

# Dual Formulations of Magic Squares

Barbara M. Smith

School of Computing, University of Leeds, U.K.  
bms@comp.leeds.ac.uk

**Abstract.** The problem of finding an  $n \times n$  magic square can be represented as a constraint satisfaction problem (CSP). It can be modelled as a permutation problem, and hence there are dual viewpoints in which the roles of the variables and values are reversed. In modelling problems as CSPs, choosing between different viewpoints and the resulting models usually requires taking the constraint solver into account, to decide which model will lead to better performance; but in this case, one viewpoint is clearly better than the other simply because of the number of constraints. For the viewpoint in which the variables represent the cells of the square, the number of constraints is linear, whereas the dual viewpoint leads to a model in which the number of constraints is exponential in  $n$ .

## 1 Introduction

In modelling a problem as a constraint satisfaction problem (CSP), there is often more than one possible viewpoint (giving the variables and values of the CSP) that could be used as a basis. Often, it is hard to choose between possible viewpoints without trying them out, and in fact the relative performance of models based on different viewpoints can depend on the search algorithm and search heuristics used to solve the CSP [1], so that neither viewpoint is inherently better than the other. Even when one viewpoint can be seen *a priori* to be better than an alternative, this is often because the constraints that can be expressed in one viewpoint will propagate better than those in the other; hence, the comparison between the viewpoints depends on the detail of how constraints propagate in the chosen constraint solver. Usually, there seems no unequivocal reason to prefer one viewpoint to another that does not depend on considering how the resulting CSP will be solved.

In this paper, two viewpoints for the problem of finding a magic square are discussed and it is shown that in this case one viewpoint is far preferable to the other, irrespective of the constraint solver. The problem is a permutation problem, and it is well-known that permutation problems can be modelled in two dual viewpoints, in which the roles of the variables and values are opposite. In this case one of the two viewpoints is inherently worse than the other, and in fact unusable except for small instances, because the number of constraints grows at least exponentially with the instance size, whereas the dual viewpoint gives a CSP in which the number of constraints increases linearly.

## 2 Preliminary Definitions

**Definition 1.** A CSP instance is a triple  $\langle V, D, C \rangle$  where:

- $V$  is a set of variables;
- $D$  is a universal domain, specifying the possible values for those variables;
- $C$  is a set of constraints. Each constraint  $c \in C$  is a pair  $c = \langle \sigma, \rho \rangle$  where  $\sigma$  is a list of variables from  $V$ , called the constraint scope, and  $\rho$  is a  $|\sigma|$ -ary relation over  $D$ , called the constraint relation.

An *assignment* of values to variables is a set  $\{\langle v_1, a_1 \rangle, \langle v_2, a_2 \rangle, \dots, \langle v_k, a_k \rangle\}$  where  $\{v_1, v_2, \dots, v_k\} \subseteq V$  and  $a_i \in D, \forall 1 \leq i \leq k$ . The constraint relation of a constraint

$c$  is intended to specify the assignments that are allowed by that constraint. A *solution* to the CSP instance  $\langle V, D, C \rangle$  is a mapping from  $V$  into  $D$  whose restriction to each constraint scope is in the corresponding constraint relation, i.e. is allowed by the constraint. The *viewpoint* on which a CSP is based is the pair  $\langle V, D \rangle$ ; this concept was introduced in [5]. The meaning ascribed to the variables and values of a viewpoint allow the complete assignments in the CSP to be related to possible solutions to the underlying problem being modelled; the role of the constraints, given a viewpoint, is to ensure that the solutions of the CSP are correct solutions to the problem.

If a CSP is binary (i.e. the constraints have arity at most 2), the details of the constraints can be captured in a graph, the *microstructure* [2, 4] of the instance.

**Definition 2.** For any binary CSP instance  $P = \langle V, D, C \rangle$ , the microstructure of  $P$  is a graph with set of vertices  $V \times D$  where each edge corresponds either to an assignment allowed by a specific constraint, or to an assignment allowed because there is no constraint between the associated variables.

The solutions of the CSP instance are the vertices of cliques of size  $|V|$  in the microstructure.

