

University of Leeds  
**SCHOOL OF COMPUTING**  
**RESEARCH REPORT SERIES**  
Report 2001.02

**Dual Models in Constraint Programming**

by

**Barbara M. Smith**

January 2001

## Abstract

In modelling a problem as a constraint satisfaction problem, it is often possible to find alternative models representing complementary views of the problem. It has been proposed that such dual models could be linked into a single model, with channelling constraints linking their variables. This can reduce search by allowing more constraint propagation than with either model separately. However, combining two complete models can increase overall running time because of the increased numbers of constraints and variables. We consider how much of the second model it may be necessary to include; in the particular case of permutation problems it is shown that additional constraint propagation is due to the channelling constraints themselves, and that the constraints of the dual model are not required. Combinations of models using integer variables and set variables are also considered. The derivation of dual models, including cases where more than two dual models exist, is discussed.

## 1 Introduction

In modelling and solving a problem as a constraint satisfaction problem (CSP) there are many issues to address, starting with the choice of problem entities to be represented by the variables and values of the CSP. It is often possible to find alternative models, based on different choices of variables and values, representing complementary views of the problem. Cheng, Choi, Lee and Wu [1] proposed the idea of ‘redundant modelling’ in such situations: models based on the different views of the problem should be combined into one. The variables of the two models are linked by *channelling* constraints. A similar idea was previously suggested, specifically for permutation problems, by Geelen [2]. The rationale given by Cheng *et al.* for linking two models in this way is that because of the linking constraints, an assignment to a variable from one set will trigger an assignment to a variable of the other set. The resulting constraint propagation may then be reflected back into domain reductions in the original model which would not otherwise have been found at that point. On the other hand, combining two models has the disadvantage of increasing the number of variables and constraints to be processed, so that increased domain pruning may not offset the extra overhead.

This paper discusses the specific case of permutation problems as well as other cases where an integer model is combined with a set model. The aim is to elucidate the behaviour of combined dual models and to derive guidelines for combining them efficiently.

## 2 Permutation Problems

A constraint satisfaction problem is a permutation problem if it has the same number of values as variables, all variables have the same domain and each variable must be assigned a different value. Any solution assigns a permutation of the values to the variables. There will be other constraints in the problem determining which permutations are acceptable solutions.

Each possible value is assigned to exactly one variable and each variable is assigned exactly one value. In principle a *dual* CSP can be derived in which

the roles of the variables and values are reversed (by analogy with primal and dual problems in linear programming, where the variables of the dual problem correspond to the constraints of the primal problem and v.v.). For instance, if there are  $n$  people to do  $n$  tasks, we can either assign tasks to people or people to tasks. The models can also be combined, and the constraint propagation resulting from combining dual models of a permutation problem is considered in this section.

## 2.1 The $n$ -queens Problem

The problem of placing  $n$  queens on an  $n \times n$  chessboard so that no two queens are in the same row, column or diagonal can be modelled as a permutation CSP. The variable  $r_i$ ,  $1 \leq i \leq n$ , represents the queen on row  $i$ ; the value assigned is the column on which this queen is placed. Because each queen must be in a different column, and all columns are occupied, the values assigned to  $r_1, r_2, \dots, r_n$  must be a permutation of  $1, 2, \dots, n$ . In this case, the constraints determining which permutations are acceptable solutions are that no two queens can be on the same diagonal. These constraints can be expressed as: allDifferent( $r_1 + 1, r_2 + 2, \dots, r_n + n$ ) and allDifferent( $r_1 - 1, r_2 - 2, \dots, r_n - n$ ), following the ILOG Solver User's Manual [5].

Table 1 shows the number of backtracks (fails) and approximate running time (on a dual processor 666MHz PIII) in finding all non-symmetric solutions to  $n$ -queens problems with  $8 \leq n \leq 15$ , using ILOG Solver. If finding just the first solution, changes to the variable ordering strategy, for instance, can result in a different solution being found, whereas finding all solutions allows a fairer comparison of different methods. The symmetry of the problem has been eliminated using SBDS (Symmetry Breaking During Search), described in [4], which has been implemented for ILOG Solver. SBDS adds constraints when the search backtracks to a choice point, so that in exploring the alternative choice, assignments are forbidden which are symmetric to assignments already considered. The functions required by SBDS, describing the effect of each of the seven symmetries of the  $n$ -queens problem on the assignment of a value to a variable, are given in full in [3]. The variable ordering used in the ILOG User's Manual [5] has been adopted: a variable with smallest domain is chosen, breaking ties by choosing the variable with smallest value.

$n$	binary		global	
	fails	sec.	fails	sec.
8	61	0.01	45	0.01
9	219	0.03	171	0.03
10	888	0.07	635	0.07
11	3829	0.29	2719	0.32
12	17940	1.43	12204	1.44
13	90512	7.04	58800	7.26
14	487948	38.4	309706	38.1
15	2804307	225	1718862	223

Table 1: No. of backtracks and running time to find all non-symmetric solutions to the  $n$ -queens problem, with either binary  $\neq$  constraints or a global allDifferent constraint.

In using an allDifferent constraint, there is a choice between treating it either as a collection of binary  $\neq$  constraints or as a global constraint, making it generalised arc consistent (GAC). The effect of these choices on the constraint that  $r_1, r_2, \dots, r_n$  are all different is shown in Table 1. (The diagonal constraints are treated as pairwise  $\neq$  constraints throughout.) In general, making non-binary constraints GAC is time-consuming and maintaining GAC during search is prohibitive, but for the allDifferent constraint there is a specialised algorithm by Régin [9] which can be cost-effective. In ILOG Solver, the level of consistency to be enforced on an allDifferent constraint (AC (binary) v. GAC (global)) is specified via an argument to the allDifferent constraint. In this case, treating the allDifferent constraint globally reduces the number of backtracks considerably, but because constraint propagation takes so much longer, there is little or no time saving overall.

The dual model has variables  $c_j, j = 1, 2, \dots, n$  representing the queen in column  $j$ . The constraints are identical to those of the primal model, with  $c_1, \dots, c_n$  substituted for  $r_1, \dots, r_n$ . There is thus no advantage in using the dual model *instead of* the primal model. However, it can be beneficial to use both models together.

As shown by Cheng *et al.*, the required channelling constraints here are  $r_i = j$  iff  $c_j = i$ , for all pairs  $1 \leq i, j \leq n$ . ILOG Solver provides an *inverse* constraint, `IlcInverse`, to express this relationship between two sets of constrained variables, represented as arrays. The constraint `IlcInverse(r, c)` leads to the same constraint propagation as the set of binary channelling constraints (and hence to the same number of fails in the search), but is otherwise more efficient.

As will be shown, in the  $n$ -queens problem the additional constraint propagation, if any, resulting from combining the two models is almost entirely due to the channelling constraints and not to the constraints of the second model. Furthermore, the inverse constraint is sufficient to ensure that the values assigned to the  $r_i$  variables must all be different, since each must correspond to a unique  $c_j$  variable. Hence the allDifferent constraint on the  $r_i$  variables can be dispensed with, unless GAC is enforced on it.

