Exploiting single-cycle symmetries in Branch-and-Prune algorithms

Vicente Ruiz de Angulo and Carme Torras

Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Llorens i Artigas 4-6, 08028-Barcelona, Spain.
{ruiz, torras}@iri.upc.edu

Abstract. As a first attempt to exploit symmetries in continuous constraint problems, we focus on permutations of the variables consisting of one single cycle. We propose a procedure that takes advantage of these symmetries by interacting with a Branch-and-Prune algorithm without interfering with it. A key concept in this procedure are the classes of symmetric boxes formed by bisecting a n-dimensional cube at the same point in all dimensions at the same time. We quantify these classes as a function of n. Moreover, we propose a simple algorithm to generate the representatives of all these classes for any number of variables at very high rates. A problem example from the chemical field and a kinematics solver are used to show the performance of the approach in practice.

1 Symmetry in Continuous Constraints Problems

Symmetry exploitation in discrete constraint problems has received a great deal of attention lately [6, 4, 5, 11]. On the contrary, symmetries have been largely disregarded in continuous constraint solving, despite the important growth in both theory and applications that this field has recently experienced [12, 1, 9].

Continuous (or numerical) constraint solving is often tackled using Branch-and-Prune algorithms [13], which iteratively locate solutions inside an initial domain box, by alternating box subdivision (branching) and box reduction (pruning) steps. Motivated by a molecular conformation problem, in this paper we deal with the most simple type of box symmetry, namely that in which domain variables (i.e., box dimensions) undergo a single-cycle permutation leaving the constraints invariant. This can be seen, thus, as a form of constraint symmetry in the terminology introduced in [3].

We are interested in solving the following general Continuous Constraint Satisfaction Problem (CCSP): Find all points $\mathbf{x} = (x_1, \ldots, x_n)$ lying in an initial box of $\mathbb{R}^n$ satisfying the constraints $f_1(\mathbf{x}) \in C_1 \ , \ldots, f_m(\mathbf{x}) \in C_m$, where f_i is a function $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, and C_i is an interval in $\mathbb{R}$.

We assume the problem is tackled using a Branch-and-Prune (B&P) algorithm. The only particular feature that we require of this algorithm is that it has to work with boxes in $\mathbb{R}^n$.

We say that a function $s : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a point symmetry of the problem if there exists an associated permutation $\sigma \in \Sigma_m$ such that $f_i(\mathbf{x}) = f_{\sigma(i)}(s(\mathbf{x}))$

and $C_i = C_{\sigma(i)},\ \forall i = 1, \ldots, m$. We consider symmetry as a property that relates points that are equivalent as regards to a CCSP. Concretely, from the above definition one can conclude that $\mathbf{x}$ is a solution to the problem iff $s(\mathbf{x})$ is a solution to the problem. Let s and t be two symmetries of a CCSP with associated permutations σ_s and σ_t. It is easy to see that the composition of symmetries $s(t(\cdot))$ is also a symmetry with associated permutation $\sigma_s(\sigma_t(\cdot))$.

An interesting type of symmetries are permutations of the components of $\mathbf{x}$. Let D be a finite set. A cycle of length k is a permutation ψ such that there exist distinct elements $a_1, \ldots a_k \in D$ such that $\psi(a_i) = \psi(a_{(i+1) mod\ k})$ and $\psi(z) = z$ for any other element $z \in D$. Such a cycle is represented as $(a_1, \ldots a_k)$. Every permutation can be expressed as a composition of disjoint cycles. In this paper we focus on a particular type of permutations, namely those constituted by a single cycle. In its simplest form, this is $s(x_1, x_2, \ldots x_n) = (x_{\theta(1)}, x_{\theta(2)}, \ldots x_{\theta(n)}) = (x_2, x_3 ... x_n, x_1)$, where $\theta(i) = (i + 1)\ mod\ n$.