For some purposes, it is more convenient to deal with the complement of this graph. The *microstructure complement* has the same set of vertices as the microstructure, but with edges joining all pairs of vertices which are *disallowed* by some constraint, or else are incompatible assignments for the same variable. In other words, two vertices  $\langle v_1, a_1 \rangle$  and  $\langle v_2, a_2 \rangle$  in the microstructure complement are connected by an edge if and only if:

1. the vertices  $v_1$  and  $v_2$  are in the scope of some constraint, but the assignment of  $a_1$  to  $v_1$  and  $a_2$  to  $v_2$  is disallowed by that constraint; *or*
2.  $v_1 = v_2$  and  $a_1 \neq a_2$ .

The solutions to a CSP instance  $P$  are the independent sets (i.e. sets of vertices containing no edge) of size  $|V|$  in its microstructure complement.

The definition of the microstructure complement extends naturally to the non-binary case. Here the microstructure complement is a *hypergraph* whose set of vertices is again the set of all variable-value pairs.<sup>1</sup> In this case, a set of vertices  $E$  is a hyperedge of the microstructure complement if it represents an assignment *disallowed* by a constraint, or else consists of a pair of incompatible assignments for the same variable. In other words, a set of vertices  $\{\langle v_1, a_1 \rangle, \langle v_2, a_2 \rangle, \dots, \langle v_k, a_k \rangle\}$  is a hyperedge if and only if:

- $\{v_1, v_2, \dots, v_k\}$  is the set of variables in the scope of some constraint, but the constraint disallows the assignment  $\{\langle v_1, a_1 \rangle, \langle v_2, a_2 \rangle, \dots, \langle v_k, a_k \rangle\}$ ; *or*
- $k = 2$ ,  $v_1 = v_2$  and  $a_1 \neq a_2$ .

### 3 Permutation Problems

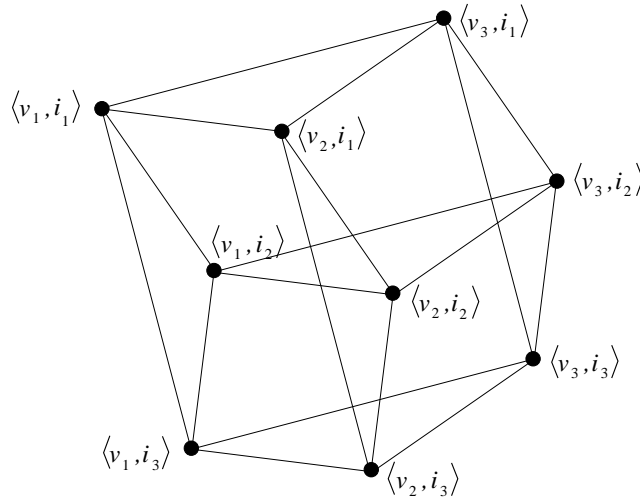
A CSP  $\langle V, D, C \rangle$  is a permutation problem if  $|D| = |V|$  and each variable must be assigned a different value. Any solution assigns a permutation of the values to the variables. Other constraints in the problem determine which permutations are acceptable solutions.

Each possible value is assigned to exactly one variable and each variable is assigned exactly one value. The *dual* viewpoint was identified by Geelen [3]; it switches the roles of the variables and values.

Let  $P = \langle V, D, C \rangle$  and  $P' = \langle V', D', C' \rangle$  be dual CSPs. Since the values of  $P$  correspond to the variables of  $P'$  and v.v., there is a one-one correspondence between the vertices of their microstructure complements.

<sup>1</sup> Note that I am *not* defining the microstructure of a non-binary CSP. In the non-binary case, ‘microstructure complement’ is just the name of a hypergraph; it is not intended to be the complement of another hypergraph, if indeed the complement of a hypergraph is well-defined.

