Compositional Derivation of Symmetries for Constraint Satisfaction

Pascal Van Hentenryck[1], Pierre Flener[2], Justin Pearson[2], and Magnus Ågren[2]

[1] Department of Computer Science
Brown University, Box 1910, Providence, RI 02912, USA
pvh@cs.brown.edu

[2] Department of Information Technology
Uppsala University, Box 337, SE - 751 05 Uppsala, Sweden
{pierref,justin,agren}@it.uu.se

Abstract. This paper reconsiders the problems of discovering symmetries in constraint satisfaction problems (CSPs). It proposes a compositional approach which derives symmetries of the applications from primitive constraints. The key insight is the recognition of the special role of global constraints in symmetry detection. Once the symmetries of global constraints are available, it often becomes much easier to derive symmetries compositionally and efficiently. The paper demonstrates the potential of this approach by studying several classes of value and variable symmetries and applying the resulting techniques to two non-trivial applications. The paper also discusses the potential of reformulations and high-level modeling abstractions to strengthen symmetry discovery.

1 Introduction

Many applications in constraint satisfaction exhibit natural symmetries which may significantly increase the difficulty of solving. It is thus not surprising that increased attention has been devoted to symmetry breaking in the last decade.

Recent research has mostly focused on *breaking* symmetries, including general symmetry-breaking schemes (e.g., SBDS [1, 14] and SBDD [8, 10]), their efficient implementations (e.g., [21]), and their specialisations for specific applications (e.g., [3, 20]). There has also been a tendency to abstract some of the techniques from particular applications to classes of CSPs [25] or models [9]. However, this line of research assumes that symmetries are given and ignores the tedious and error-prone task of discovering them.

The detection of symmetries is a research avenue pioneered by Freuder [13] and subsequently investigated by many others. Freuder introduced various forms of value interchangeability and his goal was to *discover* and remove symmetries. Unfortunately, it is not tractable to discover many, apparently simple, classes of symmetries in CSPs arising in practical applications. However, see [5] for results on neighbourhood interchangeability, which is a much weaker form of symmetry than considered in this paper.

This research reconsiders the problem of discovering symmetries from a fundamentally different angle. The key insight is to recognise [24] that global (optimisation) constraints [4, 22, 11] offer significant benefits for deriving symmetries compositionally and efficiently. Global constraints are a fundamental aspect of constraint programming: They capture common combinatorial substructures of practical applications and exploit the substructure semantics to obtain more effective filtering algorithms, linear relaxations, and cooperation schemes between solvers. *The main contribution of this research is to show that, once the symmetries of global constraints are specified, it becomes much simpler to derive the symmetries of an application*. This research can also be seen as shifting the burden of discovering symmetries from users to solver designers who are experts in the underlying combinatorics.

The purpose of this paper is to demonstrate the potential of this research direction. The paper makes the following technical contributions:

1. It considers various classes of symmetries and shows how to derive symmetries compositionally and efficiently, starting from global constraints. They include value and variable symmetries, and symmetries in matrix models.
2. It shows how to apply these results to derive the symmetries of two non-trivial applications: scene allocation and progressive party.
3. It shows how various problem reformulations can improve the accuracy of the derivations and suggests a variety of modeling practices to improve symmetry detection.

These technical results should be viewed as a first (small) step towards a comprehensive automated tool for discovering symmetries. What is particularly interesting however is their ability to handle non-trivial applications already, as well as the various research directions they suggest for modeling languages and reformulation tools.

The rest of the paper is organised as follows. After some preliminaries, the paper shows how constraint and function symmetries can be composed for various forms of interchangeability. The techniques are then illustrated on two applications: scene allocation and the progressive party problem. The next section discusses how problem reformulations improve symmetry detection. Finally, symmetries in matrix models are presented and illustrated.

2 Preliminaries

This section defines the main concepts used in this paper. The definitions are borrowed from [25], which uses them for different purposes. The basic idea is to abstract the set of constraints by a Boolean function which holds if all the constraints are satisfied. Solutions are also represented as functions (assignments), namely from variables to the set of values.

Definition 1. *A* CSP *is a triple $\langle V, D, C\rangle$, where V denotes the set of variables, D denotes the set of possible values for these variables, and $C : (V \to D) \to Bool$ is a constraint that specifies which assignments of values to the variables are solutions. A* solution *to a CSP $\mathcal{P} = \langle V, D, C\rangle$ is a function $\sigma : V \to D$ such that $C(\sigma) = true$. The set of solutions to a CSP $\mathcal{P}$ is denoted by $Sol(\mathcal{P})$.*

Many practical problems involve the optimisation of objective functions and much research in recent years has focused on applying filtering algorithms to prune the resulting "optimisation" constraints (e.g., [23, 11]). In general, in existing languages and systems, these optimisation constraints are expressed using auxiliary variables. However, it is more elegant from a modeling standpoint, and more effective when deriving symmetries, to capture these functions directly.

Definition 2. *A* global function *over variables V and values D is a function $f : (V \to D) \to \mathcal{N}$.*

A constraint optimisation problem (COP) consists of minimising an objective function subject to a set of constraints.