Example: $n = 3, m = 4, \mathbf{x} = (x_1, x_2, x_3) \in [-1, 1] \times [-1, 1] \times [-1, 1]$,

$$
\begin{aligned}
f_1(\mathbf{x}): &\quad x_1^2 + x_2^2 + x_3^2 \in [5, 5] \equiv\ x_1^2 + x_2^2 + x_3^2 = 5 \\
f_2(\mathbf{x}): &\quad 2x_1 - x_2 \in [0, \infty] \equiv\ 2x_1 - x_2 \geqslant 0 \\
f_3(\mathbf{x}): &\quad 2x_2 - x_3 \in [0, \infty] \equiv\ 2x_2 - x_3 \geqslant 0 \\
f_4(\mathbf{x}): &\quad 2x_3 - x_1 \in [0, \infty] \equiv\ 2x_3 - x_1 \geqslant 0
\end{aligned}
$$

There exists a symmetry $s(x_1, x_2, x_3) = (x_2, x_3, x_1)$. The constraint permutation associated to s is $\sigma(1) = 1$, $\sigma(2) = 3$, $\sigma(3) = 4$, and $\sigma(4) = 2$.

For $n \geqslant 3$ there is never a unique symmetry for a given problem. If there exists a symmetry s, then for example $s^2(\mathbf{x}) = s(s(\mathbf{x}))$ is another symmetry. In general, using the convention of denoting $s^0(\mathbf{x})$ the identity mapping, $\{s^i(\mathbf{x}), i = 0 \ldots n-1\}$ is the set of different symmetries that can be obtained by composing $s(\mathbf{x})$ with itself, while for $i \geqslant n$ we have that $s^i(\mathbf{x}) = s^{i\ mod\ n}(\mathbf{x})$. Thus, a single-cycle symmetry generates by composition $n-1$ symmetries, excluding the trivial identity mapping. Some of them may have different numbers of cycles. The algorithm presented in this paper deals with all the compositions of a single-cycle symmetry, even if some of them are not single-cycle symmetries. The gain obtained with the proposed algorithm will be shown to be generally proportional to the number of different compositions of the selected symmetry. Therefore, when several single-cycle symmetries exist in a CCSP problem, the algorithm should be used with that generating the most symmetries by composition, i.e., with that having the longest cycle.

2 Box symmetry

Since B&P algorithms work with boxes, we turn our attention now to the set of points symmetric to those belonging to a box $\mathcal{B} \subseteq \mathbb{R}^n$.

Let s be a single-cycle symmetry corresponding to the circular variable shifting θ introduced in the preceding section, and $\mathcal{B} = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n]$ a box in $\mathbb{R}^n$. The *box symmetry* function S is defined as $S(\mathcal{B}) = \{s(\mathbf{x})\ \ s.t.\ \ \mathbf{x} \in$

$\mathcal{B}\} = [\underline{x}_{\theta(1)}, \overline{x}_{\theta(1)}] \times \ldots \times [\underline{x}_{\theta(n)}, \overline{x}_{\theta(n)}] = [\underline{x}_2, \overline{x}_2] \times \ldots \times [\underline{x}_n, \overline{x}_n] \times [\underline{x}_1, \overline{x}_1]$. The box symmetry function has also an associated constraint permutation σ, which is the same associated to s. S^i will denote S composed i times. We say, then, that $S^i(\mathcal{B})$ and $S^j(\mathcal{B})$ are symmetric boxes, $0 \leqslant i, j < n$, $i \neq j$.

As in the case of point symmetry, box symmetry has implications for the CCSP. If there is no solution inside a box $\mathcal{B}$, there is no solution inside any of its symmetric boxes either. A box $\mathcal{B}_f \subseteq \mathcal{B}$ is a solution iff $S^i(\mathcal{B}_f) \subseteq S^i(\mathcal{B})$ is a solution box for any $i \in \{1 \ldots n-1\}$.

Box symmetry is an equivalence relation defining symmetry equivalence classes. Let $R(\mathcal{B})$ be the set of *different* boxes in the symmetry class of $\mathcal{B}$, $R(\mathcal{B}) = \{S^i(\mathcal{B}), i \in \{0, \ldots, n-1\}\}$. We define the *period* $P(\mathcal{B})$ of a box $\mathcal{B}$ as $P(\mathcal{B}) = |R(\mathcal{B})|$. Therefore $R(\mathcal{B}) = \{S^i(\mathcal{B}), i \in \{0, \ldots, P(\mathcal{B})-1\}\}$. For example, for box $\mathcal{B}' = [0,4] \times [2,5] \times [2,5] \times [0,4] \times [2,5] \times [2,5]$, $P(\mathcal{B}') = 3$.

