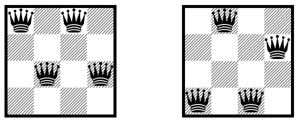
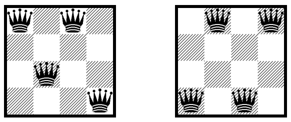

(b) Solution with level 1.

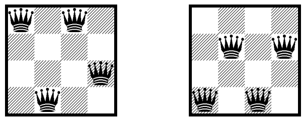
Fig. 4. Solutions for $n = 4$.

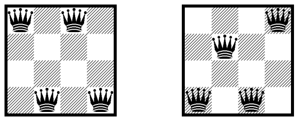
- $\phi(2,4,1,3) = (3,1,2,4)$, and $\phi(3,1,4,2) = (2,4,1,3)$,
- when restricted to variable v_1, v_3, and v_4 ($\phi_{\downarrow\{v_1,v_3,v_4\}} = \phi^v_{\downarrow\{v_1,v_3,v_4\}}$), and
- when restricted to v_2 ($\phi_{\downarrow\{v_2\}}(1) = 3, \phi_{\downarrow\{v_2\}}(2) = 4, \phi_{\downarrow\{v_2\}}(3) = 1, \phi_{\downarrow\{v_2\}}(4) = 2$.

The mapping just defined, satisfies the condition of symmetry for all solutions greater than $\frac{1}{2}$. Fig. 4 illustrates the mappings. Obviously the semiring value associated with a solution and its transformed equivalent is the same.


(a) The mapping of the first solution with level $\frac{1}{2}$.


(b) The mapping of the second solution with level $\frac{1}{2}$.


(c) The mapping of the third solution with level $\frac{1}{2}$.


(d) The mapping of the fourth solution with level $\frac{1}{2}$.

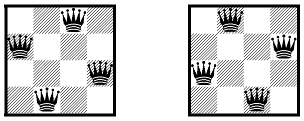

(e) The mapping of the solution with level 1.

Fig. 5. The mappings of the solutions with level $\geq \frac{1}{2}$.

Obviously, we can check that the mapping maintains the same solution level, so $1 \simeq_{\frac{1}{2}} 3$ and $2 \simeq_{\frac{1}{2}} 4$. Similarly, we also have $1 \simeq_1 3$ and $2 \simeq_1 4$ as an example of Theorem 3.

5 Experiments

In this section we present some empirical results supporting the theoretical framework presented in Section 3. In particular, Section 5.1 demonstrates that if values a and b for variable v are α_1 threshold symmetric, they are also α_2 threshold symmetric, where $\alpha_1 \leq \alpha_2$, and that the number of mappings ϕ that satisfy the definition for threshold symmetry for all solutions (Definition 4), increases with larger threshold value $\bar{\alpha}$.

In Section 5.2 we present results from an implementation of a Symmetry Breaking During Search algorithm that utilizes soft symmetries to reduce search effort and the number of solutions produced. Experimental results confirm the improvement in the search performance, and show a significant reduction in the number of distinct solutions found.

5.1 Counting Soft Symmetries

In crisp CSPs, the only mappings which can be used to find symmetrical values in the n-queens problem are the geometric mappings, for example, ϕ^v and ϕ^h, the vertical and horizontal axial symmetries, respectively. In the soft CSP framework, we can utilize the notion of threshold symmetry to find many more mappings, and hence more symmetrical values, which allow symmetry breaking methods to break more symmetries at each step.

In our experimental evaluations, we use a small subset of all possible mappings ϕ to test for threshold symmetry between two values a and b for a variable v. In particular, we chose to systematically generate the subset of all possible mappings for the soft queens problem in which domain values are mapped directly to other domain values *irrespective* of what variable they are assigned to. We computed this subset by generating $n!$ permutations of the domain values and mapping directly between values in the original domain and the corresponding position in the permuted one. This set of mappings is a very small subset of all the possible mappings ϕ among chess-board configurations; for the sake of computational tractability we chose to use this subset to enable us to find useful mappings. This subset allowed us to find a significant number of non-geometric mappings, and provides us with useful results. Many other methods of generating a manageable subset of mappings are possible. Using a larger set of mappings would potentially give rise to more symmetries, which is one possible avenue for future work.

In Table 1 we show the symmetrical values found for v_1 using this set of mappings for various values of threshold $\bar{\alpha}$ in the 5-Queens problem; we present results for values 1 and 3 as examples. We can see that at lower levels of $\bar{\alpha}$, we identify, for example, values 1 and 5 as symmetrical to 1 for v_1. At all levels of $\bar{\alpha}$ greater than this, we also identify these values as symmetrical, supporting Theorem 3.

Table 2 presents results attained by systematically evaluating the threshold symmetry for all solutions (Definition 4) for all (v, a, b, ϕ) combinations, using the set of mappings we generated as discussed above. Results shown in the table indicate the number of mappings that satisfy the definition for the specified value of $\bar{\alpha}$. The results clearly show that the number of mappings which give rise to symmetries is much larger

Table 1. Table of symmetrical values for $a = 1$ and $a = 3$ for v_1 in 5-Queens problem at various values of threshold $\bar{\alpha}$.