In the microstructure complement of  $P$ , for each variable  $v_i \in V$  the vertices corresponding to  $v_i$  form a clique of size  $|D|$ , representing the fact that only one value can be assigned to this variable. There is also an allDifferent constraint on the variables in  $V$ , of arity  $|V|$ . This could be represented in the microstructure complement by hyperedges linking vertices  $\langle v_1, j_1 \rangle, \langle v_2, j_2 \rangle, \dots, \langle v_n, j_n \rangle$  for all tuples  $(j_1, j_2, \dots, j_n)$  whose elements are in  $D$  and are not all different. Alternatively and more simply, the allDifferent constraint can be decomposed into a set of binary  $\neq$  constraints, giving a set of edges in the microstructure complement joining the vertices  $\langle v_i, a \rangle$  and  $\langle v_j, a \rangle, \forall i, j \in V, a \in D$ . For each value  $a \in D$ , these edges form a clique of size  $|V|$ .



**Fig. 1.** The microstructure complement of a permutation problem with three variables, showing only the  $\neq$  constraints.

Figure 1 shows the microstructure complement of a permutation problem with three variables (and so three values), with these cliques. The independent sets of size 3 in this graph, for instance  $\{\langle v_1, i_1 \rangle, \langle v_2, i_2 \rangle, \langle v_3, i_3 \rangle\}$  correspond to all possible ways of assigning a permutation of the values  $i_1, i_2, i_3$  to the variables  $v_1, v_2, v_3$ . Typically, the microstructure complement will also have edges or hyperedges representing the permutations that are not allowed; these are specific to the problem being modelled by  $P$ .

In the microstructure complement of the dual CSP,  $P'$ , both types of clique are preserved; a clique representing the fact that a variable can only have one value becomes a clique representing the fact that a value can only be assigned to one variable, and v.v. Since  $|V| = |D|$ , in both graphs, the cliques of both types are of the same size.

Furthermore,  $P$  and  $P'$  have the same set of solutions, and hence any edge in the microstructure complement of one also represents a forbidden assignment in the other. (An assignment is forbidden either because it violates a constraint or it represents assigning more than one value to the same variable.) Hence, the same graph can serve as the microstructure complement of both. We need only reinterpret the label of a node as a value-variable pair, rather than a variable-value pair, as can be seen in Figure 1.

#### 4 Microstructure Complement for the Magic Square Problem

Suppose that we want to represent the problem of finding an  $n \times n$  magic square as a CSP, and start by building the microstructure complement. The magic square is to consist of the numbers  $1, 2, \dots, n^2$ , laid out in a  $n \times n$  grid, in such way that the rows, the columns and

the two main diagonals all have the same sum,  $s$ . (The magic square constant  $s$  can easily be calculated to be  $n(n^2 + 1)/2$ .)

In this case, each vertex in the microstructure complement represents a combination of a cell in the square and one of the possible numbers that can go in the cell. There are  $n^4$  vertices in the graph. We can refer to a vertex as a pair  $\langle (i, j), k \rangle$  where  $(i, j)$  means the cell in row  $i$ , column  $j$ , and  $k$  is one of the integers in the range 1 to  $n^2$ . The vertex represents placing the integer  $k$  into cell  $(i, j)$ .

As usual, we have the binary nogoods, discussed in the last section, to ensure that each cell contains a different number and each number goes in a different cell. The remaining nogoods are  $n$ -ary. A hyperedge in the microstructure complement representing an  $n$ -ary nogood joins the vertices  $\langle (i_1, j_1), k_1 \rangle, \langle (i_2, j_2), k_2 \rangle, \langle (i_3, j_3), k_3 \rangle, \dots, \langle (i_n, j_n), k_n \rangle$ , where  $(i_1, j_1), (i_2, j_2), (i_3, j_3), \dots, (i_n, j_n)$  are the cells in a row, column or one of the main diagonals, and  $k_1 + k_2 + \dots + k_n \neq s$  and  $k_1, k_2, \dots, k_n$  are all distinct. That is, any set of vertices representing  $n$  collinear cells and  $n$  distinct integers whose sum is not  $s$  represents a set of assignments that cannot be allowed.

