# Reducing Symmetry in a Combinatorial Design Problem

**Barbara M. Smith**
School of Computing & Mathematics
University of Huddersfield
Huddersfield HD1 3DH, U.K. *
e-mail: b.m.smith@hud.ac.uk

## Abstract

A combinatorial design problem is considered which can be modelled as a constraint satisfaction problem in several different ways. The models all have a large number of symmetries which cause difficulties when searching for solutions. Different approaches to reducing the symmetry are discussed: remodelling the problem; adding constraints to the model at the outset; and adding constraints during search to prevent symmetric assignments being explored on backtracking. The most successful strategy for the problem of this paper employs a complex model with less inherent symmetry than the others, combined with symmetry breaking during search.

## 1 Introduction

Symmetry occurring in the problem being solved causes considerable difficulties for complete search algorithms, such as those commonly used to solve constraint satisfaction problems (CSPs). Symmetries divide the set of possible assignments into equivalence classes. Each class contains either only solutions or no solutions. If the search algorithm has proved that a partial assignment will not lead to a solution, it is a waste of effort to prove that every symmetric assignment will also not lead to a solution. If the algorithm finds a solution but all solutions are required, it is again a waste of effort to find all the symmetric solutions: the other solutions in the equivalence class can be retrieved using the symmetries, if required. The aim in dealing with symmetry is therefore to ensure that whenever a partial assignment is shown to be inconsistent, no symmetrical assignment is ever tried, and that if we are finding all solutions we only find distinct solutions, i.e. never two from the same equivalence class. The symmetry may be inherent in the problem being solved. Alternatively, it may arise as a result of modelling the problem, whether as a CSP or using some other representation, typically when objects that are in reality indistinguishable are modelled by separate entities.

In constraint programming there are three main ways of eliminating, or at least reducing, the adverse effects of symmetry:

- remodelling the problem, to give a model with less symmetry;
- adding constraints to the model which will only be satisfied by one assignment in each equivalence class;
- adding constraints during search to ensure that any assignment symmetric to one already considered will not be explored.

The first two approaches are relatively well-known, whereas the third is a recent development [1; 4; 5].

In this paper, a combinatorial design problem is discussed in which symmetry breaking seems crucial to successful solution using constraint programming. All three approaches listed earlier are shown to be useful in this case.

## 2 Problem Description

Problem 10 in CSPLib (at www.csplib.org), proposed by Warwick Harvey, is:

*32 golfers want to play in 8 groups of 4 each week, in such a way that any two golfers play in the same group at most once. How many weeks can they do this for?*

It is easy to show that they cannot play for more than 10 weeks, because by then each golfer will have been in the same group as 30 other golfers, leaving only 1 that they have not yet played. A solution for 9 weeks, found using constraint programming, is given in CSPLib.

This problem can be generalised to different sizes and numbers of groups. One instance is Kirkman's schoolgirls problem, first published in 1850 in the *Lady's and Gentleman's Diary* and solved in the same year. The statement given in [6] is: .. *a schoolmistress was in the habit of taking her girls for a daily walk. The girls were fifteen in number, and were arranged in five rows of three each, so that each girl might have two companions. The problem is to dispose them so that for seven consecutive*

*Most of this work was completed while the author was employed in the School of Computing at the University of Leeds.

*days no girl will walk with any of her school-fellows in any triplet more than once.* The corresponding problems for 9 girls for 4 days and 27 girls for 13 days (still in groups of 3) were also solved in 1850.

From the combinatorial point of view, Kirkman's problem and its variants (including variants with group size different from 3) can be viewed as a special case of the golfers problem in which every player meets every other player exactly once.

There is evidently a lot of symmetry in the problem. In all the models described here, the players are in effect labelled as separate entities, and it is hard to see how this could be avoided. The players can then be rearranged in any solution to give another, equally valid. In the following sections, different ways of modelling the problem as a CSP are presented, giving varying degrees of symmetry. The results of combining these models with symmetry-breaking constraints added at the start and/or with SBDS are discussed.

## 3 A Model using Set Variables

The Eclipse example program in CSPLib uses a set variable (a variable whose value is a set) to represent each group of golfers in each week. In principle, this is a good idea, since the order of the players within a group is immaterial. Representing the groups as arrays or lists instead, and thereby imposing an arbitrary order, would introduce further symmetry.