We can define a *minimal combined model* of a permutation problem: it has a complete set of constraints defining the problem in terms of one set of variables, but without the allDifferent constraint on these variables. Instead, the dual variables are defined and the primal and dual variables are constrained to be inverses of each other, which will ensure that any solution found is a permutation. This model minimizes the overhead of combining dual models of the problem, but retains the additional pruning given by the channelling constraints.

Table 2.1 shows the results of using a minimal combined model of the  $n$ -queens problem. Column (a) shows that the channelling constraints reduce search considerably in comparison with the binary  $\neq$  constraints, although not as much as a global allDifferent constraint on the primal variables.

A further choice is available when there are two sets of variables: the search variables can be either set, or both sets at once. The latter possibility was pointed out by Cheng *et al.* and is particularly interesting with a dynamic variable ordering strategy, such as that used here, since the search can then swap from one set of variables to the other, as the search progresses. In terms of the primal variables, the search can switch between assigning a value to the variable with smallest domain and assigning the value that appears in fewest

domains. Table 2.1, column (b), shows the effect of this, using the variable ordering strategy described earlier: it gives the smallest running time for the larger problems.

$n$	(a)		(b)	
	fails	sec.	fails	sec.
8	45	0.01	49	0.01
9	176	0.03	165	0.02
10	643	0.08	711	0.08
11	2758	0.33	2736	0.32
12	12391	1.53	12431	1.48
13	59833	7.51	58946	7.15
14	315914	40.1	307525	37.6
15	1755260	229	1670747	211

Table 2: No. of backtracks and running time to find all non-symmetric solutions to  $n$ -queens problems, using a minimal combined model, (a) using  $r_i$  variables as search variables, (b) using both  $r_i$  and  $c_j$  variables as search variables with dynamic variable ordering.

Cheng *et al.* proposed using all the constraints of both models. If this is done here, rather than using the minimal combined model, there is no reduction in search compared with columns (a) and (b) of Table 2.1, respectively, depending on whether the allDifferent (permutation) constraints are treated as binary constraints or as global constraints. In both cases, the number of fails reported by Solver is 2 fewer than in Table 2.1 when  $n = 12$  and  $n = 15$ : this is because in these two problems, the diagonal constraints of the dual model twice detect that an assignment is forbidden by the SBDS constraints, and it is not tried; without the diagonal constraints, at the same point in the search, the assignment is tried and immediately fails. The running time using both complete models is, of course, much longer than with the minimal complete model, since there are far more constraints to handle.

If the minimal combined model is used, with the addition of an allDifferent constraint on the  $r_i$  variables, treated as a global constraint, the number of fails is the same as in the second column of Table 1. The channelling constraints do not result in any further domain pruning than the allDifferent constraint. Furthermore, the running time is increased because of the additional variables and constraints. Hence, the minimal combined model should not be used with a global allDifferent constraint, since it merely increases the overhead to no purpose. The experiments also show that in this problem adding the constraints of the dual model, as well as the variables and channelling constraints, only increases the running time and does not reduce search.

## 2.2 What Do The Channelling Constraints Do?

Walsh [12] compares theoretically the levels of pruning achieved by the binary  $\neq$  constraints, the channelling constraints and the global allDifferent constraint under various consistency algorithms. Here an informal account is given of the different effects of these constraints, supposing a search algorithm that maintains arc consistency on binary constraints and generalized arc consistency on

allDifferent constraints. Solver’s search algorithm maintains AC, as do other constraint programming tools, and it is assumed that, as in Solver, the user can choose whether or not to maintain GAC on an allDifferent constraint.

The channelling constraints linking the dual representations of the  $n$ -queens problem, modelled as a permutation CSP, are in effect binary constraints, although input to Solver as a single inverse constraint. Hence, during search, Solver will ensure that these constraints remain arc consistent. It has been shown experimentally in the previous sections that channelling constraints achieve more search reduction than binary  $\neq$  constraints on the primal variables, but less than the global allDifferent constraint on those variables. Constraint propagation in the three cases will delete values in the domains of variables as follows:

- binary  $\neq$  constraints: if the domain of a primal variable is reduced to a single value, and the value is assigned (either directly as a result of the search or as the result of constraint propagation), that value will be removed from the domains of all the other original variables.
- channelling constraints: if  $r_i$  has only the value  $j$  remaining, and so is assigned the value  $j$ , then  $c_j$  will be assigned the value  $i$ , and the channelling constraints will ensure that no other variable  $r_k$  can be assigned the value  $j$ . Hence, these constraints will do as much propagation as the  $\neq$  constraints. In addition, since the constraints are symmetric in the primal and dual variables, if the domain of a *dual* variable is reduced to a single value, this value will be removed from the domains of all other dual variables. Thus, if a value of a primal variable appears in the domain of only one such variable, propagation of the channelling constraints detects that it must be assigned to that variable.
- global allDifferent constraint: GAC on the allDifferent constraint will delete any value which cannot be part of any assignment satisfying the constraint. In particular, if a value appears in the domain of only one variable in a permutation problem, it will delete all other values from the domain of that variable, just as the channelling constraints would do. However, GAC on an allDifferent constraint can do further deletions (e.g. if  $k$  of the variables have only  $k$  values between them, those values will be deleted from the domains of all other variables involved in the constraint).

Hence, as already shown, the constraint propagation done by AC on the channelling constraints is intermediate between that done by AC on the binary  $\neq$  constraints and GAC on the allDifferent constraint. Specifically, AC on the channelling constraints detects singleton *values*, i.e. values appearing in the domain of only one variable, as well as singleton *variables*.

### 2.3 Langford’s Problem

The  $n$ -queens problem, considered as a permutation CSP, is unusual in that the primal and dual models are identical, apart from the renaming of the variables. Often the problem constraints will be much easier to express in terms of one set of variables than the other, and then the advantages of combining dual models are somewhat different. Here such a problem is considered.

One instance of Langford’s problem [6] can be stated as follows:

*A 27-digit sequence includes the digits 1 to 9 three times each. There is just one digit between the first two 1s, and one digit between the last 2 1s. There are just two digits between the first two 2s and two digits between the last two 2s,.. and so on. Find all possible such sequences.*

For consistency with Miller’s notation, this instance will be referred to below as the (3,9) problem (the sequence has 9 different digits, each used 3 times). A solution is **181915267285296475384639743** and there are two others, together with the reverse of each sequence, because of the symmetry of the problem. There are 5 solutions, excluding reversals, to the (3,10) instance, using hexadecimal digits, say, but there are no solutions to the (3,11) or (3,12) problems.

Possible ways of modelling the problem as a permutation CSP are considered in greater detail in [11]. Here the focus is on dual models of the problem. One way of modelling the (3,9) instance as a CSP is to have 27 variables,  $p_1, p_2, \dots, p_{27}$  which might be thought of as corresponding to  $1_1, 1_2, 1_3, 2_1, \dots, 9_2, 9_3$ , where  $1_1$  represents the first 1 in the sequence,  $1_2$  is the second 1 and so on. The value assigned to a variable gives the position in the sequence of the corresponding digit. Any solution must assign a permutation of  $1, \dots, 27$  to the variables, so  $p_1, p_2, \dots, p_{27}$  must be all different. The constraints specifying which permutations are acceptable for this problem are *separation constraints* to ensure the correct spacing between the digits, for instance  $p_2 = p_1 + 2$ ,  $p_5 = p_4 + 3$ , and so on.