$\bar{\alpha}$	$(v_1 := a)$	symmetric values
0.2	1	$\{1, 5\}$
	3	$\{3\}$
0.4	1	$\{1, 5\}$
	3	$\{3\}$
0.6	1	$\{1, 2, 3, 4, 5\}$
	3	$\{1, 2, 3, 4, 5\}$
0.8	1	$\{1, 2, 3, 4, 5\}$
	3	$\{1, 2, 3, 4, 5\}$
1.0	1	$\{1, 2, 3, 4, 5\}$
	3	$\{1, 2, 3, 4, 5\}$

at higher thresholds, a useful property when using symmetrical values to guide search in soft CSPs.

Table 2. Number of times threshold symmetry for all solutions definition is satisfied with threshold $\bar{\alpha} = \frac{1}{n}$ and $\bar{\alpha} = 1$ when iterating through all (v, a, b, ϕ) combinations for soft queens problems of various sizes.

n	$\bar{\alpha} = \frac{1}{n}$	$\bar{\alpha} = 1$
2	6	6
3	15	27
4	24	68
5	40	100
6	54	1620

5.2 Exploiting Soft Symmetries during Search

To demonstrate the utility of the ideas developed in this paper we implemented an algorithm to break ${}_{\alpha}$symmetries in soft CSPs. In this algorithm we attempt to break symmetry in the search space to avoid searching for solutions which are symmetrical to solutions of the same (or higher) consistency which we have already found. Our implementation is based on the Symmetry Breaking During Search (SBDS) approach [20].

We implemented this search algorithm by augmenting a simple backtracker to break symmetry during search by avoiding sections of the search space which are symmetrical to those we have already successfully explored. In this way we can significantly reduce the search effort required to find a set of solutions and reduce the number of distinct solutions produced by providing a set of representative solutions instead of an exhaustive list of all possible solutions of the required consistency.

Table 3. Results for our Soft SBDS Backtracker

		Soft SBDS		*Backtracker*	
n	α	#bts	#sols	#bts	#sols
2	1	2	0	2	0
2	0.5	2	1	4	4
3	1	4	0	4	0
3	0.667	3	1	5	2
3	0.333	14	8	27	27
4	1	10	1	13	2
4	0.75	10	1	13	2
4	0.5	11	6	26	16
4	0.25	30	16	256	256
5	1	4	1	43	10
5	0.8	4	1	43	10
5	0.6	4	1	43	10
5	0.4	103	66	233	184
5	0.2	363	243	3125	3125
6	1	87	2	131	4
6	0.833	50	6	155	32
6	0.667	67	20	197	70
6	0.5	124	61	358	198
6	0.333	485	304	3019	2642
6	0.167	1092	729	46656	46656

To enable us to prune symmetric states, we maintain a value exclusion set for each level of the search tree. Each time that we find a solution of the required consistency α, we update these exclusion sets with values $_\alpha$symmetric to the relevant value from that solution. To ensure that symmetries act locally, we empty value exclusion sets for subsequent levels of the search tree each time we backtrack to a choice point.

This approach does not improve on the effort required to find one solution to a soft CSP. However, in soft CSPs our goal is usually to find the best solution(s). Therefore, if we find one solution of a high level of consistency we do not need to search for states which are symmetrical to this solution, significantly reducing search effort. Furthermore, we also reduce the effort involved in finding *all* solutions to a soft CSP.

The set of symmetrical values used for this algorithm is pre-computed by following an approach similar to that outlined above for computing the results in Table 2. In this case, we evaluate the threshold symmetry for all solutions definition for all $(v, a, b, \phi, \bar{\alpha})$ combinations possible in the relevant instance of the soft queens problem. We then store each (v, a, b) triple which is found to be $_\alpha$symmetric, thus avoiding the significant overhead of searching through a large set of mappings each time we wish to add to our value exclusion sets. With the current lack of an efficient means of identifying useful non-geometric mappings, we see this approach as a reasonable compromise between the added search efficiency gained by utilizing $_\alpha$symmetries and the off-line overhead of computing these symmetries.

The set of the solutions produced by this algorithm can be seen as a representative subset of all possible solutions of the required consistency, which is a useful method of producing diverse solutions to a loosely constrained problem.

Results achieved using this approach are encouraging: we significantly reduce the number of distinct solutions found and the number of backtracks required to find those solutions. In Table 3 we present results demonstrating the utility of our approach to breaking soft symmetries, particularly in loosely constrained problems. For example, if we examine the results for $n = 6$ and $\alpha = \frac{1}{6}$ we can see that a very large reduction in the number of backtracks is attained to find a small representative subset of a large number of possible solutions.

6 Conclusions

Exploiting symmetry in constraint satisfaction problems has become a very popular topic of research in recent times. The existence of symmetry in a problem has the effect of artificially increasing the size of the search space that is explored by search algorithms. Another significant topic of research has been approaches to reasoning about preferences. As constraint processing applications are becoming more widespread in areas such as electronic commerce, configuration, etc., it is becoming increasingly important that we can reason about preferences in as efficiently a manner as possible.