In the following sections we will show that the implications of box symmetry for a CCSP can be exploited to save much computing time in a meta-algorithm that uses the B&P algorithm as a tool without interfering with it.

3 Algorithm to exploit box symmetry classes

The algorithm we will propose to exploit box symmetry makes much use of the symmetry classes formed by bisecting a n-dimensional cube I^n (i.e., of period 1) in all dimensions at the same time and at the same point, resulting in 2^n boxes. We will denote L and H the two subintervals into which the original range I is divided. For example, for $n = 2$, we have the following set of boxes $\{L \times L, L \times H, H \times L, H \times H\}$ whose periods are 1, 2, 2 and 1, respectively. And their symmetry classes are: $\{L \times L\}$, $\{L \times H, H \times L\}$, and $\{H \times H\}$. Representing the two intervals L and H as 0 and 1, respectively, and dropping the $\times$ symbol, the sub-boxes can be coded as binary numbers. Let $\mathbb{SR}_n$ be the set of representatives, formed by choosing the smallest box in binary order from each class. For example, $\mathbb{SR}_2 = \{00, 01, 11\}$. Note that the cube I^n to be partitioned can be thought of as the the set of binary numbers of length n, and that $\mathbb{SR}_n$ is nothing more than a subset whose elements are different under circular shift.

Section 4 will show how many components $\mathbb{SR}_n$ has, how they are distributed and, more importantly, how can they be generated. The symmetry exploitation algorithm we propose below uses the B&P algorithm as an external routine. Thus, the internals of the B&P algorithm do not need to be modified.

The idea is to first divide the initial box into a number of symmetry classes. Next, one needs to process only a representative of each class with the B&P algorithm. At the end, by applying box symmetries to the solution boxes obtained in this way, one would get all the solutions lying in the space covered by the whole classes, i.e., the initial box. The advantage of this procedure is that the B&P algorithm would have to process only a fraction of the initial box. Assuming that the initial box is a n-cube covering the same interval $[x_l, x_h]$ in all dimensions, we can directly apply the classes associated to $\mathbb{SR}_n$. A procedure to exploit single-cycle symmetries in this way is presented in Algorithm 1.

Algorithm 1: CSym algorithm.

Input: A n-cube, $[x_l, x_h] \times \cdots \times [x_l, x_h]$; S: a single-cycle box symmetry; $B\&P$: a B&P algorithm.
Output: A set of boxes covering all solutions.

```
1 SolutionBoxSet ← EmptySet
2 x* ← SelectBisectionPoint(x_l, x_h)
3 foreach b ∈ 𝕊ℝ_n do
4 |   B ← GenerateSubBox(b, x_l, x_h, x*)
5 |   SolutionBoxSet ← SolutionBoxSet ∪ ProcessRepresentative(B)
6 return SolutionBoxSet
```

Algorithm 2: The ProcessRepresentative function.

Input: A box $\mathcal{B}$; S: a single-cycle box symmetry; $B\&P$: a B&P algorithm.
Output: The set of solution boxes contained in $\mathcal{B}$ and its symmetric boxes.

```
1 SolSet ← B&P(B)
2 SymSolSet ← SolSet
3 for i=1: P(B) − 1 do
4 |   SymSolSet ← SymSolSet ∪ ApplySymmetry(SolSet, S^i)
5 return SymSolSet
```