The symmetry can be eliminated by adding constraints; for instance, in the (3,9) problem the second 9 cannot be placed in the second half of the sequence, and if it is in the central position in the sequence, the second 8 must be placed in the first half of the sequence. Such constraints have been added in what follows<sup>1</sup>. Smallest-domain-first variable ordering (sdf) has been used for all the results shown; it is much quicker than lexicographic ordering, especially for the  $p_i$  model.

Table 3 shows the results of using this model to find all solutions, or prove that there is no solution, and compares the effect of different levels of propagation on the allDifferent constraint. In this case, the overhead of maintaining GAC on the allDifferent constraint is outweighed by the saving in search, so that the overall time taken is reduced.

allDiff constr.	Prob. (3,9)		Prob. (3,10)		Prob. (3,11)		Prob. (3,12)	
	fails	sec.	fails	sec.	fails	sec.	fails	sec.
binary	469	0.64	1557	2.42	7256	10.8	31008	47.8
global	208	0.57	658	1.88	2553	8.34	10421	33.5

Table 3: No. of backtracks and approximate running time (on a 166MHz Pentium PC) to find all solutions to four instances of Langford’s problem, treating the allDifferent constraints as either binary  $\neq$  constraints or as a global constraint.

The model just described finds the place in the sequence for each digit. The dual model finds the digit to go in each place in the sequence. For the

<sup>1</sup>Because of this, the results given here are not the same as in [11], where symmetry-breaking constraints were not used.

(3,9) instance, there are 27 variables  $d_j, j = 1, 2, \dots, 27$ , whose possible values correspond to the three 1s, the three 2s, ...

As before, constraints are needed to ensure the correct separation of the different copies of a digit as well as an allDifferent constraint. Unfortunately, it is much harder to express the separation constraints in Solver in terms of the  $d_j$  variables, in a form that can propagate well. With the resulting model, it is possible to solve very small instances, e.g. (2,4), but finding all solutions to the (3,9) instance takes far too long.

However, the fact that the  $d_j$  variables do not themselves allow the problem to be solved easily is no barrier to combining these variables with the  $p_i$  variables to give a minimal combined model, as for the  $n$ -queens problem. We need only define the constraints in terms of one set of variables: in this case the  $p_i$  variables are the obvious choice.

We can then use either the  $p_i$  variables or the  $d_j$  variables, or both, as the search variables, even though the constraints are defined only in terms of the former. The results are shown in Table 4. As in the  $n$ -queens problem,

Search vars	allDiff constr.	Prob. (3,9)		Prob. (3,10)		Prob. (3,11)		Prob. (3,12)	
		fails	sec.	fails	sec.	fails	sec.	fails	sec.
$p_i$	-	213	0.55	666	1.70	2618	6.83	10597	28.9
$p_i$	global	208	0.66	658	2.36	2553	9.11	10421	38.6
$d_j$	-	178	0.50	549	1.54	2120	6.00	7926	23.0
$d_j$	global	170	0.64	517	1.83	2023	7.14	7527	27.3
both	-	120	0.36	433	1.22	1550	4.34	6240	18.0
both	global	114	0.46	419	1.50	1514	5.31	6153	22.2

Table 4: No. of backtracks and approximate running time to find all solutions to four instances of Langford’s problem, using a minimal combined model based on the  $p_i$  variables.

binary  $\neq$  constraints on the primal variables are redundant if the channelling constraints are defined. Hence, the second column indicates that either the channelling constraints alone ensure that solutions are permutations, or that a global allDifferent constraint is posted on the primal variables. The number of fails in the second line is the same as in the second line of Table 3: in both cases the  $p_j$  variables are the search variables and the allDifferent constraint is made GAC. As in the  $n$ -queens problem, adding the channelling constraints to a global allDifferent constraint gives no additional domain pruning, and the running time is increased. However, the channelling constraints on their own give almost as much domain pruning as the allDifferent constraint, with a smaller running time.

Clearly, the channelling constraints are necessary even in conjunction with the global allDifferent constraint when the search variables are either the  $d_j$  variables or both the  $p_i$  and  $d_j$  variables, since they are then fulfilling their original role of linking the two sets of variables. Table 4 shows that the  $d_j$  variables are a better choice of search variable than the  $p_i$  variables. This is perhaps unexpected, because with sdf variable ordering, the  $p_i$  variables give a smaller branching factor at the root of the search tree than the  $d_j$  variables. In the (3,9) problem, for instance, variable  $9_1$  can only go in positions 1-4 (given the symmetry constraints). In the dual model, any one of  $1_1, 2_1, \dots, 9_1$  can be assigned to  $d_1$  at the start, and this is the smallest domain size. However, after

the first choice has been made, the smallest domain size amongst the  $p_i$  variables increases, whereas with the  $d_j$  variables it decreases, and this is sufficient to give a smaller search tree overall using the  $d_j$  variables.

Table 4 also shows that being able to search on both sets of variables together, using the smallest domain size heuristic, is a useful additional benefit of dual modelling, just as in the  $n$ -queens problem. With this heuristic, finding all solutions to Langford’s problem, or proving there is no solution, for all four instances, is quicker than searching on either set separately. In all the experiments shown, the value assigned to a variable is just the smallest in its domain: [11] also gives experimental results on using the dual variables as the basis for value ordering in Langford’s problem. This value ordering heuristic was used by Cheng *et al.* [1]; having chosen a variable, the value assigned to it is the one corresponding to the dual variable with smallest domain. When finding the first solution, this heuristic gives patchy results on Langford’s problem; however, [11] shows that it gives uniformly better results than the default ordering when finding all solutions or proving that there are no solutions, when the search variables are either the  $p_i$  or  $d_j$  variables, but not both. This is unexpected, given that the accepted wisdom on value ordering is that it makes no difference when finding all solutions or if there is no solution. However, the best results are still obtained by using smallest domain variable ordering on both primal and dual variables combined, and in that case using the dual variables for value ordering as well makes little difference. The reason suggested in [11] is that the same information is essentially being used twice (for both variable and value ordering).

As with the  $n$ -queens problem, combining two complete models gives the same number of fails as in Table 4, but takes much longer, because of the additional overhead.

Hence, the most successful of the strategies considered here in finding all solutions to Langford’s Problem, modelled as a CSP, is to define the problem constraints on the  $p_i$  variables only, to define the  $d_j$  variables as their duals, and to use the minimal combined model. The resulting CSP should be solved using the sdf variable ordering on both sets of variables at once. Although adding a global allDifferent constraint gives some additional pruning of the search tree, it increases the running time. Hence, the allDifferent constraint is unnecessary; it can be superseded by the channelling constraints. It is remarkable that this strategy makes the  $d_j$  variables very successful as search variables, even without a good model of the problem using these variables alone.

### 3 A Quasi-Permutation CSP

Nadel [7] lists nine different ways of modelling the  $n$ -queens problem as a CSP. Apart from those already considered, two of the others are also duals. Since there is at most one queen on each diagonal, a variable can be associated with each upper-left  $\rightarrow$  lower-right diagonal whose values correspond to the diagonals in the opposite direction, together with a dummy value to allow for the fact that the diagonal may not contain a queen. The variables of the dual model also correspond to the opposite diagonals. As with the row/column models, the primal and dual models are identical apart from a renaming of variables.