Definition 3. *A* COP *is a quadruple $\mathcal{O} = \langle V, D, C, f \rangle$, where $\mathcal{P} = \langle V, D, C \rangle$ is a CSP and f is a global function over V and D. The* optimal value *f^* of $\mathcal{O}$ is the minimal value of f taken by any solution to $\mathcal{P}$, i.e.,*

$$f^* = \min_{\sigma \in Sol(\mathcal{P})} f(\sigma).$$

An optimal solution *of $\mathcal{O}$ is a solution σ of $\mathcal{P}$ whose objective value is optimal, i.e., $f(\sigma) = f^*$. We use $Sol(\mathcal{O})$ to denote $Sol(\mathcal{P})$ in the following.*

The key idea behind this paper is that symmetries can be systematically derived through composition of CSPs (or COPs). The next definition captures compositions of CSPs formally.

Definition 4. *Let $\mathcal{P}_1 = \langle V, D, C_1 \rangle$ and $\mathcal{P}_2 = \langle V, D, C_2 \rangle$ be two CSPs. The* composition *of $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted by $\mathcal{P}_1 \wedge \mathcal{P}_2$, is the CSP $\mathcal{P} = \langle V, D, C_1 \wedge C_2 \rangle$, whose solutions satisfy $Sol(\mathcal{P}) = Sol(\mathcal{P}_1) \cap Sol(\mathcal{P}_2)$.*

3 Value and Variable Interchangeability

There are many applications in resource allocation and scheduling where the exact values taken by the variables are not important. What is significant is which variables take the *same* values or, in other terms, how the variables are clustered. Other applications exhibit weaker notions of value interchangeability, such as the concept of piecewise value interchangeability where only subsets of values are interchangeable. As shown in [25], these symmetries can be broken efficiently during search and it is thus particularly important to discover them automatically.

Definition 5. *Let $\mathcal{P} = \langle V, D, C \rangle$ be a CSP. $\mathcal{P}$ is* value-interchangeable *if, for each solution $\sigma \in Sol(\mathcal{P})$ and each bijection $b : D \to D$, the function $b \circ \sigma \in Sol(\mathcal{P})$.*

Example 1. Let $V \supseteq \{v_1, v_2, v_3\}$. The CSP $\mathcal{P} = \langle V, D, \mathit{allDifferent}(v_1, v_2, v_3) \rangle$ is value-interchangeable.

We now define piecewise value-interchangeability.

Definition 6. *Let $\mathcal{D} = \{D_1, \ldots, D_n\}$ be a partition of D. A bijection $b : D \rightarrow D$ is* piecewise interchangeable *over $\mathcal{D}$ if $\forall v \in D_i : b(v) \in D_i \ \ (1 \leq i \leq n)$.*

Definition 7. *Let $\mathcal{P} = \langle V, D, C\rangle$ be a CSP and $\mathcal{D}$ be a partition of D. $\mathcal{P}$ is* piecewise-value-interchangeable *(PVI) over $\mathcal{D}$ if, for each solution $\sigma \in Sol(\mathcal{P})$ and each piecewise-interchangeable bijection b over $\mathcal{D}$, $b \circ \sigma \in Sol(\mathcal{P})$.*

Note that, if $\mathcal{P} = \langle V, D, C\rangle$ is value-interchangeable, then it is piecewise-value-interchangeable over $\{D\}$. As a consequence, it is easy to compose these two forms of symmetries.

Example 2. Let $V \supseteq \{v_1, v_2, v_3\}$, $D \ni 1$, and consider a constraint *atmost*$(o, d, \langle v_1, \ldots, v_k\rangle)$ which holds for an assignment σ if there are at most o occurrences of d in $\langle \sigma(v_1), \ldots, \sigma(v_k)\rangle$. The CSP $\langle V, D, atmost(2, 1, \langle v_1, v_2, v_3\rangle)\rangle$ is PVI over $\{\{1\}, D \setminus \{1\}\}$.

Value-interchangeability also applies to global functions, in which case the value of a function must not change under various forms of bijection.

Definition 8. *A global function $f : (V \rightarrow D) \rightarrow \mathcal{N}$ is* value-interchangeable *if, for each assignment $\sigma : V \rightarrow D$ and each bijection $b : D \rightarrow D$, $f(\sigma) = f(b \circ \sigma)$.*

Example 3. Let $V \supseteq \{v_1, \ldots, v_5\}$ and consider global functions of the form $nbDistinct$ $(v_1, \ldots, v_k)$ which, given an assignment σ, return the number of distinct values in $\langle \sigma(v_1), \ldots, \sigma(v_k)\rangle$. The global function *nbDistinct*$(v_1, \ldots, v_5)$ is value-interchangeable.

Definition 9. *Let $\mathcal{D}$ be a partition of D. A global function $f : (V \rightarrow D) \rightarrow \mathcal{N}$ is* piecewise-value-interchangeable *over $\mathcal{D}$ if, for each assignment $\sigma : V \rightarrow D$ and piecewise-interchangeable bijection b over $\mathcal{D}$, $f(\sigma) = f(b \circ \sigma)$.*

These concepts can be generalised to COPs.

Definition 10. *Let $\mathcal{O} = \langle V, D, C, f\rangle$ be a COP. $\mathcal{O}$ is* value-interchangeable *if, for each solution $\sigma \in Sol(\mathcal{O})$ and each bijection $b : D \rightarrow D$, $b \circ \sigma \in Sol(\mathcal{O})$ and $f(\sigma) = f(b \circ \sigma)$.*

Definition 11. *Let $\mathcal{O} = \langle V, D, C, f\rangle$ be a COP and $\mathcal{D}$ be a partition of D. $\mathcal{O}$ is* piecewise-value-interchangeable *over $\mathcal{D}$ if, for each solution $\sigma \in Sol(\mathcal{O})$ and each piecewise-interchangeable bijection b over $\mathcal{D}$, $b \circ \sigma \in Sol(\mathcal{O})$ and $f(\sigma) = f(b \circ \sigma)$.*

In the following, we often assume fixed sets V and D in examples for simplicity and talk directly about the composition and interchangeability of constraints, since they are essentially equivalent to their CSP counterparts.