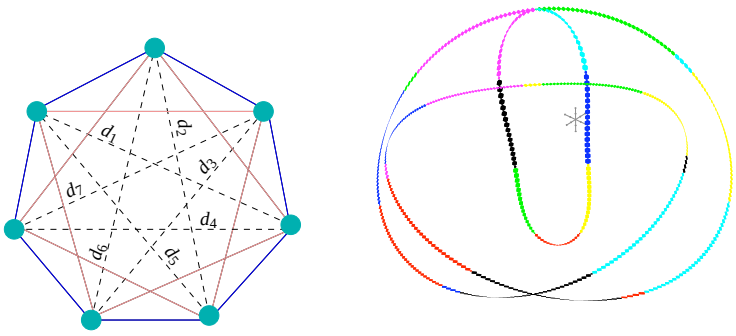
The operator GenerateSubBox($\mathbf{b}, x_l, x_h, x^*$) returns the box corresponding to binary code $\mathbf{b}$ when $[x_l, x_h]$ is the range of the initial box in all dimensions and x^* is the point in which this interval is bisected. The iterations over line 4 of Algorithm 1 generate a set of representative boxes such that, together with their symmetries, cover the initial n-cube.

ProcessRepresentative($\mathcal{B}$) returns all the solution boxes associated to $\mathcal{B}$, that is, the solutions inside $\mathcal{B}$ and inside its symmetric boxes. Since the number of symmetries of $\mathcal{B}$ is $P(\mathcal{B})$, the benefits of exploiting the symmetries of a class representative is proportional to its period.

4 An illustrative example

Molecules can be modeled as mechanical chains by making some reasonable approximations, such as constant bond length and constant angle between consecutive bonds. Finding all valid conformations of a molecule can be formulated as a distance-geometry [2] problem in which some distances between points (atoms) are fixed and known, and one must find the set of values of unknown (variable) distances by solving a set of constraints consisting of equalities or inequalities of determinants formed with subsets of the fixed and variable distances [2].

The problem can be solved using a B&P algorithm [9, 8]. Figure 1(a) displays the known and unknown distances of the cycloheptane, a molecule composed of a ring of seven carbon atoms. The distance between two consecutive atoms of



(a) (b)

Fig. 1. (a) Cycloheptane. Disks represent carbon atoms. Constant and variable distances between atoms are represented with continuous and dashed lines, respectively. (b) Three-dimensional projection of the cycloheptane solutions. The lightest (yellow) boxes are the solutions found inside the representatives using the B&P algorithm (line 1 in Algorithm 2). The other colored boxes are the solutions obtained by applying symmetries to the yellow boxes (line 4 in Algorithm 2).

the ring is constant and equal everywhere. The distance between two atoms connected to a same atom is also known and constant no matter the atoms. The problem has several symmetries. One is $s(d_1, \ldots, d_7) = (d_{\theta(1)}, d_{\theta(2)}, \ldots, d_{\theta(7)}) = (d_2, d_3 \ldots, d_7, d_1)$. When this symmetry is exploited with the CSym algorithm the problem is solved in 4.64 minutes, which compares very favorably with the 31.6 minutes spent when using the algorithm in [8] alone. Thus, a reduction by a factor close to $n = 7$ (i.e., the length of the symmetry cycle) in computing time is obtained, which suggests that the handling of box symmetries doesn't introduce a significant overhead. Figure 1(b) shows a projection into d_1 d_2 and d_3 of the solutions obtained using CSym.

5 Counting and generating box symmetry classes

Let us define some quantities of interest:

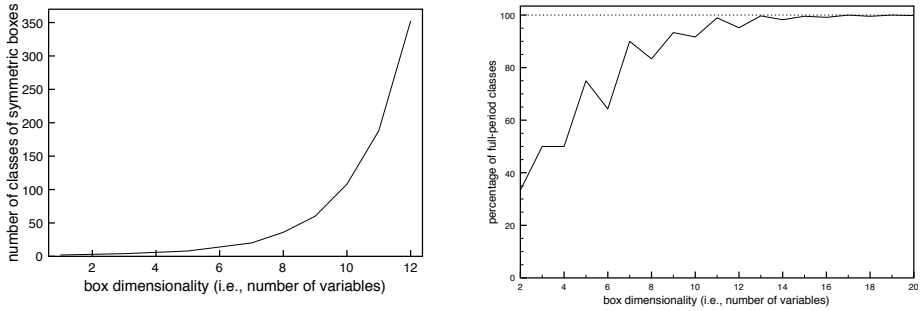
-$\mathcal{N}_n$: Number of elements of $\mathbb{SR}_n$.