Suppose the upper-left  $\rightarrow$  lower-right diagonals are numbered from 0 to

$2n - 2$ , starting at the lower left, i.e. diagonal 0 consists of just the bottom left square, and diagonal  $n - 1$  is the main diagonal in this direction. The opposite diagonals can be numbered starting at the top left, say.

To allow a constraint that  $d_0, d_1, \dots, d_{2n-2}$  are all different to represent the restriction that each diagonal contains at most one queen, each variable can be given a different dummy value. This gives us a *quasi-permutation* CSP [12], since the variables must all have different values but there are more values than there are variables.

The conditions that there is at most one queen on every row and at most one in every column are represented by the constraints  $d_{i+j} - d_i \neq j$  and  $d_i - d_{i+j} \neq j$  for  $i = 0, \dots, 2n-3$  and  $j = 1, 2, \dots, \min(2n-2-i, n-1)$ , provided that the dummy values are chosen suitably (at least  $n$  larger than any ‘real’ value, and increasing in steps of at least  $n$ , so that any two dummy values satisfy these constraints). We also need to ensure that there are  $n$  queens on the board, which can be done by a constraint that the number of variables with real values is  $n$ .

If the dual variables (corresponding to the diagonals in the opposite direction) are  $e_0, e_1, \dots, e_{2n-2}$ , the channelling constraints are that  $d_i = j$  iff  $e_j = i$ , for  $0 \leq i, j < 2n - 1$ , as with the dual models of  $n$  queens considered in section 2. However, in this case, the channelling constraints do not themselves lead to any search reduction. The circumstances in which the channelling constraints in permutation problems can prune values do not arise here; if a value (other than dummy value) appears in the domain of only one variable, the dual variable will not have a singleton domain since it will also contain its dummy value.

Combining the two complete models does give some reduction in the number of fails: the additional pruning is due solely to the constraint in the dual model that exactly  $n$  diagonals must have queens placed on them. In terms of running time, however, it is not worthwhile to add the dual variables and the channelling constraints in order to be able to add this constraint and so reduce search.

## 4 The Golfers Problem

The dual of a model with integer variables does not always have integer variables itself. If several variables in the integer model must be assigned the same value, a dual model will have variables which must be assigned several values at once. This situation can be modelled by set variables whose values are sets of integers. ILOG Solver, for instance, supports set variables: their domains are represented by upper and lower bounds. The lower bound is the set of elements which are members of any possible set of the variables (the *required set*) and the upper bound is the set of elements which are members of at least one possible value of the variable (the *possible set*).

Cheng *et al.* describe a nurse rostering example in which a model using integer variables is combined with one using set variables; again, two complete models were linked, incurring a significant overhead. Here a problem is presented which can similarly be modelled using both set and integer variables. Problem 10 in CSPLib<sup>2</sup> 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?*

---

<sup>2</sup>CSPLib is a problem library for constraints, at [www-users.cs.york.ac.uk/~tw/csplib/](http://www-users.cs.york.ac.uk/~tw/csplib/)

Two different ways of modelling the problem are presented below, each of which allows dual models to be linked. The problem presents a high degree of symmetry and handling this successfully is crucial to solving large instances using constraint programming; the principal motivation for the second, more complex, model is to reduce symmetry.

## 4.1 Group-based Models

The Eclipse example program in CSPLib uses a set variable to represent each group of golfers in each week. Suppose that in general there are  $n$  players, to be arranged in  $g$  groups of  $s$  players in each week, playing for  $w$  weeks. Each group can be represented as a set variable:  $G_{ik}$  is the  $i^{\text{th}}$  group in week  $k$ . The constraints are then that:

- the cardinality of each set is  $s$
- the groups in any week do not overlap, i.e.  $G_{ik} \cap G_{jk} = \emptyset$ ,  $1 \leq i < j \leq g$ , for all  $k$ ,
- any two groups in different weeks have at most one member in common

In addition, some simple symmetry breaking can be done: the groups in the first week can be specified, as say  $\{1, 2, \dots, s\}$ ,  $\{s + 1, \dots, 2s\}$ , ..... and after the first week, players 1, 2, ..  $s$  must be in different groups, so these can be groups 1, 2, ...,  $s$ .

An alternative model uses integer variables:  $p_{lk}$  has value  $i$  if player  $l$  is in group  $i$  in week  $k$ . This is the dual of the set model, since the variables of that model represent groups, and the values are (sets of) players, whereas the variables of the integer model represent players and the values are groups. The constraints can be represented equally well in this model: in fact, the constraints that the groups in a week do not overlap are not needed, since each variable (player) can only be assigned one value (group). The constraint that the cardinality of the set variable corresponding to the  $i^{\text{th}}$  group in week  $k$  must be equal to  $s$  can be replaced by a constraint that the number of variables  $p_{lk}$  assigned the value  $i$  must be equal to  $s$ . Each model can be combined with its dual, as with the permutation problems considered earlier. Either the integer model or the set model can be considered as the primal model, and the set or integer variables, respectively, added to it, with suitable channelling constraints.

The channelling constraints are of the same form as those in the nurse rostering example given by Cheng *et al.*:  $p_{lk} = i$  iff  $l \in G_{ik}$ . These channelling constraints do not give a one-to-one mapping between assignments in the respective models, since an assignment in the set model would correspond to several assignments in the integer model. However, there is a one-to-one mapping between an assignment to an integer variable and the allocation of an element to the dual set variable. In Solver, the action of the channelling constraints ensures that  $p_{lk}$  is assigned the value  $i$  iff  $l$  is in the required set of  $G_{ik}$ , and  $i$  is not in the domain of  $p_{lk}$  iff  $l$  is not in the possible set of  $G_{ik}$ .

In contrast to permutation problems, there appears to be no reduction in search from the channelling constraints, and so solving the problem takes longer because of the extra variables and constraints. However, circumstances can be imagined in which the channelling constraints might lead to additional pruning,

if the set variables are used as search variables. By analogy with permutation problems, where the channelling constraints detect and assign values which occur in the domain of only one primal variable, a value (in this case a golfer) might be a possible member of only one of the set variables representing the groups in a given week. In that case, it must be a member of that group. In the integer model, the assignment would be done automatically (the variable representing the player has only one possible value) but in the set model, there is no way of explicitly representing the fact that each player must belong to a group each week; this is only represented implicitly by the fact that there are  $g$  mutually exclusive groups of size  $s$ , and  $gs$  players in all. In this problem, these circumstances do not appear to occur, but in other problems they might conceivably lead to a significant reduction in search.

Even so, a dual model of the golfers problem is still useful: there are constraints that could be added to the basic models which are only easy to express using one set of variables. Firstly, we can add more symmetry-breaking constraints to distinguish between the weeks after the first, e.g. that the smallest player number in week  $k$  in the same group as player 1 is smaller than the corresponding number in week  $k + 1$ , for  $k = 1, 2, \dots, w - 1$ . It does not seem possible to express these constraints in Solver in terms of the set variables. Secondly, we can add implied constraints, due to Warwick Harvey, which reduce the number of backtracks considerably: for all  $i, k$ ,  $G_{ik}$  has a member in common with exactly  $s$  of the groups in any other week. These constraints are naturally expressed in terms of the set variables.