Suppose that in general we have $n$ players, to be arranged in $g$ groups of $s$ players in each week, playing for $w$ weeks. The groups can be represented as an array of arrays of set variables: `Group[k][i]` is the $i^{th}$ group in week $k$. The constraints are then that:

- the cardinality of each set is $s$;
- the sets in any week do not overlap, i.e. for all $k$, the sets `Group[k][i]`, $i = 0, ..., g-1$ have an empty intersection;
- any two sets in different weeks have at most one member in common.

We can include further constraints to do some simple symmetry breaking: the groups in the first week can be specified, as say $\{1, 2, .., s\}$, $\{s+1, .., 2s\}$, $\{2s+1, ..., 3s\}$,...... In addition, after the first week, players 1, 2, .. $s$ must be in different groups, so we can specify that these are the 1st, 2nd, ... $s$th groups.

This model has been implemented in ILOG Solver and run on a 166MHz Pentium PC. Using simple lexicographic variable ordering, a solution can be found with this model for 4 weeks for 8 groups of 4 golfers, but a solution was not found for 5 weeks overnight. On the other hand, the Solver program proved that there is no solution for 4 groups of 3 people for 5 weeks (in 5800 sec.), showing that in some instances, a complete search is feasible with this model.

A drawback with this model is that it is not easy to add more symmetry-breaking constraints. The weeks after the first are interchangeable in any solution, and it should in principle be possible to add constraints to distinguish between the weeks. For instance, we could insist that the smallest player number in week $k$ which appears in the same group as player 1 is smaller than the corresponding number in week $k+1$, for $k = 1, 2, ..., w-1$. It does not seem possible to express these constraints in Solver in terms of the set variables of this model. Furthermore, SBDS is currently implemented in Solver only for constrained integer variables (i.e. variables whose possible values are integers).

## 4 A Model Using Integer Variables

An alternative model using integer variables was developed, partly to allow SBDS to be used. However, the model also has less intrinsic symmetry.

In this model, the basic variables form an array `playInWeek` of integer-valued variables. The index of this array is an integer ranging over all possible pairs of players, where pair 0 is players 1 and 2, pair 1 is players 1 and 3, and the final pair is players $n-1$ and $n$. The value assigned to `playInWeek[pair(i,j)]`, where `pair(i,j)` returns the index for the pair consisting of players $i$ and $j$, indicates the week that this pair meet, with an extra dummy week to allow for players who never play each other. (In the 8 groups of 4 for 9 weeks problem, for instance, for each player there are 4 other players that he/she never plays, whereas in the schoolgirls problem, every girl walks with every other girl exactly once.)

As already mentioned, this model has less symmetry than the set model. We still do not have symmetry between the members of a group, and now we have also lost the symmetry between the groups in a week. In fact the groups are no longer explicitly represented. However, this also means that some of the constraints are now very cumbersome to represent. In particular, we need a large number of constraints to ensure transitivity: if $i, j$ play together in week $k$ and $j, l$ play together in week $k$, then $i, l$ play together in week $k$. We also need constraints to ensure that each person plays $s-1$ other people each week; this is straightforward, but we need such a constraint for every person in every week, rather than for each group in each week as before. On the other hand, we no longer need the constraints to ensure that each pair of players are assigned to the same group at most once: this is implicit in the fact that we have a variable for each pair, which must be assigned exactly one value (including the dummy week).

It is possible in this model, though not straightforward, to add the symmetry-breaking constraints described earlier to distinguish between the weeks after the first. i.e. if $j$ is the smallest value for which `playInWeek[pair(1,j)]` $= k$ and $l$ is the smallest value for which `playInWeek[pair(1,l)]` $= k+1$, then $j < l$. Constraints specifying the first week can also be added as in the first model, but since there is no longer a 1st group, 2nd group, etc. there is no equivalent to (and no need for) the constraints which say that after the first week, players 1, 2, .. , $s$ are in the 1st, 2nd, .. $s$th group respectively.

