Models and Symmetry Breaking for 'Peaceable Armies of Queens'

Barbara M. Smith[1], Karen E. Petrie[1], and Ian P. Gent[2]

[1] School of Computing & Mathematics, University of Huddersfield, Huddersfield, West Yorkshire HD1 3DH, U.K.
b.m.smith/k.e.petrie@hud.ac.uk
[2] School of Computer Science, University of St. Andrews, St Andrews, Fife KY16 9SS, U.K.
ipg@dcs.st-and.ac.uk

Abstract. We discuss a difficult optimization problem on a chess-board, requiring equal numbers of black and white queens to be placed on the board so that the white queens cannot attack the black queens. We show how the symmetry of the problem can be straightforwardly eliminated using SBDS, allowing a set of non-isomorphic optimal solutions to be found. We present three different ways of modelling the problem in constraint programming, starting from a basic model. An improvement on this model reduces the number of constraints in the problem by introducing ancillary variables representing the lines on the board. The third model is based on the insight that only the white queens need be placed, so long as there are sufficient unattacked squares to accommodate the black queens. We also discuss variable ordering heuristics: we present a heuristic which finds optimal solutions very quickly but is poor at proving optimality, and the opposite heuristic for which the reverse is true. We suggest that in designing heuristics for optimization problems, the different requirements of the two tasks (finding an optimal solution and proving optimality) should be taken into account.

1 Introduction

Robert Bosch introduced the "Peaceably Coexisting Armies of Queens" problem in his column in Optima in 1999 [1]. It is a variant of a class of problems requiring pieces to be placed on a chessboard, with requirements on the number of squares that they attack: Martin Gardner [3] discusses more examples of this class. In the "Armies of Queens" problem, we are required to place two equal-sized armies of black and white queens on a chessboard so that the white queens do not attack the black queens (and necessarily v.v.) and to find the maximum size of two such armies. Bosch asked for an integer programming formulation of the problem and how many optimal solutions there would be for a standard 8×8 chessboard.

Here we discuss a range of possible models of the problem as a CSP, and show how Symmetry-Breaking During Search (SBDS) [4] can be used to eliminate the symmetry in each model, and hence find all non-isomorphic optimal solutions.

We have implemented some of the models in both ECL^{i}PS^e and ILOG Solver, so that our conclusions should be independent of any quirks of a particular constraint programming tool.

2 Basic Model

In a later issue of Optima, Bosch gives an IP formulation due to Frank Plastria. This has two binary variables for each square of the board:

$$\begin{aligned} b_{ij} &= 1 \text{ if there is a black queen on square } (i,j) \\ &= 0 \text{ otherwise} \\ w_{ij} &= 1 \text{ if there is a white queen on square } (i,j) \\ &= 0 \text{ otherwise} \end{aligned}$$

For the general case of an $n \times n$ board:

$$\begin{aligned} \text{maximize } & \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij} \\ \text{subject to } & \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} \\ & b_{i_1 j_1} + w_{i_2 j_2} \leq 1 \text{ for all } ((i_1,j_1),(i_2,j_2)) \in M \\ & b_{ij}, w_{ij} \in \{0,1\} \text{ for all } 1 \leq i,j \leq n \end{aligned}$$

where M is the the set of ordered pairs of squares that share a line (row, column or diagonal) of the board. Bosch reported that finding an optimal solution for an 8×8 board (with value 9) took just over 4 hours using CPLEX on a 200 MHz Pentium PC.

A straightforward model of the problem as a CSP is similar to this IP formulation. There is no difficulty in having variables with more than 2 values, so the number of variables can be reduced to n^2:

$$\begin{aligned} s_{ij} &= 2 \text{ if there is a white queen on square } (i,j) \\ &= 1 \text{ if there is a black queen on square } (i,j) \\ &= 0 \text{ otherwise} \end{aligned}$$

We can express the 'non-attacking' constraints as:

$$\begin{aligned} s_{i_1 j_1} &= 1 \Rightarrow s_{i_2 j_2} \neq 2 \\ \text{and } s_{i_1 j_1} &= 2 \Rightarrow s_{i_2 j_2} \neq 1 \text{ for all } ((i_1,j_1),(i_2,j_2)) \in M \end{aligned}$$

or more compactly as:

$$s_{i_1 j_1} + s_{i_2 j_2} \neq 3 \text{ for all } ((i_1, j_1), (i_2, j_2)) \in M$$

In both ECL^{i}PS^e and Solver, the single constraint gives the same number of backtracks as the two implication constraints, but is faster.