-$\mathcal{FP}_n$: Number of elements of $\mathbb{SR}_n$ that correspond to full-period boxes , i.e., boxes of period n.

-$\mathcal{N}_n^p$: Number of elements of $\mathbb{SR}_n$ having period p.

Polya's theorem [7] could be used to determine some of these quantities for a given n by building a possibly huge polynomial and elucidating some of its coefficients. We present a simpler way of calculating them by means of the formulae below. A demonstration of their validity as well as expressions for similar quantities distinguishing the number of 1's can be found in [10]. We begin with $\mathcal{FP}_n$:

$$\mathcal{FP}_n = \frac{2^n}{n} - \sum_{p \in div(n),\ p<n} \frac{p}{n}\, \mathcal{FP}_p. \tag{1}$$



(a) (b)

Fig. 2. (a) Number of elements of $\mathbb{SR}_n$ as a function of n. (b) Percentage of full-period elements in $\mathbb{SR}_n$ as a function of n.

This recurrence has a simple baseline condition: $\mathcal{FP}_1 = 2$. $\mathcal{N}_n$ is calculated with

$$\mathcal{N}_n = \frac{2^n}{n} + \sum_{p \in div(n),\ p<n} \frac{n-p}{n}\ \mathcal{FP}_p. \tag{2}$$

This formula is valid for $n > 1$. The remaining case is $\mathcal{N}_1 = 2$. Finally,

$$\mathcal{N}_n^p = \begin{cases} 0 & \text{if } p \notin div(n) \\ \mathcal{FP}_p & \text{otherwise} \end{cases} \tag{3}$$

Figure 2(a) displays the number of classes ($\mathcal{N}_n$) as a function of n. The curve indicates an exponential-like behavior. Figure 2(b) shows the percentage of full-period classes in $\mathbb{SR}_n$ ($100\ \mathcal{N}_n^n/\mathcal{N}_n$). Note that the percentage of classes with period different from n is significant for low n, but approaches quickly 0 as n grows.

The naive procedure to generate $\mathbb{SR}_n$ involves a huge number of operations. Here we suggest an algorithm capable of calculating $\mathbb{SR}_n$ on the fly. We use a compact coding of the binary numbers representing the boxes consisting of chains of numbers. The first number in the code is the number of 0's appearing before the first 1. The i-th number in the code for $i > 1$ is the number of 0's between the $(i-1)$-th and the i-th 1's. For example, the number 0100010111 is codified as 13100. The length of this numerical codification is the number of 1's of the codified binary number, which has been denoted by m.

Algorithm 3, explained in [10], generates $\mathbb{SR}_{nm}^n$, the subset of the elements of $\mathbb{SR}_n$ having m 1's and period n, from which the whole $\mathbb{SR}_n$ can be obtained, as showed also in [10]. The algorithm outputs a list of codes in decreasing numerical order. For instance, the output obtained when requesting $\mathbb{SR}_{93}^9$ with ClassGen$(6, 1, 1, 3, A)$ is: {600, 510, 501, 420, 411, 402, 330, 321, 312}.

6 Conclusions

We have approached the problem of exploiting symmetries in continuous constraint satisfaction problems using B&P algorithms. Our approach is general

Algorithm 3: ClassGen algorithm.

Input: *sum* is the sum of the digits that remain to be written on the right (from position *pos* to m); *pos*: the index of the next position to be written; *ctrol*: the index of the current control element, whose value cannot be surpassed in the next position; m: the length of the code; A: array where class codes are being generated.

Output: A set of codes representing classes, $\mathbb{SR}$.

```
1  SR ← EmptySet
2  if pos = m then
3      if sum < A[ctrol] then      /* otherwise, SR will remain EmptySet */
4          A[m] ← sum
5          SR ← {A};
6  else
7      if pos ≠ 1 then
8          LowerLimit = 0
9          UpperLimit ← Minimum(A[ctrol], sum)
10     else
11         LowerLimit = ⌈sum/m⌉
12         UpperLimit ← sum
13     for i = UpperLimit to LowerLimit do
14         A[pos] ← i
15         if i = A[ctrol] then                    /* i = A[ctrol] = UpperLimit */
16             SR ← SR ⋃ ClassGen(sum − i, pos + 1, ctrol + 1, m, A)
17         else SR ← SR ⋃ ClassGen(sum − i, pos + 1, 1, m, A); /* i < A[ctrol] */
18 return SR
```

and could be used also with other box-oriented algorithms, such as Branch-and-Bound for nonlinear optimization. The particular symmetries we have tackled are single-cycle permutations of the problem variables.