The basic constraints of the two models seem to have exactly the same pruning power. For instance, the integer model (including the symmetry-breaking constraints just described) gives the same number of fails as the set model together with the integer variables, the channelling constraints and the symmetry-breaking constraints, using the integer variables as the search variables. Similarly, searching on the set variables gives the same number of fails whether the basic constraints are defined in terms of the integer variables or the set variables. However, because the constraints to distinguish between the weeks need the integer variables, a pure set model can give very poor results, especially when there is no solution. For instance, proving that 4 groups of 3 golfers cannot play for 5 weeks takes 2.7m. fails using the set variables alone. If the integer variables are included to allow the symmetry-breaking constraints to be defined, but the set variables are still used as the search variables, the number of fails is reduced by 93%.

## 4.2 Pair-based Models

Even with the symmetry-breaking constraints, the model just described still has a high degree of symmetry. For instance, the groups within a week are interchangeable, apart from the first week and the first  $s$  groups thereafter. The model described in this section has less intrinsic symmetry, because the numbering of the groups within a week is dispensed with. Again dual integer and set models are defined.

In the integer model, there is a variable  $q_{ij}$  for each pair of players  $i$  and  $j$ . Its value is the week in which these players are in the same group, or a value representing a dummy week, in case  $i$  and  $j$  never play each other.

The groups are no longer explicitly represented. However, this means that

some of the constraints are cumbersome to represent. In particular, a large number of constraints are needed to ensure transitivity: if  $q_{ij} = k$  and  $q_{jl} = k$ , then  $q_{il} = k$ . We also need constraints to ensure that each person plays  $s - 1$  other people each week; this is straightforward, but such a constraint is needed for every person in every week, rather than for each group in each week as before. On the other hand, the constraints to ensure that each pair of players are assigned to the same group at most once are no longer needed: this is implicit in the fact that there is a variable for each pair, which must be assigned exactly one value (including the dummy week).

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.

The need for transitivity constraints can be avoided by introducing set variables to represent the groups, alongside the integer variables.  $H_{ik}$  is defined to be the set of players that play with player  $i$  in week  $k$  (including him/herself). The constraints to ensure that the size of each group is  $s$  can be defined in terms of either the  $q_{ij}$  variables or the  $H_{ik}$  variables.

We need to include channelling constraints to link the two sets of variables. It is less obvious what they should be in this case. Suitable constraints were found to be:

- $q_{ij} = k$  iff  $H_{ik} = H_{jk}$
- $q_{ij} \neq k$  iff  $H_{ik} \cap H_{jk} = \emptyset$

Table 5 shows that adding elements of the set model to the integer model can reduce search, apart from the fact that the transitivity constraints are then unnecessary. Some experimentation is necessary to establish which elements of each model should be used. For instance, in theory, the size of each group could be determined by constraints in either the integer or the set model, or both. In practice, although any of these choices give correct solutions, it is better to keep the constraints of the integer model. As shown in the third column of Table 5, defining the group size only by cardinality constraints on the set variables increases the number of fails. In fact, these cardinality constraints are redundant: if constraints on the integer variables are used to determine the group size, adding cardinality constraints on the set variables as well has no effect on the search.

No. of weeks	Integer model		Combined (a)		Combined (b)	
	fails	sec.	fails	sec.	fails	sec.
3	107	0.25	95	0.33	126	0.40
4	91	0.63	89	0.72	196	1.02
5	11541	88.6	8409	77.6	70681	357

Table 5: Comparison of integer and combined pair-based models on instances of the golfers problem with 4 groups of 3 players (on a 166MHz PC). (a) Define group size by constraints in the integer model; (b) define group size by cardinality constraints on set variables.

As before, the five-week instance is especially difficult because it has no solution, and without the additional symmetry-breaking constraints ordering the weeks it takes much longer. This is one reason why the set-based model alone would not be a good choice, since these constraints need the integer variables. Furthermore, the instances in Table 5 are small: to solve larger instances, the remaining symmetry needs to be addressed. Although not within the scope of this paper, SBDS has proved useful for this problem, and (at least in its current Solver implementation) requires integer search variables.

## 5 Boolean formulations

### 5.1 Permutation Problems

The dual models given for the  $n$ -queens problem can both be transformed into a model with Boolean-valued variables:  $x_{ij} = 1$  if there is a queen in row  $i$ , column  $j$ , 0 otherwise. An equivalent transformation is used by Weigel and Bliet [13], as a means of deriving reformulations of CSPs: their approach will be discussed further below in section 7. It is assumed here that once the variables of the Boolean model have been decided, the constraints can be added by hand. However, two sets of constraints in the Boolean transformation are of interest: those ensuring that there is exactly one queen on every row and exactly one queen in every column:

$$\sum_j x_{ij} = 1 \quad \text{for } i = 1, \dots, n$$

$$\sum_i x_{ij} = 1 \quad \text{for } j = 1, \dots, n$$

Boolean variables of the same form, with the two sets of constraints just given, could be used to reformulate any permutation CSP: the Boolean models would only differ in the representation of the constraints determining the allowable permutations (in the  $n$ -queens problem, the diagonal constraints).

We might view the  $r_i$  model of the  $n$ -queens problem, with variable  $r_i = j$  if the queen on row  $i$  is in column  $j$ , as derived from this Boolean model by transmuting one of the two subscripts of the  $x_{ij}$  variables into a set of values. In comparison with the Boolean model, the  $r_i$  model does not need constraints that there is one queen on every row. The constraint that there is one queen in every column is fulfilled by a constraint that a solution must be a permutation: either an allDifferent constraint, or the channelling constraints in the combined model. Clearly *either* subscript of the  $x_{ij}$  variables could be converted into a set of values in this case, confirming that the  $r_i$  and  $c_j$  models are duals. The same transformations between a Boolean model and a pair of dual integer models could be done for any permutation problem.

The quasi-permutation CSPs given by the diagonal models of  $n$ -queens described in section 3 lead to a similar Boolean model with variables:

$$x_{ij} = 1 \text{ if there is a queen in } \begin{array}{l} \text{upper-left} \rightarrow \text{lower-right diagonal } i, \\ \text{lower-left} \rightarrow \text{upper-right diagonal } j \end{array}$$

$$0 \text{ otherwise.}$$

The constraints that there is at most one queen on any diagonal become:

$$\sum_j x_{ij} \leq 1 \quad \text{for } i = 0, \dots, 2n - 2$$

$$\sum_i x_{ij} \leq 1 \quad \text{for } j = 0, \dots, 2n - 2$$

with, in addition, a constraint that there are exactly  $n$  queens on the board:

$$\sum_{i,j} x_{ij} = n$$

The first two constraints can be converted to equations by introducing Boolean variables to indicate whether or not a queen is placed on a given diagonal, e.g.

$$\begin{aligned} \delta_i = 1 &\iff \sum_j x_{ij} = 0 \\ &= 0 \iff \sum_j x_{ij} = 1 \end{aligned}$$

In terms of the integer models,  $\delta_i = 1$  iff  $d_i$  is assigned its dummy value. In section 3, the dummy values are chosen to be all different so that the constraint  $\sum_j x_{ij} + \delta_i = 1$  can be translated into an allDifferent constraint. However, the Boolean model, as described here, only indicates the basic variables and values of the different models, but not in detail how the constraints should be defined.