It is also important to emphasise that all results presented in the next sections have direct counterparts for variable interchangeability. This is due to the fact that the definition of variable interchangeability is essentially similar to value interchangeability. Consider the simplest definition of variable interchangeability.

Definition 12. *Let $\mathcal{P} = \langle V, D, C\rangle$ be a CSP. $\mathcal{P}$ is* variable-interchangeable *if, for each solution $\sigma \in Sol(\mathcal{P})$ and each bijection $b : V \rightarrow V$, the function $\sigma \circ b \in Sol(\mathcal{P})$.*

The difference is the composition order of σ and the bijection (which also has a different signature).

4 Composition of Constraint Symmetries

Value symmetries arise in many applications and can be broken efficiently during search. Unfortunately, there is no general efficient algorithm for computing interchangeable values in CSPs [13]. The key insight is that symmetries can be compositionally inferred from global constraints. More precisely, given two constraints (or CSPs) C_1 and C_2, the symmetries of their composition $C_1 \wedge C_2$ can be inferred automatically from the symmetries of C_1 and C_2. The following result is immediate.

Proposition 1. *Let $\mathcal{P}_1 = \langle V, D, C_1 \rangle$ and $\mathcal{P}_2 = \langle V, D, C_2 \rangle$ be two value-interchangeable CSPs. Then, their composition $\mathcal{P}_1 \wedge \mathcal{P}_2$ is value-interchangeable.*

The following example illustrates the result.

Example 4. Let $V \supseteq \{v_1, \ldots, v_6\}$ and let C_1 and C_2 be the constraints *allDifferent* (v_1, v_2, v_3) and *allDifferent*(v_4, v_5, v_6). Then $C_1 \wedge C_2$ is value-interchangeable.

Note that constraints in practice only "constrain" a subset of the variables, although they are formally defined over all variables. The next result specifies how to compose piecewise-value-interchangeable CSPs.

Proposition 2. *Let $\mathcal{P}_1 = \langle V, D, C_1 \rangle$ and $\mathcal{P}_2 = \langle V, D, C_2 \rangle$ be two CSPs. Assume that $\mathcal{P}_i$ is piecewise-value-interchangeable over partition $\mathcal{D}_i$ of D $(1 \leq i \leq 2)$. Then the composition $\mathcal{P}_1 \wedge \mathcal{P}_2$ is piecewise-value-interchangeable over*

$$\mathcal{D} = \{D_1 \cap D_2 \mid D_1 \in \mathcal{D}_1 \;\&\; D_2 \in \mathcal{D}_2 \;\&\; D_1 \cap D_2 \neq \emptyset\}.$$

Proof. First observe that $\mathcal{D}$ is a partition of D. Now let b be a piecewise-interchangeable bijection over $\mathcal{D}$. We show that b is piecewise-interchangeable over $\mathcal{D}_1$. Indeed, consider a set $D_1 \in \mathcal{D}_1$ and a value $d \in D_1$. By definition of $\mathcal{D}$, there exists $D_2 \in \mathcal{D}_2$ such that $I = D_1 \cap D_2$ and $d \in I$. Since b is piecewise-interchangeable over $\mathcal{D}$, $b(d) \in I \subseteq D_1$ and b is piecewise-interchangeable over $\mathcal{D}_1$. Similarly, we can show that b is piecewise-interchangeable over $\mathcal{D}_2$. As a consequence, if $\sigma \in Sol(\mathcal{P}_1 \wedge \mathcal{P}_2)$, then $b \circ \sigma \in Sol(\mathcal{P}_1)$ and $b \circ \sigma \in Sol(\mathcal{P}_2)$. Hence, $b \circ \sigma \in Sol(\mathcal{P}_1 \wedge \mathcal{P}_2)$.

Example 5. Let $D = \{1, \ldots, 10\}$ and let C_1 and C_2 be the constraints *atmost*$(1, 1, \langle v_1, \ldots, v_5 \rangle)$ and *atmost*$(2, 2, \langle v_1, \ldots, v_5 \rangle)$ which are PVI over $\mathcal{D}_1 = \{\{1\}, \{2, \ldots, 10\}\}$ and $\mathcal{D}_2 = \{\{2\}, \{1, 3, \ldots, 10\}\}$ respectively. The composition $C_1 \wedge C_2$ is PVI over

$$\mathcal{D} = \{\{1\}, \{2\}, \{3, \ldots, 10\}\}.$$

It is important to emphasise that the derivation of symmetries using propositions 1 and 2 is polynomial in $|D|$. As a consequence, the compositional symmetry analysis of a CSP is polynomial in $|D|$ and the number of constraints. Of course, it is not guaranteed to be precise, i.e., it may not report all symmetries in the application. However, whenever global constraints are used to model an application, the symmetries appear naturally and the loss of precision is often avoided. Furthermore, we discuss this later in the paper how reformulations may help in addressing this issue.

5 Composition of Function Symmetries

This section shows how to compose function symmetries from global functions. It also shows how to infer symmetries in COPs and how function symmetries can be used to infer symmetries on numerical constraints.