This model can prove that 4 groups of 3 people cannot play for 5 weeks in 570 sec., 10 times faster than the set model. However, the huge number of transitivity constraints means that the memory requirements are very large and increase very fast. It is not possible (on the PC used for these experiments) to run the 8 groups of 4 golfers for 9 weeks problem, let alone solve it.

## 5 Linking Set and Integer Variables

The main difficulty with the second model is the need for the transitivity constraints. These are not needed in the set-based model, because all the players who play together in a week are automatically in the same set, but in the second model we have no explicit representation of a group of players. The need for transitivity constraints in the second model can be avoided by introducing set variables to represent the groups, alongside the integer `playInWeek` variables, and then linking the two sets of variables together. In this combined model `PlayWith[i][k]` is defined to be the set of players that play with player $i$ in week $k$ (including him/herself). Note that these set variables are different from those in the first model. There are more of them (one for each player in each week, instead of one for each group in each week) but we have thereby avoided introducing symmetry between the groups.

The constraints to ensure that the size of each group is $s$ can be defined in terms of either the `playInWeek` variables or the `PlayWith` variables. We no longer need any transitivity constraints, and as in the model with just the `playInWeek` variables, we do not need constraints to ensure that each pair meets at most once.

We do, however, need to include constraints to link the two sets of variables: such constraints are called *channelling constraints* by Cheng, Choi, Lee and Wu [2]. Suitable channelling constraints in this case are:

- if pair $(i, j)$ play together in week $k$, then `PlayWith[i][k] = PlayWith[j][k]`
- if pair $(i, j)$ do not play together in week $k$, then `PlayWith[i][k]` and `PlayWith[j][k]` have an empty intersection

As in the previous model, the `playInWeek` variables will still be used as the search variables. Hence, the channelling constraints can define the effect on the `PlayWith` variables of assignments to the `playInWeek` variables, rather than v.v.

The set-based Eclipse formulation of this problem incorporates implied constraints due to Warwick Harvey: 'for each group of golfers in each week, for each other week, the number of groups in the given other week which share a player with the given group must be equal to the number of players per group.'

It would not be easy to express this idea using the integer variable model described in Section 4. However, it can be done in the model combining set and integer variables. There we have a set variable for each player in each week, and the constraint can be expressed as: any group, say `PlayWith[i][k]`, has an empty intersection with exactly $n - s^2$ of the set variables representing the groups in any other week (where there are $n$ golfers and the group size is $s$).

## 6 Results for Set and Integer Models

Table 6 shows the results for all three models on one set of problems, with 4 groups of 3 golfers. The golfers can play for 4 weeks, but there is no solution for 5 weeks. The times are given in seconds on a 166MHz Pentium PC; the number of fails is the number of times the search reaches a dead-end and has to backtrack.

The three models give similar performance in finding the first solution for 2, 3 or 4 weeks. However, proving that there is no solution for 5 weeks requires a complete search. The additional symmetry in the set-based model compared with the other two (the groups in each week are explicitly represented and there are no constraints to distinguish between the weeks) is reflected in the fact thqt this model has to do much more search to prove insolubility.

As previously observed [2; 7], adding a second set of variables and channelling constraints to give the combined model improves constraint propagation, even though the main reason for introducing them in this case was to make the constraints easier to express.

The final column shows the effect of adding implied constraints to the model with both integer and set variables. Unfortunately, there is a significant overhead in applying these constraints, and the results show an increase in running time even though the number of fails is greatly reduced.

The only symmetry breaking in the programs used for these results is that described in Section 4, i.e. the first week's groups are specified, and the weeks are distinguished by adding the constraint that the smallest player number appearing in player 1's group is smaller in week $k$ than in week $k+1$. To reduce the remaining symmetry, Symmetry Breaking During Search (SBDS) can be used, and this is first described briefly, before considering its application in this case.

## 7 Symmetry Breaking During Search

In this section, the SBDS method, described fully in [4; 5], is outlined. The key property of a symmetry is that it is solution-preserving: that is, for any full assignment $A$ and any symmetry $g$ of the problem, $g(A)$ is a solution iff $A$ is. We also assume that a symmetry $g$ acts piecewise. Formally, if we extend a partial assignment $A$ to $A + (var = val)$, then $g(A + (var = val)) = g(A) + g(var = val)$. We also write $A$ for the constraint that all assignments in $A$ hold, and $g(A)$ for the constraint that the symmetric versions of these assignments hold.