## 5.2 Latin Squares

The examples just given suggest that dual models can be viewed as different transformations of a Boolean model. Applying the same transformations to the problem of finding an  $n \times n$  Latin square (although this is not a challenging problem) shows that sometimes more than two dual models can be found. A Boolean CSP for this problem has variables:

$$\begin{aligned} x_{ijk} &= 1 \text{ if the number in row } i, \text{ column } j \text{ is } k, \quad 1 \leq i, j, k \leq n \\ &0 \text{ otherwise.} \end{aligned}$$

and the (only) constraints are that each number occurs exactly once on every row and exactly once in every column and each cell contains exactly one number:

$$\sum_j x_{ijk} = 1 \quad 1 \leq i \leq n, 1 \leq k \leq n$$

$$\sum_i x_{ijk} = 1 \quad 1 \leq j \leq n, 1 \leq k \leq n$$

$$\sum_k x_{ijk} = 1 \quad 1 \leq i \leq n, 1 \leq j \leq n$$

It is clear from the symmetry between the  $i$ ,  $j$  and  $k$  subscripts that we can choose any one of the three subscripts to be the values of variables with two subscripts i.e.  $w_{jk} = i$  if the number  $k$  in column  $j$  occurs in row  $i$ ;  $y_{ik} = j$  if the number  $k$  in row  $i$  occurs in column  $j$ ;  $z_{ij} = k$  if the number in row  $i$ , column  $j$  is  $k$ .

As in  $n$ -queens, one set of constraints in each case is then no longer necessary, and the other two will become allDifferent constraints. For instance,  $w_{1k}, w_{2k}, \dots, w_{nk}$  are all different (the occurrences of the number  $k$  in each column are in different rows) and  $w_{j1}, w_{j2}, \dots, w_{jn}$  are all different (in column  $j$ , the numbers 1, 2, ...,  $n$  occur in different rows). Any pair of the dual models could be combined, or all three together, with channelling constraints linking each pair. However, it seems likely that linking more than two dual models will yield diminishing returns.<sup>3</sup>

### 5.3 Boolean Models of the Golfers Problem

A Boolean model derived from the group-based models of the golfers problem, given in section 4.1, has variables  $x_{ijk}$ , where:

$$x_{ijk} = 1 \text{ if golfer } i \text{ plays in group } j \text{ in week } k, 1 \leq i \leq n, 1 \leq j \leq g, 1 \leq k \leq w, \\ 0 \text{ otherwise.}$$

Some of the constraints in this model are that each group in each week contains exactly  $s$  golfers, where  $s$  is the group size:

$$\sum_i x_{ijk} = s \quad 1 \leq j \leq g, 1 \leq k \leq w$$

and each player plays in exactly one group each week:

$$\sum_j x_{ijk} = 1 \quad 1 \leq i \leq n, 1 \leq k \leq w$$

However, there is no limit (other than  $w$ ) on  $\sum_k x_{ijk}$ , i.e. on the number of weeks that player  $i$  is in group  $j$ .

If we consider dropping each subscript in turn, we cannot have variables:

$$x_{jk}^* = i \quad 1 \leq j \leq g, 1 \leq k \leq w$$

meaning that  $i$  is the golfer that plays in group  $j$  in week  $k$ , because each group contains more than one golfer. But as shown earlier, we can use set variables instead of integer variables:

$$G_{jk} = \{i | i \text{ plays in group } j \text{ in week } k\}$$

---

<sup>3</sup>It might seem that the Mystery Shopper problem described by Cheng *et al.* is a real example where more than two simultaneously dual models can be found. However, although their paper suggests that any pair of the four models given for the Mystery Shopper problem can be related by simple channelling constraints of the form  $x_i = j$  iff  $y_j = i$ , this is not so. The number of visits to each saleslady is different from the number of visits by each shopper, and there is no simple mapping from one set of visits to the other. For instance, it is not possible to say that shopper  $i$  visits saleslady  $j$  on her  $k^{\text{th}}$  visit iff the  $k^{\text{th}}$  visit to saleslady  $j$  is by shopper  $i$ . I am grateful to Jimmy Lee for clarifying this point.

with constraints to ensure that the cardinality of each of these variables is  $s$ , the equivalent of  $\sum_i x_{ijk} = s$ , and further constraints to ensure that the groups are mutually exclusive, the equivalent of  $\sum_j x_{ijk} = 1$ .

Similarly, we can have:  $p_{ik} = j$  where  $j$  is the group in which golfer  $i$  plays in week  $k$ , since  $\sum_j x_{ijk} = 1$ . As seen earlier, the models with variables  $p_{ik}$  and  $G_{jk}$  are duals.

Although the Boolean model has three subscripts, we cannot have variables

$$x_{ij}^* = k \quad 1 \leq i \leq n, 1 \leq j \leq g$$

meaning that  $k$  is the week in which golfer  $i$  plays in group  $j$ , because the numbering of the groups is purely arbitrary, and a golfer could play in, say, group 1, every week. This corresponds to the fact that there is no constraint on  $\sum_k x_{ijk}$  in the Boolean model.

We could similarly consider a Boolean model based on the integer and set models of the golfers problem given in section 4.2, with variables:  $x_{ijk} = 1$  if players  $i$  and  $j$  play together in week  $k$ , 0 otherwise.

Some of the constraints are that each pair of players play together at most once:

$$\sum_k x_{ijk} \leq 1$$

and each player plays with  $s - 1$  other players in week  $k$ :

$$\sum_i x_{ijk} = s - 1 \text{ and } \sum_j x_{ijk} = s - 1$$

These give the models described in section 4.2, with integer variables  $q_{ij}$  if the subscript  $k$  is dropped from the Boolean model and set variables  $H_{ik}$  (or equivalently  $H_{jk}$ ) if either of the other two subscripts are dropped.

As with the quasi-permutation models of the  $n$ -queens problem, the Boolean model contains inequality constraints ( $\sum_k x_{ijk} \leq 1$ ). In the integer model, this corresponds to the fact that there is a dummy week in the domain of each of the variables  $q_{ij}$  represent the case that the players  $i$  and  $j$  do not play together in weeks 1 to  $w$ .

## 5.4 Deriving dual models

The examples just considered suggest that a possible way of finding a dual model of a CSP is to convert its variables and values to a set of Boolean variables: the values of the original variables will correspond to a subscript of the new variables. It may then be possible, in effect, to drop a different subscript from the Boolean variables and form a dual model. A set of constraints of the form:

$$\sum_{i_1} x_{i_1 i_2 \dots i_k} = 1, \quad \forall i_2, i_3, \dots, i_n$$

in the Boolean model indicates that a model with integer variables could be derived by converting the range of values of  $i_1$  into the domains of the integer variables, say  $y_{i_2 i_3 \dots i_n}$ . The constraints indicate that each variable must be

assigned exactly one of the possible values of  $i_1$ . If the constraint has  $\leq$  replacing  $=$ , then in the integer model each variable must be assigned *at most* one of the possible values of  $i_1$ , and the domain of each variable must be augmented with a suitable dummy value.