Proposition 3. *Let f_1 and f_2 be two global functions of signature $(V \to D) \to \mathcal{N}$. If f_1 and f_2 are value-interchangeable, then so are $f_1 \star f_2$, where $\star \in \{+, -, \times\}$.*

Of course, the result can be generalised to other operators.

Example 6. Let $V \supseteq \{v_1, \ldots, v_6\}$ and let f_1 and f_2 be the global functions *nbDistinct* (v_1, v_2, v_3) and *nbDistinct*(v_4, v_5, v_6). Then, the global function $3f_1 + 4f_2$ is value-interchangeable.

Proposition 4. *Let $f_1 : (V \to D) \to \mathcal{N}$ and $f_2 : (V \to D) \to \mathcal{N}$ be two global functions. If f_1 and f_2 are piecewise-value-interchangeable over $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively, then $f_1 \star f_2$, where $\star \in \{+, -, \times\}$, is piecewise-value-interchangeable over*

$$\mathcal{D} = \{D_1 \cap D_2 \mid D_1 \in \mathcal{D}_1 \;\&\; D_2 \in \mathcal{D}_2 \;\&\; D_1 \cap D_2 \neq \emptyset\}.$$

We now show how to derive symmetries for COPs by considering both the constraint and the objective function.

Proposition 5. *Let $\mathcal{O} = \langle V, D, C, f\rangle$ be a* COP *and $\mathcal{P} = \langle V, D, C\rangle$. If $\mathcal{P}$ and f are value-interchangeable, then $\mathcal{O}$ is value-interchangeable. If $\mathcal{P}$ is piecewise-value-interchangeable over partition $\mathcal{D}_1$ of D and f is piecewise-value-interchangeable over partition $\mathcal{D}_2$ of D, then $\mathcal{O}$ is piecewise-value-interchangeable over*

$$\mathcal{D} = \{D_1 \cap D_2 \mid D_1 \in \mathcal{D}_1 \;\&\; D_2 \in \mathcal{D}_2 \;\&\; D_1 \cap D_2 \neq \emptyset\}.$$

In many applications, constraints are built from global functions and arithmetic operators. The next proposition shows how to derive symmetries for such constraints.

Proposition 6. *Let $f : (V \to D) \to \mathcal{N}$ be a global function and $\mathcal{D}$ be a partition of D. If f is piecewise-value-interchangeable over $\mathcal{D}$, then the CSP $\langle V, D, f \approx 0\rangle$ is piecewise-value-interchangeable over $\mathcal{D}$ as well, where $\approx \in \{>, \geq, =, \neq, \leq, <\}$.*

6 Scene Allocation

We now illustrate how these results can be used to detect value symmetries on the scene-allocation problem, which consists of producing a movie at minimal cost by deciding when to shoot scenes. Each scene involves a number of actors and at most 5 scenes a day can be filmed. All actors of a scene must be present on the day the scene is shot. The actors have fees representing the amount to be paid per day they spend in the studio. The goal of the application is to minimise the production costs and an optimal solution is an assignment of scenes to days which minimises the production costs. On some reasonably small instances, a state-of-the-art MIP solver took about 2 minutes and a CP

```
range Scenes = ...;
range Days = ...;
range Actors = ...;
int fee[Actors] = ...;
{Scenes} S[Actors] = ...;
var Days shoot[Scenes];

minimise
  sum(a in Actors)
    fee[a]*nbDistinct(all(s in S[a]) shoot[s])
subject to
  atmost(5,Days,shoot);
```

Fig. 1. The Scene Allocation Model

solver took about 8 minutes for solving the problem. By removing value symmetries during search, the execution time of the CP solver fell to below 10 seconds [26].

It is also interesting to quote [25] here: “It should be apparent that the exact days assigned to the scenes have no importance in this application and are fully interchangeable. What is important is how the scenes are clustered together. Our approach does not aim at discovering this fact; rather it focuses on how to exploit it to eliminate the symmetries it induces.” *The main contribution of this paper is entirely orthogonal: it shows how the value interchangeability of the scene allocation problem can be automatically derived from the properties of the constraints.*

Consider Figure 1 which depicts an OPL-like model for scene allocation, where the instance data is given in a separate file as typical. The first three lines specify the various ranges for scenes, days, and actors. The next two lines specify the fee of each actor and the set of scenes `S[a]` which actor `a` plays in. The next line specifies the variables and `shoot[s]` represents the day assigned to scene `s`. The constraint `atmost(5,Days,shoot)` is a global cardinality constraint which specifies that at most 5 scenes can be shot every day. The objective function sums the fees of each actor, each actor being paid her fee for each different day in which one of her scenes is shot. Indeed, the expression `all(s in S[a]) shoot[s]` collects the variables associated with the scenes of actor `a` in an array of variables, which is used in the function `nbDistinct`. Observe now that constraint `atmost(5,Days,shoot)` is value-interchangeable. The global function `nbDistinct` is also value-interchangeable (see Example 3). By Proposition 3, the objective function is value-interchangeable. Hence, by Proposition 5, the scene-allocation model in Figure 1 is value-interchangeable. In summary, as mentioned earlier, once the value symmetries of the global objects are known, it is possible to derive value symmetries of the entire model using the results of this paper.