Constrained variables w, b count the number of white and black queens respectively (using the counting constraints provided in constraint programming tools such as ECL^{i}PS^e and Solver). We then have the constraint $w = b$, and the objective is to maximize w. This is achieved by adding a lower bound on w whenever a solution is found, so that future solutions must have a larger value of w; when there are no more solutions, the last one found has been proved optimal.

The model has n^2 search variables and approximately $4n^3$ binary constraints, as well as the counting constraints which have arity n^2.

3 Symmetry

The problem has the symmetry of the chessboard, as in the familiar n-queens problem; in addition, in any solution we can swap all the white queens for all the black queens, and we can combine these two kinds of symmetry. Hence the problem has 16 symmetries. It is well-known that symmetry in CSPs can result in redundant search, since subtrees may be explored which are symmetric to subtrees already explored. If only one solution is required, these difficulties do not always arise in practice. However, if a complete traversal of the search tree is required, either because there is no solution, or because all solutions are wanted, symmetry must lead to wasted search unless dealt with. This means that symmetry will cause difficulties in optimization problems, where proving optimality entails a complete search to prove that there is no better solution.

Symmetry Breaking During Search [4] is ideal for problems such as this since it only requires a simple function to describe the effect of each symmetry (other than identity) on the assignment of a value to a variable. Hence, in this case, just 15 such functions are required. Briefly, on backtracking to a choice point in the search, represented by the two constraints $var = val$ and $var \neq val$, SBDS adds a constraint to the second branch for any symmetry which has not yet been broken along the path from the root of the search tree to this node. The constraint is the symmetric equivalent of $var \neq val$ and prevents exploration of partial solutions equivalent under this symmetry to those which have already been explored following the choice $var = val$. If the effect of each individual symmetry is described, SBDS will eliminate all symmetry: all solutions produced are non-isomorphic to each other, and the search never explores any part of the search tree which is symmetric to a subtree already explored.

The seven board symmetries for which symmetry functions are required can be labelled `x`, `y`, `d1`, `d2`, `r90`, `r180` and `r270` (reflection in the horizontal, vertical and both diagonal axes, and rotations through 90°, 180° and 270°, respectively). An assignment $s_{ij} = v$ is passed to each function as a constraint, and the equivalent constraint under the relevant symmetry is returned. For instance, if the rows and columns of the board are numbered $1, .., n$, $\texttt{r90}(s_{ij} = v)$ is the constraint $s_{j,n+1-i} = v$. The symmetry which interchanges the black and white queens, bw, returns $s_{ij} = v'$, where $v' = 0$ if v =0, and otherwise $v' = 3 - v$. We also need to describe the 7 symmetries which combine a board symmetry with interchanging black and white: for instance, the symmetry $\texttt{bw} \circ \texttt{r90}$ returns $s_{j,n+1-i} = v'$.

4 Basic Model: Results

The square variables are assigned in a predefined (static) order: top row, left to right, 2nd row, left to right, and so on. To ensure that good solutions are found early, values are assigned in descending order; otherwise, the first solution found has 0 assigned to every variable, corresponding to no queens of either colour, which is valid but far from optimal. The running times given relate to a 1.6GHz Pentium 4 PC for ECL^{i}PS^e and a 600MHz Celeron PC for Solver. The implementation of SBDS in ECL^{i}PS^e is due to Warwick Harvey.

Table 1. Search effort and running time to find an optimal solution to the armies of queens problem, with no symmetry breaking. Value = optimal number of queens of each colour; F = number of fails (backtracks) to find the optimal solution; P = number of fails to prove optimality; sec. = running time in seconds

n	Value	ECL^{i}PS^e			ILOG Solver		
		F	P	sec.	F	P	sec.
2	0	7	7	0.0	7	14	0.01
3	1	6	18	0.01	6	24	0.01
4	2	0	134	0.01	0	148	0.03
5	4	25	978	0.13	30	1031	0.11
6	5	10	21469	3.2	9	24210	2.9
7	7	64	393806	78	51	435598	56
8	9	4339	10846300	3500	5270	12002608	2100

Tables 1 and 2 show that using SBDS makes a huge difference to the time required to prove optimality, although not to the time to find the optimal solution. There is a more than 10-fold reduction in the number of fails, except for the smallest values of n, though the reduction in running time is less. It would be possible to achieve some of the speed-up without SBDS, by adding constraints to the model, for instance that the top half of the board contains more white queens than the bottom, but simple constraints of this kind cannot remove all the symmetry. Table 3 compares finding all solutions with and without symmetry breaking using SBDS. It proved impracticable to find all solutions for the 8×8 board without any symmetry breaking: there are evidently hundreds of possible solutions, although only 71 are distinct.