We have presented an approach to dealing with symmetry in the semiring framework for soft constraints. We demonstrate that breaking symmetries in soft constraint satisfaction problems improves the efficiency of search. The paper contributes to the state-of-the-art in symmetry breaking, as well as in reasoning about preferences.

References

[1] R. Backofen and S. Will. Excluding symmetries in concurrent constraint programming. In *Proceedings of CP-99*, LNCS 1520, pages 72–86, 1999.

[2] Belaid Benhamou. Study of symmetry in constraint satisfaction problems. In *Proceedings of CP-94*, 1994.

[3] S. Bistarelli. *Soft Constraint Solving and programming: a general framework.* PhD thesis, Dipartimento di Informatica, Università di Pisa, Italy, mar 2001. TD-2/01.

[4] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and Valued CSPs: Frameworks, properties, and comparison. *Constraints*, 4(3), 1999.

[5] S. Bistarelli, U. Montanari, and F. Rossi. Constraint Solving over Semirings. In *Proc. IJCAI95*, pages 624–630, San Francisco, CA, USA, 1995. Morgan Kaufman.

[6] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of the ACM*, 44(2):201–236, Mar 1997.

[7] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Logic Programming: Syntax and Semantics. *ACM Transactions on Programming Languages and System (TOPLAS)*, 23:1–29, 2001.

[8] S. Bistarelli, U. Montanari, and F. Rossi. Soft concurrent constraint programming. In *Proc. ESOP, April 6 - 14, 2002, Grenoble, France*, LNCS, pages 53–67, Heidelberg, Germany, 2002. Springer-Verlag.

[9] Stefano Bistarelli, Boi Faltings, and Nicoleta Neagu. A definition of interchangeability for soft csps. In *Recent Advances in Constraints*, LNAI 2627. Springer, 2003.

[10] J. Bowen, R. Lai, and D. Bahler. Lexical imprecision and fuzzy constraint networks. In *Proceedings of AAAI-92*, pages 616–621, July 1992.

[11] C.A. Brown, L. Finkelstein, and P.W. Purdon Jr. Backtrack searching in the presence of symmetry. In T. Mora, editor, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 357 of *LNCS*, pages 99–110. Springer-Verlag, 1988.

[12] J. Crawford. A theoretical analysis of reasoning by symmetry in first-order logic. In *Proceedings of the AAAI-92 Workshop on Tractable Reasoning*, 1992.

[13] J. Crawford, G. Luks, M. Ginsberg, and A. Roy. Symmetry breaking predicates for search problems. In *Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning – KR-96*, pages 148–159, 1996.

[14] C. Domshlak, F. Rossi, B. Venable, and T. Walsh. Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques. In *Proceedings of IJCAI-2003*, August 2003.

[15] T. Fahle, S. Schamberger, and M. Sellmann. Symmetry breaking. In *Proceedings of CP-01*, LNCS 2239, pages 93–107, 2001.

[16] P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetries in matrix models. In *Proceedings of CP-02*, LNCS 2470, pages 462–476, 2002.

[17] F. Focacci and M. Milano. Global cut framework for removing symmetries. In *Proceedings of CP-01*, LNCS 2239, pages 75–92, 2001.

[18] E.C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *Proceedings of the AAAI*, pages 227–233, 1991.

[19] I.P. Gent, W. Harvey, and T. Kelsey. Groups and constraints: Symmetry breaking during search. In *Proceedings of CP-02*, LNCS 2470, pages 415–430, 2002.

[20] I.P. Gent and B.M. Smith. Symmetry breaking in constraint programming. In W. Horn, editor, *Proceedings of ECAI-2000*, pages 599–603. IOS Press, 2000.

[21] U. Junker. Preference programming for configuration. In *Proceedings of the 4th International Workshop on Configuration (IJCAI-01)*, pages 50–56, August 2001.

[22] I. McDonald and B. Smith. Partial symmetry breaking. In *Proceedings of CP-02*, LNCS 2470, pages 431–445, 2002.

[23] Pedro Meseguer and Carme Torras. Exploiting symmetries within constraint satisfaction search. *Artificial Intelligence*, 129(1–2):133–163, 2001.

[24] J.-F. Puget. On the satisfiability of symmetrical constrained satisfaction problems. In *Proceedings of ISMIS-93*, LNAI 689, pages 350–361, 1993.

[25] J.-F. Puget. Symmetry breaking revisited. In *Proceedings of CP-02*, LNCS 2470, pages 446–461, 2002.

[26] T. Schiex. Possibilistic constraint satisfaction problems, or "how to handle soft constraints?". In *Proc. 8th Conf. of Uncertainty in AI*, pages 269–275, 1992.

[27] B. Smyth and P. McClave. Similarity vs. diversity. In *Proceedings ICCBR-2001*, LNCS 2080, pages 347–361, 2001.

[28] P. van Hentenryck. *Constraint Satisfaction in Logic Programming*. MIT Press, Cambridge, MA, USA, 1989.