It is also useful to stress the benefits of global constraints. The value symmetries derived on the model above are dramatically more complicated to detect on the MIP model. Indeed, the values are not even explicit in that model, which encodes the scene assignment in terms of 0/1 variables.

```
range Boats = ...;
range Parties = ...;
range Periods = ...;
int size[Parties] = ...;
int cap[Boats] = ...;
var Boats b[Parties,Periods];
solve {
  forall(g in Parties)
    allDifferent(all(p in Periods) b[g,p]);
  forall(p in Periods)
    weightedAtmost(size,
                   all(g in Parties) b[g,p],
                   cap);
  forall(i in Parties, j in Parties: j>i)
    meetAtmost(all(p in Periods) b[i,p],
               all(p in Periods) b[j,p],
               1);
};
```

Fig. 2. The Progressive Party Model

7 Progressive Party Problem

The progressive party problem is a traditional benchmark which is often used to compare constraint programming, mathematical programming, and local search. Figure 2 depicts an OPL-like model for this problem, which is a direct translation of the Comet model in [18]. The first three lines specify the ranges, i.e., the boats, the parties, and the periods. The next two lines specify the size of the parties and the capacities of the boats. The variables are declared next and assign a boat `b[g,p]` to party `g` at period `p`. The first set of constraints specifies that a party never visits the same boat twice. The second set of constraints are weighted cardinality constraints which specify that the sizes of the parties visiting a boat during a period cannot exceed the boat capacity. The final set of constraints are again cardinality constraints specifying that two parties meet at most once: a constraint $meetAtmost(\langle v_1, \ldots, v_p\rangle, \langle w_1, \ldots, w_p\rangle, k)$ holds for an assignment σ if $\#\{i \in 1..p \mid \sigma(v_i) = \sigma(w_i)\} \leq k$. Observe that the *allDifferent* constraints are value-interchangeable. The *meetAtmost* constraints are also value-interchangeable. The interesting part in this model are the *weightedAtmost* constraints. A constraint

$$weightedAtmost(\langle s_1, \ldots, s_n\rangle, \langle v_1, \ldots, v_n\rangle, \langle c_1, \ldots, c_m\rangle)$$

holds for an assignment σ if $\forall k \in 1..m : \sum_{i \in S_k} s_i \leq c_k$ where $S_k = \{i \in 1..n \mid \sigma(v_i) = k\}$. This constraint is piecewise-value-interchangeable over $\mathcal{D} = \{D_1, \ldots, D_m\}$, where $D_k = \{i \in 1..m \mid c_i = c_k\}$. As a consequence, our compositional derivation automatically infers that boats with the same capacity are piecewise-value-interchangeable. Note that a similar derivation for the variables infers that parties with the same sizes are piecewise-variable-interchangeable.

8 Reformulations

The symmetry derivations presented earlier can often be strengthened by model reformulations which can be seen as adaptations to constraint satisfaction of “presolve” techniques used in mixed-integer programming [16]. We present two first reformulations, aggregation and projection.

8.1 Aggregation

The symmetry derivations presented earlier may be suboptimal as the following example indicates.

Example 7. Let $V \supseteq \{v_1, \dots, v_3\}$ and $D \supseteq \{1, 2\}$. Constraint $C_1 = \mathit{atmost}(2, 1, \langle v_1, v_2, v_3\rangle)$ is PVI over $\mathcal{D} = \{\{1\}, D \backslash \{1\}\}$. Constraint $C_2 = \mathit{atmost}(2, 2, \langle v_1, v_2, v_3\rangle)$ is PVI over $\mathcal{D} = \{\{2\}, D \setminus \{2\}\}$. By Proposition 2, $C_1 \wedge C_2$ is PVI over $\mathcal{D} = \{\{1\}, \{2\}, D \setminus \{1, 2\}\}$. However, $C_1 \wedge C_2$ is also PVI over $\mathcal{D} = \{\{1, 2\}, D \setminus \{1, 2\}\}$, which is stronger.

This precision loss can be remedied by modeling the problem more globally using, say, a global cardinality constraint [23]. Again, the observation is that *global constraints are fundamental tools to derive stronger symmetries.*

Example 8. Consider a global cardinality constraint $\mathit{atmost}(\langle o_1, \dots, o_k\rangle, \langle d_1, \dots, d_k\rangle, \langle v_1, \dots, v_n\rangle)$ which holds for an assignment σ if there exist at most o_i occurrences of d_i in $\langle \sigma(v_1), \dots, \sigma(v_n)\rangle$ $(1 \leq i \leq k)$. It is PVI over $\{D_1, \dots, D_k, D \backslash (D_1 \cup \dots \cup D_k)\}$ where $D_i = \{d_j \mid o_j = o_i \ \& \ 1 \leq j \leq k\} \quad (1 \leq i \leq k)$. For instance, $\mathit{atmost}(\langle 1, 2, 1\rangle, \langle 1, 2, 3\rangle, \langle v_1, \dots, v_n\rangle)$ is PVI over $\{\{1, 3\}, \{2\}, D \setminus \{1, 2, 3\}\}$ since $D_1 = D_3 = \{1, 3\}$.

Of course, a more global modeling of the problem will likely also lead to better propagation. *As a consequence, automated tools for symmetry detection (and modeling in general) should provide aggregation operators exploiting the semantics of constraints.* They can be specified as follows.