Table 2. Search effort and running time to find an optimal solution to the armies of queens problem, with SBDS.

n	Value	ECL^{i}PS^e F	ECL^{i}PS^e P	ECL^{i}PS^e sec.	ILOG Solver F	ILOG Solver P	ILOG Solver sec.
2	0	1	1	0.0	1	2	0.01
3	1	4	6	0.01	4	11	0.01
4	2	0	15	0.04	0	16	0.02
5	4	24	96	0.16	29	131	0.04
6	5	10	1609	1.7	9	1865	0.41
7	7	64	29255	27	51	34008	7.80
8	9	4339	806056	640	5270	938652	240

Table 3. Search effort and running time to find all optimal solutions to the armies of queens problem, with ECL^{i}PS^e.

n	No symmetry breaking Solutions	No symmetry breaking F	No symmetry breaking sec.	With SBDS Solutions	With SBDS F	With SBDS sec.
2	1	0	0.0	1	0	0.0
3	16	15	0.02	1	2	0.01
4	112	219	0.05	10	18	0.06
5	18	1856	0.02	3	169	0.24
6	560	44400	5.8	35	3306	3.0
7	304	822108	130	19	59876	48
8	*not completed*			71	1604456	1130

5 Combining Squares and Lines

The basic model has a constraint between two variables if they represent squares which are on the same line (row, column or diagonal) of the board. We could consider an alternative model in which the lines are also represented by variables in the CSP. Any line must have either only white queens on it, or only black queens, or be empty, so we could create the line variables with three values corresponding to these possibilities. More compactly, we can have two possible values for each line variable: 0 means that there is no white queen on the line, and 1 means that there is no black queen on the line (unoccupied lines can have either value).

The advantage of adding the line variables is that we can reduce the number of constraints. Whenever a queen is placed on a square the values of the corresponding line variables are set accordingly. Thereafter, a queen of opposite colour cannot be placed on any of these lines, and we no longer need the constraints between square variables to enforce this.

Taking the rows as an example, we have n variables $r_1, ..., r_n$, and constraints:

$$s_{ij} = 1 \Rightarrow r_i = 0$$
$$\text{and } s_{ij} = 2 \Rightarrow r_i = 1 \text{ for all } 1 \leq i, j \leq n$$

As before, we can reduce the pair of constraints to a single constraint:

$$s_{ij} + r_i \neq 2 \text{ for all } 1 \leq i, j \leq n$$

The combined model has more variables than the basic model (another $6n$, approximately), but we can still use just the n^2 square variables as the search variables. There are approximately $4n^2$ constraints, each between a line variable and a square variable, rather than $4n^3$ constraints between pairs of square variables as before.

Adding the line variables to the model makes no difference to the number of fails in ECL^{i}PS^e (there are some differences in Solver), but reduces the running time to solve the problems optimally by about one-third in Solver and about one-sixth in ECL^{i}PS^e, for $n = 8$.

Note that since the search variables, and hence the branching decisions made during search, are unchanged, SBDS is unaffected by the change in the model.

6 Combined Model: Discussion

Figure 1 compares the constraints required in the two models: it shows the constraints required to express that row i cannot have queens of different colours, in the case $n = 5$. The solid lines show the clique of constraints required between the variables corresponding to the squares on row i in the basic model. The dotted lines are the constraints that replace them in the combined model: we have replaced an n-clique of constraints by just n.