At choice points during search, suppose that the alternative branches are created by the constraints $var = val$ and $var \neq val$. On backtracking to this point, having

| No. of weeks | Set model | | Integer model | | Combined model | | Combined model with implied constraints | |
|---|---|---|---|---|---|---|---|---|
| | Fails | time | Fails | time | Fails | time | Fails | time |
| 2 | 22 | 0.03 | 16 | 0.11 | 16 | 0.11 | 4 | 0.13 |
| 3 | 36 | 0.04 | 107 | 0.27 | 95 | 0.26 | 6 | 0.26 |
| 4 | 91 | 0.10 | 91 | 0.57 | 89 | 0.51 | 15 | 0.58 |
| 5 | 2744282 | 5800 | 72544 | 570 | 51208 | 350 | 18874 | 1160 |

Table 1: Comparison of three models on problems with 4 groups of 3 players

explored the $var = val$ branch, the search will then explore the alternative branch, $var \neq val$. We also add to this branch the constraint $g(A) \Rightarrow g(var \neq val)$ for each symmetry $g$. This can be interpreted as: if the symmetric equivalent of the assignments in $A$ hold, then $g(var \neq val)$ must also hold along this branch. Since $g(A)$ involves all search variables set so far, it is potentially large. However, we note that if we extend $A$ to the assignment $A^+ = A + \texttt{vars}[\texttt{i}] = \texttt{j}$, then $g(A^+) = g(A) + g(\texttt{vars}[\texttt{i}] = \texttt{j})$. We construct a new boolean variable for each symmetry $g$ representing whether $g(A)$ is satisfied or not. The value of this variable for $g(A^+)$ is the conjunction of its value for $g(A)$ and $g(\texttt{vars}[\texttt{i}] = \texttt{j})$. Hence, we can compute $g(A)$ incrementally. These boolean variables have a further advantage. When one is proven to be false, i.e. the value true is removed from its domain, we know that $g$, the corresponding symmetry, is permanently broken on this branch. This makes it safe to discard $g$ from further consideration on the branch.

To show how this works in the 8-queens problem, suppose that the first assignment is $v_1 = 2$, i.e. the queen on the first row of the board is placed in column 2. There are 7 symmetries in the $n$-queens problem: rotations through 90°, 180° and 270° respectively, reflections in the horizontal and vertical axes, reflections in the main diagonals. The effect of these symmetries on $v_1 = 2$ are $v_2 = 8$, $v_8 = 7$, $v_7 = 1$, $v_8 = 2$, $v_1 = 7$, $v_2 = 1$ and $v_7 = 8$, respectively. The last four of these are inconsistent with $v_1 = 2$, so that we immediately know that only the rotational symmetries are not broken by this assignment, and hence the others need not be considered further on this branch.

If the second assignment made is $v_2 = 4$, on backtracking the right branch is created, with the constraint $v_2 \neq 4$. We add the equivalent of $g(A) \Rightarrow g(var \neq val)$ for each remaining symmetry: for instance, for the rotation through 90° we would have $v_2 = 8 \Rightarrow v_4 \neq 7$.

To implement SBDS in ILOG Solver, we provide a new search function `IlcGenerateSym` to replace Solver's `IlcGenerate`. At present, we assume that all branching points are of the form `vars[i] = j` for some array index `i` and value `j` (although the implementation could be adapted for other branching strategies). SBDS can be used with minimal overhead in cpu-time and minimal effort from the programmer, who simply writes one function for each symmetry in the problem. The function for the symmetry $g$ takes three arguments, representing `vars`, `i` and `j` and returns a constraint representing $g(\texttt{vars}[\texttt{i}] = \texttt{j})$. An array of these functions is passed to `IlcGenerateSym`. For instance, the function for the rotation through 90° in the $n$-queens problem is:

```
IlcConstraint r90 (IlcIntVarArray vars,
                   IlcInt i, IlcInt j)
   {return vars[j] == nQueen-1-i;}
```