Definition 13. *Let C_1 and C_2 be two constraints of signature $\mathcal{C} = (V \to D) \to \mathit{Bool}$. A* compositional aggregator *is a binary operator $\otimes$ of signature $(\mathcal{C} \times \mathcal{C}) \to \mathcal{C}$ such that $C_1 \otimes C_2$ is a single global constraint equivalent to $C_1 \wedge C_2$.*

Example 9. Let $V \supseteq \{v_1, v_2, v_3\}$, $D \supseteq \{1, 2\}$ and constraints $C_1 = \mathit{atmost}(2, 1, \langle v_1, v_2, v_3\rangle)$ and $C_2 = \mathit{atmost}(2, 2, \langle v_1, v_2, v_3\rangle)$. A compositional aggregator of C_1 and C_2 may return the constraint $\mathit{atmost}(\langle 2, 2\rangle, \langle 1, 2\rangle, \langle v_1, \dots, v_3\rangle)$.

8.2 Projection

Projections, the second class of reformulations considered in this paper, are important in many applications. On the one hand, they are often useful when a general model (e.g., a round-robin sport-scheduling model) is specialised to a specific problem (e.g., the ACC basketball schedule for the 2004 season) by introducing, among others, some fixed decisions. On the other hand, they are useful in deriving dynamic symmetries, i.e., symmetries not present in the original problem but arising after a number of variable assignments. The following example illustrates the significance of projections when deriving symmetries.

Example 10. Let $V = \{v_1, \ldots, v_5\}$, let C_1 be the constraint *atmost*$(\langle 3, 2\rangle, \langle 1, 2\rangle, \langle v_1, \ldots, v_5\rangle)$ and C_2 be $v_1 = 1$. The CSP $\langle V, D, C_1 \wedge C_2\rangle$ is derived to be PVI over $\mathcal{D} = \{\{1\}, \{2\}, D \setminus \{1, 2\}\}$ since C_1 is PVI over $\{\{1\}, \{2\}, D \setminus \{1, 2\}\}$ and C_2 is PVI over $\{\{1\}, D \setminus \{1\}\}$, which is as strong as possible. However, consider $V' = V \setminus \{v_1\}$ and the constraint C defined as *atmost*$(\langle 2, 2\rangle, \langle 1, 2\rangle, \langle v_2, \ldots, v_5\rangle)$. The CSP $\langle V', D, C\rangle$ is PVI over $\mathcal{D} = \{\{1, 2\}, D \setminus \{1, 2\}\}$.

This example indicates that more symmetries may be available on subproblems when some variables are projected out. Moreover, since the assignment of values to variables is the fundamental operation of many search procedures, projections are an important tool to derive symmetries dynamically. *As a consequence, symmetry detection tools should ideally include projection operators exploiting the semantics of primitive constraints.*

Definition 14. *Let C be a constraint of signature $\mathcal{C} = (V \to D) \to Bool$, and $V' = V \setminus \{v\}$. A* projection operator *for $\mathcal{C}$ wrt $v = d$ is a function $\uparrow_{v=d}$ of signature $\mathcal{C} \to \mathcal{C}'$, where $\mathcal{C}' = (V' \to D) \to Bool$, satisfying*

$$Sol(\langle V, D, C \wedge v = d\rangle) = Sol(\langle V, D, C \uparrow_{v=d} \wedge v = d\rangle.$$

The key intuition here is that constraint $C \uparrow_{v=d}$ is only expressed in terms of variables in V' and does not add or remove any solution to the original problem.

9 Symmetries in Matrix Models

This section considers the derivation of variable symmetries in matrix models, which have been found useful in a variety of applications involving symmetries. In particular, we show how the techniques presented earlier apply to the detection of column symmetries in matrix models. (The derivation of row symmetries is similar.) Figure 3 presents a specification of the progressive party problem using matrix modeling. It is essentially similar to the model presented earlier but uses matrices and rows of matrices directly in constraints. We now show how to systematically derive column-interchangeability on this model.

Formally, a matrix M of variables can be modelled as a bijection $X \times Y \to V$, where X are the row indices of M, Y its column indices, and V its set of variables. For clarity, we use traditional notations: $M[i, j]$ denotes the variable in row i and in column j, $M[i]$ row i, and $M[*, j]$ column j. We assume that all matrices are defined over row indices X and column indices Y.

Definition 15. *A* matrix-CSP *(MCSP) is a triple $\langle M, D, C\rangle$, where M is a matrix of variables, D denotes the set of values for these variables, and $C : (M \to D) \to Bool$*

```
range Boats = ...;
range Parties = ...;
range Periods = ...;
int size[Parties] = ...;
int cap[Boats] = ...;
var Boats b[Parties,Periods];
solve {
  forall(g in Parties)
    allDifferent(b[g]);
  weightedAtmost(size,b,cap);
  forall(i in Parties, j in Parties: j>i)
    meetAtmost(b[i],b[j],1);
};
```

Fig. 3. The Progressive Party Matrix Model

specifies which assignments of values to the variables are solutions. A solution *to an MCSP $\mathcal{P} = \langle M, D, C\rangle$ is a function $\sigma : M \to D$ such that $C(\sigma) = true$. The set of solutions to $\mathcal{P}$ is denoted by $Sol(\mathcal{P})$.*

The next definitions specify column interchangeability, a "global" form of variable interchangeability.

Definition 16. *A* column permutation *for a matrix M is a function $\rho : M \to M$ such that*

$$M[i, j] = \rho(M)[i, b(j)] \quad (i \in X \ \& \ j \in Y)$$

for some bijection $b : Y \to Y$.