There are  $2n + 2$  sets of  $n$  collinear cells ( $n$  rows,  $n$  columns and two diagonals). Let  $N_n$  be the number of sets of  $n$  distinct integers in the range 1 to  $n^2$  whose sum is not  $s$ . There are  $\binom{n^2}{n}$  sets of  $n$  distinct integers in the range 1 to  $n^2$  and at most  $\binom{n^2}{n-1}$  whose sum is  $s$  (since if we choose  $n - 1$  distinct integers, there is at most one other integer which will make their sum equal to  $s$ ). Hence,  $N_n \geq \binom{n^2}{n} - \binom{n^2}{n-1}$ , and increases at least exponentially with  $n$ . Since the order of variables and values in an assignment is significant, any list of  $n$  collinear cells can be combined with any permutation of any  $n$  distinct integers whose sum is not  $s$  to give an assignment that cannot be allowed. Hence, the number of hyperedges, representing  $n$ -ary nogoods, is  $n!(2n + 2)N_n$ .

From their common microstructure complement, we can construct either  $P$  or  $P'$ , choosing the variables to represent either the cells or the integers, respectively. In  $P$ , suppose  $c_{ij}$  is the variable representing the cell  $(i, j)$  and its value is the integer in that cell; in  $P'$ , suppose  $l_k$  is the variable representing the integer  $k$  and its value is the cell in which that integer is placed. (In the latter case, the value of a variable is an ordered pair of integers, although in practice it would likely be represented as an integer in the range 1 to  $n^2$ .) The constraints of each CSP can be constructed by collecting together nogoods referring to the same set of variables. For instance, if  $n = 3$  then in  $P$ , where the variables are the cells, all hyperedges joining vertices  $\langle (1, 1), p \rangle, \langle (1, 2), q \rangle, \langle (1, 3), r \rangle$  give the nogoods of the constraint with scope  $c_{11}, c_{12}, c_{13}$  imposed on the top row of the magic square (that the values placed in that row must have sum  $s$ ). In  $P'$ , where the variables represent the integers, all the hyperedges joining vertices  $\langle (i_1, j_1), 1 \rangle, \langle (i_2, j_2), 2 \rangle, \langle (i_3, j_3), 3 \rangle$  give a constraint with scope  $l_1, l_2, l_3$  (that the integers 1, 2, 3 cannot form a row, column or diagonal of the square).

In each CSP, the  $n!N_n$  hyperedges are partitioned to give the constraints. The two CSPs have very different numbers of constraints, apart from the allDifferent constraints in each CSP. In  $P$ , there are  $2n + 2$  constraints, one for each row, column and main diagonal. In  $P'$ , the number of constraints is  $N_n$ , the number of sets of  $n$  distinct integers in the range 1,  $\dots, n^2$  whose sum is not  $s$ , which as before is at least  $\binom{n^2}{n} - \binom{n^2}{n-1}$ .

In  $P$ , each of the row, column and diagonal constraints has the same constraint relation. For instance, when  $n = 3$ , the set of allowed tuples consists of  $\langle 1, 5, 9 \rangle, \langle 1, 6, 8 \rangle, \langle 2, 4, 9 \rangle, \langle 2, 5, 8 \rangle, \langle 2, 6, 7 \rangle, \langle 3, 4, 8 \rangle, \langle 3, 5, 7 \rangle, \langle 4, 5, 6 \rangle$  and all their permutations. Each constraint can alternatively be represented intensionally, for instance by the conjunction of  $c_{11} + c_{12} + c_{13} = 15$  and allDifferent( $c_{11}, c_{12}, c_{13}$ ). In practice, the conjunction would probably be decomposed into a sum constraint on each row, column or diagonal, and the allDifferent constraint on all the variables. The resulting model is probably the most natural and obvious way to model the magic square problem as a CSP.

In  $P'$ , again each  $n$ -ary constraint has the same relation; in this case there are  $(2n + 2)n!$  forbidden tuples, representing the cells in a row, column or diagonal and all their

permutations. The constraints could be represented intensionally in this case, too, though less easily than in  $P$ . (For instance, since the value of a variable is an ordered pair of integers, the constraint relation could be that the first elements are not all equal *and* the second elements are not all equal *and* the differences between the first and second elements are not all 0 *and* the sums of the first and second elements are not all  $n + 1$ .)

In each case, the  $n$ -ary constraints partition the hyperedges of the microstructure complement into equal sized sets. In  $P$ , there are  $2n + 2$  sets of size  $n!N_n$ ; in  $P'$ , there are  $N_n$  sets, each of size  $n!(2n + 2)$ .