There are two main advantages of using SBDS as well as, or instead of, adding constraints to the model. First, it is more comprehensive: if we can write a function describing the action of a symmetry, we can eliminate that symmetry, whereas we might not be able to devise a constraint to add to the model which will have the same effect. Secondly, SBDS does not pre-empt the search strategy: whatever order variables and values are assigned in, SBDS will simply prevent symmetric assignments being considered on backtracking. Adding constraints to the model, on the other hand, can conflict with the search strategy. For instance, the constraints ordering the weeks assume that small player numbers will be assigned first to the groups containing player 1, and might hinder rather than assist in finding a solutions if the search strategy were different.

## 8 SBDS in the Golfers Problem

Given the model of the golfers problem described in Section 5, the symmetries are that we can permute the labels of the golfers and we can permute the weeks, or both, to get another solution. Hence, if there are $n$ golfers playing for $w$ weeks, any solution will give up to $n! \times w!$ symmetric solutions. We could not contemplate dealing with so many symmetries by SBDS alone, so adding constraints to the model to break some of the symmetry still might be a good idea.

We can eliminate some symmetry by fixing the groups in the first week, as described earlier. Given a solution in which the first week is specified, we can no longer, for instance, interchange players 1 and $s + 1$ in the rest of the solution and get another solution, as we could do before. However, we can still permute the golfers in any way which leaves the groups in the first week unchanged. For instance, we can permute players 1, 2, .. , $s$; or we can interchange 1, 2, .., $s$ with $s + 1, .., 2s$ in some way. We can also still permute the weeks after the first. Furthermore, and far harder to deal with, any combination of these changes can be made; we could for instance permute players 1, 2, ..., $s$ and simultaneously players $s+1,..,2s$.

Having specified the first week groups, we can now think about adding functions describing at least some of the remaining symmetries; a useful feature of symmetry breaking is that it is not necessary to eliminate the symmetry completely. If we describe only some of the symmetries, we will get correct solutions, and we are likely to get a reduction in wasted search. However, it cannot be guaranteed that only non-symmetric solutions will be found unless all the symmetries are broken.

Since it is clearly impossible to describe all the symmetries remaining after the first week groups have been specified, the choice made is somewhat arbitrary. It seems intuitively plausible that the simpler symmetries might give highest returns, as well as being easiest to describe.

The simpler of the remaining symmetries are those within the first week groups, e.g. the permutations of 1, 2, .. , $s$. The ordering constraints described earlier, which distinguish between the weeks after the first, involve player 1, so that if those constraints were kept, it would not be possible to interchange player 1 with any other player throughout a solution and guarantee to get a new solution. It seemed preferable to drop the ordering constraints and instead to include player 1 in the SBDS symmetry functions. The within-group symmetries chosen are the pair-wise transpositions, e.g. 1 with 2. For a problem with $k$ groups of size $s$, this gives $k \times s(s-1)/2$ symmetry functions. For the symmetries between two first week groups, the pairwise interchanges between the first group and a permutation of the second group are described. For instance, one such symmetry, for groups of size 3, interchanges 1 with 4, 2 with 6, 3 with 5. For a problem with 7 groups of 4 golfers, this gives 546 symmetry functions in all: 42 for the symmetries within groups and the rest for those between groups. Since SBDS has previously been used with more than 1000 symmetry functions, and in that case resulted in a 40-fold reduction in cpu time [3], the number of functions in this case is well within the limits of the method. Both sets of symmetry functions are output by a program written for the purpose.

## 9 Results of Adding SBDS

The tables below show the effect of using SBDS rather than just adding symmetry constraints to the model. They also show again the effect of the implied constraints. When SBDS is not used, the ordering constraints on the weeks after the first are used; in both cases, the groups in the first week are specified.

It is clear from the results that SBDS can speed up the search for a solution considerably; the most dramatic effect is in proving that 4 groups of 3 golfers cannot play for 5 weeks. Moreover, the overhead incurred in adding the symmetry-breaking functions appears to be small. As noted previously, the implied constraints reduce the search space enormously in some cases, but often the running time increases. The equivalent of Kirkman's schoolgirls problem (5 groups of 3 golfers for 7 weeks) is still intractable, however.