Definition 17. *An MCSP $\mathcal{P} = \langle M, D, C\rangle$ is* column-interchangeable *if, for each solution $\sigma \in Sol(\mathcal{P})$ and each column permutation $\rho : M \to M$, the function $\sigma \circ \rho \in Sol(\mathcal{P})$.*

Proposition 7. *Let $\mathcal{P}_1 = \langle M, D, C_1\rangle$ and $\mathcal{P}_2 = \langle M, D, C_2\rangle$ be two column-interchangeable MCSPs. Then, their composition $\mathcal{P}_1 \wedge \mathcal{P}_2$ is column-interchangeable.*

Example 11. Consider the matrix model in Figure 3. The constraints *allDifferent* and *meetAtmost* are column-interchangeable. Indeed, the variable (resp. pair) order is not significant in *allDifferent* (resp. *meetAtmost*) and both are applied on rows of the matrix. The global *weightedAtmost* constraint is column-interchangeable, since it applies the same constraint to all columns. It is an aggregation of

```
forall(p in Periods)
  weightedAtmost(size,b[*,p],cap);
```

which cannot be shown column-interchangeable compositionally.

We conclude this section by generalising the results to piecewise interchangeability.

Definition 18. *Let $\mathcal{Y}$ be a partition over Y. A* piecewise column permutation *over $\mathcal{Y}$ for a matrix M is a function $\rho : M \to M$ such that*

$$M[i,j] = \rho(M)[i, b(j)] \quad (i \in X \;\&\; j \in Y)$$

for some piecewise-interchangeable bijection b over $\mathcal{Y}$.

Definition 19. *Let $\mathcal{Y}$ be a partition over Y. An MCSP $\mathcal{P} = \langle M, D, C \rangle$ is* piecewise-column-interchangeable *over $\mathcal{Y}$ if, for each solution $\sigma \in Sol(\mathcal{P})$ and each piecewise column permutation ρ over $\mathcal{Y}$, the function $\sigma \circ \rho \in Sol(\mathcal{P})$.*

Proposition 8. *Let $\mathcal{P}_1 = \langle M, D, C_1 \rangle$ and $\mathcal{P}_2 = \langle M, D, C_2 \rangle$ be two piecewise-column-interchangeable MCSPs over $\mathcal{Y}_1$ and $\mathcal{Y}_2$ respectively. Then, their composition $\mathcal{P}_1 \wedge \mathcal{P}_2$ is piecewise-column-interchangeable over*

$$\mathcal{Y} = \{Y_1 \cap Y_2 \mid Y_1 \in \mathcal{Y}_1 \;\&\; Y_2 \in \mathcal{Y}_2 \;\&\; Y_1 \cap Y_2 \neq \emptyset\}.$$

These results naturally generalise to matrix-COPS.

10 Conclusion

This paper reconsidered the problem of discovering symmetries in constraint satisfaction problems by exploiting one of the fundamental aspects of constraint programming: the ability of global constraints to capture combinatorial substructures. The paper showed that, once the symmetries of global constraints are specified, various classes of symmetries can be derived precisely and efficiently in a compositional fashion. The paper studied value and variable interchangeability, as well as column and row interchangeability in matrix models. It also stressed the benefits of traditional reformulations such as aggregation and projection to strengthen symmetry detection. The potential of this approach was demonstrated on two non-trivial applications.

It is important to stress that symmetries can be discovered *fully automatically* in this way. While a human modeller is usually aware of *some* symmetries in a model, such as full column- or row-interchangeability of a matrix CSP, the discovery of more, if not all, symmetries can be tedious and error-prone, especially for *piecewise* interchangeabilities. The latter usually change from one problem instance to the next, so it is safer and faster to let the system discover symmetries. Another strong motivation for the automatic discovery of symmetries is that dynamic symmetries can be discovered as well, using projection operators, as outlined in Section 8.2.

In practice, an implementation of the ideas in this paper would feature a database with the value and variable interchangeability results of each global constraint and global function. In fact, there is not even a need to consider only global constraints and global functions. For example, the CSP $\mathcal{P} = \langle V, D, v_1 < v_2 \rangle$, where $V \supset \{v_1, v_2\}$, is piecewise variable-interchangeable over the partition $\{\{v_1\}, \{v_2\}, V \setminus \{v_1, v_2\}\}$ of V, by application of the variable-interchangeability counterparts of Definition 9 and Propositions 4 and 6. Another example is the piecewise value-interchangeability of constraint C_2 in Example 10. Work in these directions has begun [7]. Such a system will only be limited by the strength of its reformulation operators, as too weak such operators prevent all the (static or dynamic) symmetries from being discovered or lead to insufficiently strong partitions upon piecewise interchangeabilities.

The work of [24] is closely related to ours: (piecewise) *variable* interchangeabilities are discovered compositionally from the model, using the intrinsic interchangeabilities of the constraints rather than their extensional definitions. It is observed that this method works particularly well in the presence of global constraints, as the sets of a variable partition are then likely to have more than two elements, unlike with binary constraints. Our work extends this work by also considering (piecewise) *value* interchangeabilities, (global) functions, and constraint *optimisation* problems, as well as by making the composition results more precise and by presenting reformulation operators. The notion of symmetrical constraint [19] corresponds to our notion of a constraint with (non-piecewise) *variable* interchangeability. Dynamic symmetries have been considered in the context of neighbourhood interchangeability [6, 17]; further investigations have been within the planning domain [12] and in [15].

It is also interesting to relate this research to the automatic modeling project of [2], which uses compositional refinement to transform abstract specifications into constraint programs. Since these transformations may introduce symmetries, [2] proposes to annotate the refinement rules with the symmetries so that they can be broken subsequently. Our *bottom-up* derivation approach is entirely orthogonal to their *top-down* refinement approach: It could in fact be applied as a first step to deduce properties of models before refinement. Both works also address the need for more automation for non-experts, a feature which is currently lacking in constraint programming when compared to MIP technology.