$P'$  is a much more cumbersome formulation of the problem than  $P$ , because the number of constraints is at least exponential in  $n$ , whereas in  $P$  it is linear in  $n$ . Even for  $n = 3$ ,  $P'$  has 78 constraints (since there are 84 ways to choose 3 distinct integers from 1 to 9, and only 8 of these have sum 15), whereas  $P$  has 8 constraints. When  $n = 4$ ,  $P$  has 10 constraints, and  $P'$  has more than 3,000.

It is true that if we were to represent the constraints extensionally, the constraint relation in  $P$  would rapidly become unmanageable as  $n$  increases, since it has  $n!N_n$  nogoods; representing it by the number of *allowed* tuples would not help either. However, representing the constraint relation in  $P'$  extensionally would also be difficult, because the number of forbidden tuples in that case also has an  $n!$  factor. However, the constraints in both viewpoints can be represented intensionally, as already discussed, and representation as a set of linear equations would be the natural way to represent the constraints in  $P$ .

Overall, the disparity in the number of constraints gives a compelling reason to prefer  $P$  to  $P'$ .

## 5 Constraint Propagation

Even if the constraints could be expressed in both viewpoints  $P$  and  $P'$ , the constraints in  $P$  may allow more domain pruning following constraint propagation during search than those in  $P'$ . Suppose that equivalent single-variable assignments are made in each CSP, say  $c_{11} = 9$  and  $l_9 = (1, 1)$ , in the  $3 \times 3$  case. Suppose that in  $P$ , the constraints are written as linear equations, for instance  $c_{11} + c_{12} + c_{13} = 15$ . This is not a complete representation of the constraint on  $c_{11}$ ,  $c_{12}$  and  $c_{13}$ , because there is also an allDifferent constraint on all the variables, although only correct solutions to  $P$  will be found. Moreover, constraint solvers will maintain bounds consistency (BC) on a sum constraint, rather than generalized arc consistency (GAC), so that in  $P$ , so that not all possible domain pruning will be achieved. When the assignment  $c_{11} = 9$  is made, restoring bounds consistency will result in the domains of  $c_{12}$  and  $c_{13}$  being reduced to  $\{1..5\}$ . Decomposing the constraint on  $c_{11}$ ,  $c_{12}$  and  $c_{13}$  into the equation and the allDifferent constraint loses some domain pruning in this instance: enforcing GAC on the complete constraint would also remove the value 3 from both domains, since the constraint would not allow the tuple  $\langle 9, 3, 3 \rangle$ .

In  $P'$ , on the other hand, enforcing GAC following the assignment  $l_9 = (1, 1)$  results in no domain pruning at all (except from the allDifferent constraint). In fact, there can be no domain pruning from the  $n$ -ary constraints until  $n - 1$  variables have been assigned values representing collinear cells.

Following the assignment  $l_9 = (1, 1)$ , we would like the constraints to be able to remove the values representing squares in the same row, column or diagonal from the domains of the variables  $l_6$  to  $l_8$ , to give (at least) as much domain pruning as the constraints in  $P$ .

The extra domain pruning could be achieved by introducing additional constraints into  $P'$ . In the case  $n = 3$ , the additional constraints would be binary. One such binary constraint would have scope  $\{l_6, l_9\}$ , with forbidden tuples consisting of every set of pairs of collinear cells, to express the fact that the integers 6 and 9 cannot be in the same row, column or diagonal, because there is no third integer in the allowed range that would give a total of 15. We might also add constraints on pairs of variables such as  $l_1$  and  $l_2$  that

correspond to pairs of distinct integers whose sum is not large enough to make 15 with any eligible third integer. When  $n = 3$ , we would need to add 8 binary constraints in total.