Although the SBDS functions are clearly beneficial in reducing search, it would be useful to know how much of the problem symmetry remains. Using SBDS, 42 solutions were found for the 4 groups of 3 for 4 weeks problem. Only three of these are unique[1]. Often the transformation of one solution into another is complex, involving several permutations within and between the first weeks groups, as well as a permutation of the weeks. It is difficult to see how all such symmetries could be eliminated by describing them directly for SBDS.

It is worth pointing out two things about these solutions. First, every solution has $\{1, 4, 7\}$ as a group in the second week. To add more symmetry-breaking constraints to the model, rather than using SBDS, we could specify that the smallest numbers from the first $s$ groups in the first week must form a group in the second week, giving say $\{1, 5, 9, 13\}$ with groups of size 4. If such a constraint is added, there are 1200 solutions to the 4 groups of 3 for 4 weeks problem, rather than the 42 found using SBDS. This shows that the limited SBDS functions used here do achieve a considerable reduction in symmetry compared with constraints added to the model, even though they do not eliminate symmetry entirely.

Second, even though the ordering constraints on the weeks were dropped in order to use SBDS, the solutions found do have the weeks ordered in the same way. This is a consequence of SBDS and the search strategy. The players in the same group as player 1 are decided first, and the value ordering will choose first the player with the smallest number. On backtracking, the symmetry functions prevent any larger number being considered.

## 10 A Model without Weeks

The fourth model of the problem to be discussed has less inherent symmetry than those previously considered, and was designed with that aim in mind.

The original motivation for the model described in Section 4 was to represent the problem using integer variables so that SBDS could be used. However, a beneficial side-effect was that the explicit representation of the groups disappeared, and along with it the symmetry between the groups in each week which occurs in the set-based model of Section 3. The remaining symmetries are those between the weeks and those between the players. It seems impossible to model the problem without labelling the players, but can we eliminate the symmetries between weeks by remodelling the problem so that there is no explicit representation of the weeks?

To achieve this we need to have some way of defining a group and the groups playing at the same time, without using a week number or a group number within a week. A possible way to do this is to focus on pairs of players: each pair appears in at most one group; and each week

[1] I am grateful to Warwick Harvey for this information.

| No. of weeks | 2 | | 3 | | 4 | | 5 | |
|---|---|---|---|---|---|---|---|---|
| | Fails | time | Fails | time | Fails | time | Fails | time |
| Symmetry constraints | 16 | 0.11 | 95 | 0.87 | 89 | 0.61 | 51208 | 350 |
| Symmetry constraints + implied constraints | 4 | 0.13 | 6 | 0.26 | 15 | 0.58 | 18874 | 1160 |
| SBDS | 7 | 0.1 | 39 | 0.20 | 69 | 0.42 | 821 | 5.62 |
| SBDS + implied constraints | 3 | 0.14 | 6 | 0.26 | 13 | 0.59 | 318 | 12.5 |

Table 2: Problems with 4 groups of 3 golfers

| No. of weeks | 2 | | 3 | | 4 | | 5 | | 6 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Fails | time | Fails | time | Fails | time | Fails | time | Fails | time |
| Symmetry constraints | 52 | 0.22 | 430 | 1.74 | 821 | 4.66 | 963 | 6.58 | 48141 | 278 |
| Symmetry constraints + implied constraints | 9 | 0.25 | 20 | 0.51 | 35 | 1.52 | 170 | 6.38 | 2222 | 94.3 |
| SBDS | 9 | 0.18 | 160 | 0.87 | 391 | 2.68 | 898 | 6.66 | 24507 | 125 |
| SBDS + implied constraints | 5 | 0.33 | 19 | 0.66 | 30 | 1.69 | 170 | 7.5 | 1360 | 59.1 |

Table 3: Problems with 5 groups of 3 golfers

of play, and the groups in that week, can be implicitly described by the pairs of players playing in that week.

However, modelling the general golfers problem in this way is complicated by the fact that some pairs may never play together.

The special case where every player plays every other player exactly once, i.e. Kirkman's schoolgirls problem and its variants, is easier to model in this way than the general golfers' problem. However, the model described below has been adapted for the general case as well.