A set of constraints:

$$\sum_{i_1} x_{i_1 i_2 \dots i_k} = s_{i_2 \dots i_m} \quad \text{where } s_{i_2 \dots i_m} > 1$$

on the other hand, indicates that a model with set variables could be derived. The set variable  $Y_{i_2 i_3 \dots i_m}$  must in this case be assigned a set of  $s_{i_2 \dots i_m}$  values. In the golfers problem, all the resulting set variables are required to have the same cardinality,  $s$ , but this is not so in general. For instance in the nurse rostering problem, the Boolean variables might be  $x_{ijk}$ , where  $x_{ijk} = 1$  means that nurse  $i$  will do shift type  $j$  on day  $k$ , with constraints  $\sum_i x_{ijk} = s_{jk}$  where  $s_{jk}$  is the number of nurses required to do type  $j$  shifts on day  $k$ . Again, the constraint might be an inequality rather than an equation, but no examples have been met in this paper.

It might seem that it is unnecessary to consider the Boolean formulation in order to find dual models, and indeed considering the Boolean formulation as proposed here will only suggest the variables and values of the dual model, and not help in specifying its constraints. Nevertheless, in practice the Boolean formulation has led to dual models which might not have been obvious otherwise, especially when there is more than one possible dual.

## 6 Combining Models which are not Duals

In the last section, it was shown that the models which have been combined in previous sections can be viewed as duals because they can be obtained from the same Boolean model of the problem by different translations of it, and so can be thought of as complementary. However, we might have different models of the same problem which cannot be related in this way.

One possibility not so far considered is to combine a (set or integer) CSP with the Boolean equivalent given in the last section, rather than with a dual model derived from the Boolean model. For instance, the model of  $n$ -queens with variables  $r_i$ ,  $i = 1, 2, \dots, n$  could be combined with the model with Boolean variables  $x_{ij}$ . The channelling constraints are straightforward:

$$r_i = j \text{ iff } x_{ij} = 1 \quad \forall i, j$$

In this case, the minimal combined model, adding just the  $x_{ij}$  variables and the channelling constraints to the  $r_i$  model, gives no reduction in search, and adds considerably to the running time.

However, adding the constraints  $\sum_i x_{ij} = 1$ ,  $j = 1, 2, \dots, n$ , to ensure that there is exactly one queen in each column, does have an effect on search. The number of fails is the same as with the minimal combined model shown in column (a) of Table 2.1, although the running time is again increased. As with the inverse constraints considered in section 2.1, the channelling constraints given here, together with the constraints  $\sum_i x_{ij} = 1$ , detect that a value that appears in the domain of only one primal variable, say  $r_p$ , must be assigned to

that variable. By the time that this value, say  $q$ , has been removed from the domain of every other  $r_i$  variable, the value 1 will have been removed from the domain of the corresponding Boolean variable  $x_{iq}$  by the channelling constraint. Eventually, the constraint  $\sum_i x_{ij} = 1$  can only be satisfied if  $x_{pq} = 1$ , and the channelling constraints again ensure that  $r_p$  is assigned the value  $q$ . Hence the channelling constraints between the  $r_i$  and  $x_{ij}$  variables give an alternative way of achieving the same level of propagation of the allDifferent constraint as the inverse constraint.

Adding the remaining (diagonal) constraints on the  $x_{ij}$  variables to the combined model gives the same results in terms of the number of fails as combining the complete  $r_i$  and  $c_j$  models, although again with a greater running time. A further disadvantage of combining the  $r_i$  model with the Boolean model rather than with its dual is that it would not be sensible to use both the  $r_i$  and  $x_{ij}$  variables together as search variables with a dynamic ordering strategy based on domain size. In contrast, as shown in Section 2, using both primal and dual variables together as search variables is very successful for both the  $n$ -queens and Langford’s problems.

Hence, if we considered combining the  $r_i$  model with the Boolean model in isolation, we might think that this was a potentially useful combination, giving considerably reduced search in comparison with the  $r_i$  model, and that for some problems, the increase in running time might be outweighed by the reduction in search. However, in comparison with the combination of the dual variables  $r_i$  and  $c_j$ , it is clearly inferior.

If we have two CSP models of the same problem, can we *always* combine them using channelling constraints? The models given in section 4 show that we cannot. It would not be possible to combine a group-based model with a pair-based model, because the group-based models explicitly refer to the  $i^{th}$  group in week  $k$ , whereas there is no such idea in the pair-based model. Considering the integer models, we cannot, as required, map an assignment in one model, meaning “player  $i$  plays in group  $l$  in week  $k$ ” to an assignment in the other, meaning “players  $i$  and  $j$  play together in week  $k$ ”.

However, in all the cases considered here, although the variable domains are integers or sets of integers, these values are simply labels for problem entities. If the possible values of the variables represent true numeric quantities (which is likely to mean that the domains are relatively large and most of the values will not be assigned to any variable) these transformations are unlikely to yield any usefully different views of the problem. An example is the template design problem, described in [8], where it has so far proved impossible to devise any promising alternative model of the problem as a CSP.

## 7 Alternative Reformulations using a Boolean Model

As already mentioned, Weigel and Bliet [13] use a similar Boolean formulation of a CSP to explore different ways of reformulating problems. The Boolean variables they define are equivalent to those used here. The constraints of the Boolean model are firstly uniqueness constraints, corresponding to the fact that each variable in the original CSP can be assigned only one value from its

domain. Secondly, for every disallowed value tuple given by the constraints of the original CSP, there is a constraint of the same arity on the Boolean variables: ensuring that at most one of these Boolean variables is assigned the value 1 is equivalent to ensuring that at most one of the values in the disallowed tuple is assigned to the corresponding variable. The uniqueness constraints, in the notation of this paper, are of the form  $\sum_j x_{ij} = 1$ . Weigel and Bliks replace each uniqueness constraint by its binary projections, e.g.  $x_{ij_1} + x_{ij_2} \leq 1$ , for all pairs  $j_1, j_2$ , together with a counting constraint to ensure that  $n$  Boolean variables are assigned the value 1, where  $n$  is the number of variables in the original CSP.

If the original CSP is binary, so is the Boolean formulation just given, apart from the  $n$ -ary counting constraint. The constraint graph will contain a clique for every set of binary constraints projected from a uniqueness constraint, i.e. for every variable in the original CSP. By partitioning the nodes of this graph differently, another set of cliques (a new *clique cover*) can be obtained. This then gives rise to a new integer reformulation of the original CSP, with an integer variable corresponding to each clique. The domain size of an integer variable is the clique size + 1, corresponding to each of the Boolean variables in the clique being assigned the value 1, together with a dummy value for the case that none of them are.

However, if the new partition of the nodes also results in  $n$  cliques, then exactly one Boolean variable from each clique must be assigned the value 1, and in that case the additional value can be removed from the domain of the new integer variables. Weigel and Bliks propose that this can be viewed as a redundant (or implied) constraint in the Boolean formulation. For instance, in the  $n$ -queens problem, if the original CSP has variables  $r_i, i = 1, 2, \dots, n$  corresponding to the rows of the board, the constraint graph of the Boolean formulation can be repartitioned to give instead  $n$  cliques corresponding to the columns. The redundant constraints represent the fact that there is exactly one queen in each column.