In general, we would need to introduce constraints of arity at least  $m$  where the sum of  $m$  different numbers in the range 1 to  $m^2$  can be at least  $n(n^2+1)/2 - (n-m)(n-m+1)/2$ . So when  $n = 5$ , the magic square constant is 65 and there are sets of 3 integers in the range 1 to 25 whose sum is more than  $(65 - 3)$ . For example, 25, 24 and 14 sum to 63 and so cannot be part of a collinear set of 5 distinct integers between 1 and 25 whose sum is 65. Similarly, there are sets of three integers, such as 1, 2 and 11, whose sum is less than 16 and so cannot form a set of 5 distinct integers between 1 and 25 whose sum is 65. We should also need to identify sets of 4 integers that cannot be extended feasibly to give a sum of 65, and introduce the corresponding 4-ary constraints.

Propagating these constraints should then give the equivalent of GAC on the  $n$ -ary constraints in  $P$ , i.e. would be stronger than BC on the linear equations. Of course, the cost of achieving more domain pruning is the overhead involved in propagating the extra constraints, in addition to the infeasibly large number of constraints in  $P'$  to start with.

## 6 Conclusions

Considering dual formulations of the magic square problem in detail leads to some general conclusions about permutation problems, as well as some specific observations about the dual viewpoints in this instance.

Dual formulations of permutation problems have effectively the same microstructure complement, if the allDifferent constraint in each formulation is represented as a set of binary  $\neq$  constraints. Each vertex in the (hyper)graph represents an ordered pair of problem entities that can be interpreted as a variable-value pair in one viewpoint and a value-variable pair in the other. In addition to the cliques of  $\neq$  constraints in the microstructure complement, representing that in either CSP no variable can have two different values and no value can be assigned to two different variables, there are (hyper)edges representing the permutations that are not allowed in the specific problem being modelled.

Given the microstructure complement and one of the two possible sets of variables, the constraints can be found by collecting together all hyperedges relating to the same subset of variables. Each collection of hyperedges gives a constraint whose scope is the variable subset and whose relation is given by extracting the tuples of values from the hyperedges. The two dual viewpoints each define a different partition of the hyperedges of the microstructure complement; within a partition, each subset of hyperedges corresponds to a constraint in that viewpoint. In the magic squares problem, each viewpoint partitions the hyperedges into equal-sized subsets; in other words, in each viewpoint the constraint relation is the same for all constraints, apart from the allDifferent constraint. However, this is not true for every permutation problem.

The numbers of constraints in the two dual CSPs representing a permutation problem can be very different; in the magic squares problem, the number of constraints in one CSP is linear in the size of the problem, whereas in the other it is at least exponential. This gives a very clear reason to choose one CSP model of the problem rather than its dual; the number of constraints makes it impracticable to use the model in which the variables represent the integers 1 to  $n^2$  and the values represent the cells of the square. In the  $n$ -queens problem, on the other hand, the dual models have the same number of constraints. In one viewpoint, the variables represent the rows of the chessboard and the value of a variable represents the column in which the queen in that row is placed; in the dual viewpoint, the roles of the rows and columns are interchanged. In that case, the dual viewpoints give equivalent CSPs, with nothing to choose between them.

It is well known that in representing a permutation problem as a CSP there are two dual viewpoints. It is somewhat surprising that in the magic squares problem, while one viewpoint leads to a well-behaved CSP that has a small number of constraints with a simple

intensional form, the other is not a practical choice except for very small problems, because of the number of constraints required. This demonstrates that although in many cases it is difficult to decide between viewpoints without an empirical investigation, sometimes the choice is clear.

## References

1. A. Beacham, X. Chen, J. Sillito, and P. van Beek. Constraint programming lessons learned from crossword puzzles. In *Proceedings of the 14th Canadian Conference on Artificial Intelligence*, pages 78–87, 2001.
2. E. C. Freuder. Eliminating Interchangeable Values in Constraint Satisfaction Problems. In *Proceedings AAAI'91*, volume 1, pages 227–233, 1991.
3. P. A. Geelen. Dual Viewpoint Heuristics for Binary Constraint Satisfaction Problems. In B. Neumann, editor, *Proceedings ECAI'92*, pages 31–35, 1992.
4. P. Jégou. Decomposition of Domains Based on the Micro-Structure of Finite Constraint-Satisfaction Problems. In *Proceedings AAAI'93*, pages 731–736, 1993.
5. Y. C. Law and J. H. M. Lee. Model Induction: a New Source of CSP Model Redundancy. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-2002)*, pages 54–60, 2002.