The proposed model has an array of set variables, one for each pair of players, such that the value assigned to `GroupWithPair[pair(i,j)]` will be the set of pairs of players that form a group with players $i$ and $j$: the pair $(i,j)$ is itself a member of the set. `PairsInSameWeek[pair(i,j)]` contains all the pairs that play in the same week as pair $(i,j)$. The cardinalities of these sets are constrained to be $s(s-1)/2$ and $gs(s-1)/2$ respectively; and `GroupWithPair[pair(i,j)]` is a subset of `PairsInSameWeek[pair(i,j)]`.

The model also has an array of 0-1 variables, one for each pair of pairs of players: the value of the `playInSameWeek` variable is 1 if the corresponding pair of pairs play in the same week, and 0 otherwise. These variables are used as the search variables.

As before, channelling constraints are needed to link the integer variables with the set variables:

- if pairs `p1` and `p2` play in the same week:
  - if the pairs have a player in common, then `GroupWithPair[p1] = GroupWithPair[p2]`
  - `PairsInSameWeek[p1] = PairsInSameWeek[p2]`
- if pairs `p1` and `p2` do not play in the same week:
  - if the pairs have a player in common, then the intersection of `GroupWithPair[p1]` and `GroupWithPair[p2]` is empty
  - the intersection of `PairsInSameWeek[p1]` and `PairsInSameWeek[p2]` is empty

The weeks are never explicitly mentioned in this model: for instance, there is nothing explicitly stating that there must be exactly $w$ different values assigned to the `PairsInSameWeek` variables, representing the play in each week. This requirement is implicit in the other constraints, and the play in each week can be reconstructed from the `PairsInSameWeek` array when a solution has been found.

Apart from the fact that the symmetry between the weeks has been removed, this might be thought to be a rather poor way of modelling the problem. It has a very large number of variables; the problem with 4 groups of 4 golfers for 5 weeks, for instance has 7140 search variables, as well as all the set variables. Nevertheless, it has given good results, as will be seen.

## 11 Results

This is the first model described in this paper which can solve the original Kirkman's schoolgirls problem without an inordinate amount of search: without adding any SBDS functions, the first solution is found after 5685 fails. However, the unwieldiness of the model is shown by the fact that this takes 770 seconds (on a 600MHz PC).

The remaining symmetry in the model is that the players can be permuted. To deal with these symmetries using SBDS, functions describing all transpositions of a pair of players were first described. This makes little difference to the effort required to find a first solution to the 5 groups of 3 for 7 weeks problem: the number of fails reduces to 5586. However, only two solutions are found to the 4 groups of 4 for 5 weeks problem, compared to 347 solutions to this problem previously, using SBDS with the combined model of Section 5. Unfortunately, the two solutions are symmetric to each other: one can be transformed into the other by interchanging 9 with 13, 10 with 14, 11 with 15, 12 with 16.[2]

The `playInSameWeek` variables are assigned in lexicographic order, with the value 1 being assigned before the value 0. This means that the first solution found has the groups $\{1, 2, .., s\}$, $\{s+1, .., 2s\}$, ... playing in the first week (or rather the first set of groups assigned). Furthermore, once transpositions of players are forbidden by SBDS, *every* solution found is like this. Hence, although these groups are not specified by adding constraints to the model as before, they still arise, given the search strategy. To reduce the remaining symmetry further, we only need consider symmetries which leave these groups unchanged. (For instance, the transformation of one of the solutions found for the 4 groups of 4 for 5 weeks problem into the other is such a symmetry.)

Next, functions describing the interchange of a permutation of a 'first-week' group with a permutation of another were described; this gave a total of 465 functions. It was then possible to find *all* solutions to Kirkman's schoolgirls problem. 312 solutions were found, whereas there are 7 unique solutions to this problem. This took 200K fails and 6.5 hours running time. A further set of functions were added, describing simultaneously exchanging permutations of two of the first-week groups and transposing two players within another first week group, giving a total of 3705 SBDS functions, the largest number used to that point. The number of solutions found was reduced to 126: this took 96K fails and 3.5 hours. There is some overhead involved in adding thousands of SBDS functions: the number of fails reduced by more than half, and the running time by less than half. Even so, the overhead is clearly worthwhile in this case.