Acknowledgements

All authors are partly supported by institutional grant IG2001-67 of STINT, the Swedish Foundation for International Cooperation in Research and Higher Education. Pascal Van Hentenryck is partly supported by NSF ITR Award DMI-0121495. We thank the referees for their useful comments, especially for pointing out [24].

References

1. Backofen, R., and Will, S. Excluding symmetries in constraint-based search. In *Proceedings of CP'99*. LNCS 1713:73–87. Springer-Verlag, 1999.
2. Bakewell, A.; Frisch, A.M.; and Miguel, I. Towards automatic modelling of constraint satisfaction problems: A system based on compositional refinement. In *Proceedings of CP'03 Workshop on Modelling and Reformulating Constraint Satisfaction Problems*. Available at `http://www-users.cs.york.ac.uk/~frisch/Reformulation/03/`, 2003.
3. Barnier, N., and Brisset, P. Solving the Kirkman's schoolgirl problem in a few seconds. In *Proceedings of CP'02*. LNCS 2470:477–491. Springer-Verlag, 2002.
4. Beldiceanu, N., and Contejean, E. Introducing global constraints in CHIP. *Journal of Mathematical and Computer Modelling* 20(12):97–123, 1994.
5. Choueiry, B.Y., and Noubir, G. On the computation of local interchangeability in discrete constraint satisfaction problems. In *Proceedings of AAAI'98*, pages 326–333, 1998.

6. Choueiry, B.Y., and Davis, A.M. Dynamic bundling: Less effort for more solutions. In *Proceedings of SARA'02*. LNCS 2371:64–82, Springer-Verlag, 2002.
7. Eriksson, M. Detecting Symmetries in Relational Models of CSPs. Master's thesis, Computing Science, Department of Information Technology, Uppsala University, Sweden, 2005.
8. Fahle, T.; Schamberger, S.; and Sellmann, M. Symmetry breaking. In *Proceedings of CP'01*. LNCS 2293:93–107. Springer-Verlag, 2001.
9. Flener, P.; Frisch, A.M.; Hnich, B.; Kızıltan, Z.; Miguel, I.; Pearson, J.; and Walsh, T. Breaking row and column symmetries in matrix models. In *Proceedings of CP'02*. LNCS 2470:462–476. Springer-Verlag, 2002.
10. Focacci, F., and Milano, M. Global cut framework for removing symmetries. In *Proceedings of CP'01*. LNCS 2293:77–92. Springer-Verlag, 2001.
11. Focacci, F.; Lodi, A.; and Milano, M. Optimization-oriented global constraints. *Constraints* 7(3-4):351–365, 2002.
12. Fox, M., and Long, D. Extending the exploitation of symmetries in planning. In *Proceedings of AIPS'02*, 2002.
13. Freuder, E.C. Eliminating interchangeable values in constraint satisfaction problems. In *Proceedings of AAAI'91*, pages 227–233, 1991.
14. Gent, I.P., and Smith, B.M. Symmetry breaking during search in constraint programming. In *Proceedings of ECAI'00*, pages 599–603, 2000.
15. Gent, I.P.; McDonald, I.; Miguel, I.; and Smith, B.M. Approaches to conditional symmetry breaking In *Proceedings of SymCon'04*, 2004.
16. Johnson, E.; Nemhauser, G.; and Savelsbergh, M. Progress in linear programming-based algorithms for integer programming: An exposition. *INFORMS Journal on Computing* 12:2–23, 2000.
17. Lal, A., and Choueiry, B.Y. Dynamic detection and exploitation of value symmetries for non-binary finite CSPs. In *Proceedings of SymCon'03*, 2003.
18. Michel, L., and Van Hentenryck, P. A constraint-based architecture for local search. In *Proceedings of OOPSLA'02, ACM SIGPLAN Notices* 37(11):101–110, 2002.
19. Puget, J.-F. On the satisfiability of symmetrical constrained satisfaction problems. In *Proceedings of ISMIS'93*. LNAI 689:350–361. Springer-Verlag, 1993.
20. Puget, J.-F. Symmetry breaking revisited. In *Proceedings of CP'02*. LNCS 2470:446–461. Springer-Verlag, 2002.
21. Puget, J.-F. Symmetry breaking using stabilizers. In *Proceedings of CP'03*. LNCS 2833:585–599. Springer-Verlag, 2003.
22. Régin, J.-C. A filtering algorithm for constraints of difference in CSPs. In *Proceedings of AAAI-94*. 1994.
23. Régin, J.-C. Arc consistency for global cardinality constraints with costs. In *Proceedings of CP'99*. LNCS. Springer-Verlag, 1999.
24. Roy, P., and Pachet, F. Using symmetry of global constraints to speed up the resolution of constraint satisfaction problems. In *Proceedings of the ECAI'98 Workshop on Non-Binary Constraints*, pages 27–33, 1998.
25. Van Hentenryck, P.; Flener, P.; Pearson, J.; and Ågren, M. Tractable symmetry breaking for CSPs with interchangeable values. In *Proceedings of IJCAI'03*, pages 277–282. Morgan Kaufmann, 2003.
26. Van Hentenryck, P. Constraint and integer programming in OPL. *INFORMS Journal on Computing* 14(4):345–372, 2002.