Weigel and Bliks suggest that combining representations is potentially valuable because of these redundant constraints, and that these are sufficient to explain the extra constraint propagation found by Cheng et al. [1] when combining models: “we can explain that the additional pruning reported .. is due to redundant uniqueness constraints coming from different formulations.” The foregoing examples show that this is not the whole story. In section 6, the example of combining the Boolean model of  $n$ -queens with the row model shows that the constraints of the Boolean model ensuring that there is exactly one queen in each column do indeed achieve the same level of constraint propagation as the inverse constraints combining the row and column integer models. However, that example also showed that combining the row and column models is a much better way of achieving the same level of propagation.

Weigel and Bliks’s method does allow some reformulations to be found, by generating new clique covers, which would be less obvious using the method described here. For instance, the Boolean model of the  $n$ -queens problems derived from the row model can yield a diagonal model by choosing the cliques appropriately. It requires some insight to view these as dual models, although it can be done by relabelling the values of the row variables according to which diagonal each square is on, and relabelling the values of the diagonal variables as the rows of the board. Then the values of the variables in one model correspond

to the variables of the other, as required.

Section 5 uses Boolean models only as a guide to finding dual models, and does not attempt to automate the process. However, how the constraints are defined is an issue in modelling CSPs which Weigel and Blik's approach does not address. Part of the potential benefit of combining dual models is that one model may give a better way of representing the constraints, so that improved constraint propagation results, in the constraint programming tool being used. This assumes that constraints will, in general, be represented intensionally. In Weigel and Blik's scheme, the constraints of the Boolean model correspond to the inconsistent tuples of the original CSP. Representing the constraints in this way is unlikely to allow as much constraint propagation in practice, especially if there are non-binary constraints. For instance, constraint programming tools provide bounds consistency on arithmetic non-binary constraints and fast generalised arc consistency algorithms for constraints such as the allDifferent constraint, as well as fast AC methods for particular classes of binary constraint. It is important to model problems as CSPs in such a way that they can make best use of these algorithms. Furthermore, repartitioning the nodes of the Boolean constraint graph to give a new clique cover seems to work best for permutation problems, where a partition into  $n$  cliques can be rearranged to give another clique cover of size  $n$ . The method would need to be significantly modified to deal with set variables, as in section 4 for instance.

## 8 Conclusions

Given a constraint satisfaction problem, there is often a dual CSP whose variables correspond to the values of the original. The possible values of the dual variables may be integers, each corresponding to a variable of the primal CSP, or sets of integers if more than one of the primal variables can be assigned the same value. Although changing to the dual formulation may result in a CSP which is easier to solve, in this paper the focus has been on the potential benefits of combining the primal and dual models, using channelling constraints to link them.

Permutation CSPs are a special case in which the number of possible values is the same as the number of variables and each variable must be assigned a different value. The dual model is also a permutation CSP. It has been shown that when the primal and dual models are linked, the channelling constraints are sufficient to ensure that any solution is a permutation of the possible values, and that making this constraint arc consistent does more domain pruning than making the binary  $\neq$  constraints arc consistent, although not as much as making the allDifferent constraint generalized arc consistent. Hence, the channelling constraints do not simply link the primal and dual models together; they can themselves provide the main benefit of linking the two models. For some permutation problems, a minimal combined model is most cost-effective: the dual variables and values are defined, together with the channelling constraints, but none of the constraints of the dual model itself are used. In that case, rather than using the channelling constraints to link the two models, we are defining the dual variables in order to allow the channelling constraints to be posted.

Often, however, the principal benefit of combining primal and dual models is to allow the constraints to be easily expressed in a form that propagates well.

Sometimes (as in Langford’s problem) good results are obtained by defining the constraints in terms of one set of variables but using the dual variables as the search variables. In other cases, some constraints are better expressed in one model and others in its dual model. The principal benefit of combining dual models is then that it gives greater freedom to express the constraints.

When the primal and dual variables both have integer values, combining the two models gives a further choice: we can use both sets of variables as the search variables, moving from one set to another dynamically during the search for solutions. This has proved valuable for permutation problems: when using a smallest domain ordering, it allows the search to swap between the (primal) variable which has fewest values and the (primal) value which can be assigned to fewest variables.

The disadvantage of combining the primal and dual models is the additional overhead from two sets of variables and more constraints to propagate. Because of this extra work, combining models may result in longer running time even if a smaller search tree is explored. It is likely that in any particular case, whether or not combining models is worthwhile can only be established by experiment. Apart from the variability between problems, it will also depend on how the channelling constraints are implemented. ILOG Solver, for instance, provides an inverse constraint which can be used to link the primal and dual variables in permutation and quasi-permutation CSPs. This is much more efficient than if the channelling constraints are defined directly, although it does exactly the same constraint propagation. In other cases, too, it might be worthwhile to re-implement the channelling constraints more efficiently, and this would affect the cost-effectiveness of combining dual models.

Finally, the primal and dual models have been related by way of a common Boolean formulation. This gives an alternative way of viewing dual models which suggests that sometimes, as with Latin squares, more than two simultaneously dual models of the same problem may be found. Although the Boolean formulation is equivalent to that used by Weigel and Blik [13], here it is suggested that it should be used simply to indicate the variables of an alternative model. Successfully combining dual models to give good results in practice still needs the user’s skill in expressing the constraints of the problem from one or both viewpoints.

## Acknowledgments

The author is a member of the APES research group (<http://www.dcs.st-and.ac.uk/~apes>) and would like to thank the other members.

## References

- [1] B. M. W. Cheng, K. M. F. Choi, J. H. M. Lee, and J. C. K. Wu. Increasing constraint propagation by redundant modeling: an experience report. *Constraints*, 4:167–192, 1999.
- [2] P. A. Geelen. Dual Viewpoint Heuristics for Binary Constraint Satisfaction Problems. In B. Neumann, editor, *Proceedings ECAI’92*, pages 31–35, 1992.

- [3] 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.
- [4] 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.
- [5] ILOG. *Solver 4.4 User's Manual*. 1999.
- [6] J. E. Miller. Langford's Problem. [Online]. Available at <http://www.lclark.edu/~miller/langford.html>. (Accessed June 30, 2000), Mar. 1999.
- [7] B. A. Nadel. Representation Selection for Constraint Satisfaction: A Case Study Using  $n$ -Queens. *IEEE Expert*, 5:16–23, June 1990.
- [8] L. G. Proll and B. M. Smith. ILP and Constraint Programming Approaches to a Template Design Problem. *INFORMS Journal on Computing*, 10:265–275, 1998.
- [9] J.-C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proceedings AAAI'94*, volume 1, pages 362–367, 1994.
- [10] 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.
- [11] T. Walsh. Permutation problems and channelling constraints. Technical Report APES-26-2001, January 2001.
- [12] R. Weigel and C. Bliet. On Reformulation of Constraint Satisfaction Problems. In *Proceedings ECAI'98*, pages 254–258, 1998.