It is remarkable that describing only a relatively small number of symmetries (compared to the 15! possible permutations of the players) reduces the number of solutions found by so much. However, reducing the number of solutions further, by adding more functions, is likely to require a greater understanding of which symmetries remain.

## 12 Conclusion

Although Kirkman's schoolgirls problem has been solved for 150 years, this problem, and the golfers problem as a generalisation of it, is very useful as a case study of symmetry for constraint programming. If multiple non-symmetric solutions are required for this or similar combinatorial design problems, complete search methods might indeed be useful.

The symmetry in the golfers problem causes severe difficulties for search algorithms; most of the approaches presented in this paper could not solve Kirkman's schoolgirls problem in any reasonable time. To tackle these problems successfully as CSPs, the symmetry has to be addressed. Several different models have been presented, with varying levels of symmetry. Designing models with less inherent symmetry has in this case lead to models which are more complex and have far more variables; however, the reduced symmetry makes the extra complexity worthwhile for some problems.

The remaining symmetry can be partially eliminated either by adding symmetry-breaking constraints to the model or by SBDS or both. It has been shown that using SBDS can reduce search effort, and the number of symmetric solutions found, compared with adding constraints. By adding SBDS functions describing selected symmetries to the final model described, it has been possible to find all solutions to Kirkman's schoolgirls problem. To the author's knowledge, this is the first time that all solutions to this problem have been found by modelling the problem as a CSP. Furthermore, the other models described in the paper have produced the first CSP-based solutions, or proofs of insolubility, for a number of other instances (see http://www.icparc.ic.ac.uk/~wh/golf/).

The original golfers' problem (with 8 groups of 4 players) is not suitable for the final model described, because of the very large number of variables generated. The best prospect for further progress with this problem seems to be the combined model, with both integer and set variables, if more SBDS functions can be added. The remaining symmetries in this model are more complex than with the final model, since the permutations of the players can be combined with permutations of the weeks, but the success of SBDS in the final model suggests that it would be worthwhile to try to eliminate more of the symmetry.

Future work is planned which will improve our understanding of how SBDS can be used, for instance by considering the algebra of the symmetry group. We hope to find ways of dealing with symmetries during search without having to specify each one individually, or to consider the interaction of the search strategy and the symmetries specified, to identify which other symmetries can still be active.

### Acknowledgment

Part of the work described here was done on a visit to IC-PARC, and I am very grateful to Warwick Harvey for helpful discussions on dealing with symmetry during that visit. I should especially like to thank him for identifying symmetric solutions. I also thank fellow-members of the APES group and in particular Ian Gent: the development of SBDS is joint work with him.

## References

[1] R. Backofen and S. Will. Excluding Symmetries in Constraint-Based Search. In J. Jaffar, editor, *Principles and Practice of Constraint Programming - CP'99*, LNCS 1713, pages 73–87. Springer, 1999.

[2] B. M. W. Cheng, K. M. F. Choi, J. H. M. Lee, and J. C. K. Wu. Increasing constraint propagation by

[2] This information is again due to Warwick Harvey.

redundant modeling: an experience report. *Constraints*, 4:167–192, 1999.

[3] I. Gent, S. Linton, and B. Smith. Symmetry breaking in the alien tiles puzzle. Report APES-22-2000, Oct. 2000. Available from http://www.dcs.st-and.ac.uk /~apes/apesreports.html.

[4] I. P. Gent and B. M. Smith. Symmetry Breaking During Search in Constraint Programming. Research Report 99.02, School of Computer Studies, University of Leeds, Jan. 1999.

[5] I. P. Gent and B. M. Smith. Symmetry Breaking During Search in Constraint Programming. In W. Horn, editor, *Proceedings ECAI'2000*, pages 599–603, 2000.

[6] W. W. Rouse Ball. *Mathematical Recreations and Essays.* Macmillan, 11th edition, 1938.

[7] B. M. Smith. Modelling a Permutation Problem. Research Report 2000.18, School of Computer Studies, University of Leeds, June 2000. Presented at the ECAI'2000 Workshop on Modelling and Solving Problems with Constraints.