The suggested strategy is to bisect the domain, the initial n-cube, simultaneously in all dimensions at the same point. This forms a set of boxes that can be grouped in box symmetry classes. A representative of each class is selected to be processed by the B&P algorithm and all the symmetries of the representative are applied to the resulting solutions. In this way, the solutions within the whole initial domain are found, while having processed only a fraction of it –the set of representatives– with the B&P algorithm. The time savings tend to be proportional to the number of symmetric boxes of the representative. Therefore, symmetry exploitation is complete for full-period representatives.

We have also developed a method for generating the classes resulting from bisecting a n-cube. The numerical analysis of the classes revealed that the average number of symmetries of the class representatives tends quickly to n as the number of variables, n, grows. These are good news, since n is the maximum number of symmetries attainable with single-cycle symmetries of n variables.

However, for small n there is still a significant fraction of the representatives not having the maximum number of symmetries. Another weakness of the proposed strategy is the exponential growth in the number of classes as a function of n.

The problems with small and large n should be tackled with a more refined subdivision of the initial domain in box symmetry classes, which is left for near future work. We are also currently approaching the extension of this work to deal with permutations of the problem variables composed of several cycles.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Education and Science through the contract DPI2004-07358 and by the "Comunitat de Treball dels Pirineus" under contract 2006ITT-10004.

References

1. Benhamou F., Goulard F.: Universally Quantified Interval Constraints. Proc. 6th CP (2000) 67-82
2. Blumenthal, L.: Theory and aplications of distance geometry. Oxford University Press, 1953.
3. Cohen D., Jeavons P., Jefferson Ch., Petrie K.E., Smith B.M.: Symmetry Definitions for Constraint Satisfaction Problems. Constraints **11(2-3)** (2006) 115-137
4. Flener P., Frisch A.M., Hnich B., Kiziltan Z., Miguel I., Pearson J., Walsh, T.: Breaking Row and Column Symmetries in Matrix Models. Proc. 8th CP (2002) 462-476
5. Gent I.P., Harvey W., Kelsey T.: Groups and Constraints: Symmetry Breaking During Search. Proc. 8th CP (2002) 415-430
6. Meseguer P., Torras C.: Exploiting symmetries within constraint satisfaction search. Artificial Intelligence **129** (2001) 133-163
7. Polya, G. and Read, R.C.: Combinatorial enumeration of groups, graphs and chemical compounds. Springer-Verlag, New York, 1987.
8. Porta, J.M., Ros Ll., Thomas F., Corcho F., Canto J. and Perez J.J.: Complete maps of molecular loop conformational spaces. Journal of Computational Chemistry (to appear)
9. Porta J.M., Ros Ll., Thomas F., Torras C.: A Branch-and Prune Solver for Distance Constraints. IEEE Trans. on Robotics **21(2)** (2005) 176-187
10. Ruiz de Angulo V., Torras C.: Exploiting single-cycle symmetries in continuous constraint satisfaction problems. IRI_DT 2007/02, June 2007, http://www.iri.upc.edu/people/ruiz/articulos/symmetriesreport1.pdf.
11. Puget J-F.: Symmetry Breaking Revisited. Constraints **10(1)** (2005) 23-46
12. Sam-Haroud D., Faltings B.: Consistency Techniques for Continuous Constraints. Constraints **1(1-2)** (1996) 85-118
13. Vu X-H., Silaghi M., Sam-Haroud D., Faltings B.: Branch-and-Prune Search Strategies for Numerical Constraint Solving. Swiss Federal Institute of Technology (EPFL) LIA-REPORT **7** (2006)