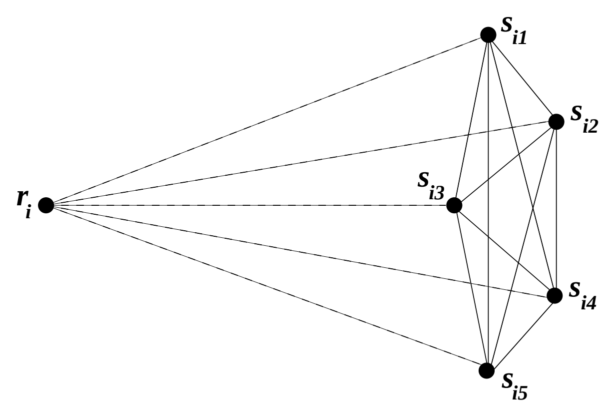

Fig. 1. Constraints in the two encodings of the 'armies of queens' problem

In addition to the constraints expressing that we cannot have queens of different colours on any line, we also need constraints on the number of queens of each colour. These would be difficult if not impossible to express solely in terms of the line variables. Hence, adding the line variables to the basic model is not exactly analogous to the idea of redundant modelling [2]: in redundant modelling, two models are combined, either of which could be used independently. Moreover, the claimed advantage of redundant modelling is that constraint propagation within either model can feed through to the other, via the channelling constraints which link them. Here, there are no constraints between the line variables, and in the combined model, the only constraints between the square variables are those counting the number of queens. We are replacing all other constraints of the basic model by the channelling constraints linking the line and square variables. This is somewhat similar to combining models of permutation problems, where the benefit comes from propagation of the channelling constraints, which in that case can replace $\neq$ constraints between the variables of the original model [5].

Although we cannot have a model with line variables alone, we could in theory have a model with both line and square variables in which we search on the line variables and not the square variables. This is an attractive idea, since there would only be $6n$ search variables, approximately, rather than n^2. However, in practice it leads to a number of difficulties. It would introduce new symmetries, since the values 0 and 1 are interchangeable if a line is unoccupied. We could avoid this by having three values rather than two for the line variables, but even then, a complete assignment to the line variables does not always uniquely determine the values of the square variables, so that not all non-isomorphic optimal solutions would be found.

7 Counting Unattacked Squares

In trying to solve the armies of queens problem by hand, it becomes apparent that we need only place the queens of one colour, say white, provided that we check as each queen is placed that the number of squares not so far attacked is at least equal to the number of white queens on the board. A black queen can be placed on any square which is not attacked by any white queen; hence if there are k white queens on the board and at least k unattacked squares, we can extend the current assignment to a complete solution with value k.

This leads to a new model of the problem. As in the basic model, there is a variable s_{ij} for each square on the board, but now with possible values 0 and 1, where 0 signifies that the square is either empty or occupied by a black queen and 1 that it contains a white queen. For each square, we also construct the set of squares, A_{ij}, that a queen placed on this square would attack. A set variable U represents the unattacked squares on the board. If $s_{ij} = 0$ but $(i, j) \notin U$, square (i, j) must be empty.

The constraints are:

$$s_{ij} = 1 \Rightarrow A_{ij} \cap U = \emptyset \quad 1 \leq i, j \leq n$$

The number of unattacked squares must be at least as great as the number of white queens, which we could express as the constraint $|U| \geq w$.

The model has n^2+1 binary constraints, as opposed to $O(n^3)$ for the squares model and about $4n^2$ for the combined model with line and square variables. The search variables are the n^2 s_{ij} variables. This is the same number as in the previous models, but now each variable has only two possible values rather than three.

The objective is again to maximize the number of white queens. However, since we now require at least as many black queens, rather than exactly the same number, the solutions found are sometimes different from the previous models. For some values of n (2, 3, 4, 6 and 8, of those considered so far), it is sometimes possible to add an extra black queen to an optimal solution with equal numbers of each. For instance, the new model finds a solution with 5 white queens and 6 black queens when $n = 6$, and only 30 solutions altogether rather than 35. It is easy to reconstruct the 35 solutions by selecting 5 out of the 6 black queens in all possible ways.

8 SBDS in the Unattacked Squares Model

The functions for the seven board symmetries are exactly as in the previous models. However, the symmetry that swaps the black and white queens, and the combination of this symmetry with the board symmetries, is less straightforward, since the black queens are not explicitly represented. Furthermore, if there are more black than white queens (i.e. $|U| > w$), the symmetry between black and white has already been broken. The assignment $s_{ij} = 1$ represents the placement of a white queen on square (i, j); its equivalent under the symmetry `bw` $\circ$ `r90` is to force the square (i, j) to be unattacked, so that it can be occupied by a black queen. The corresponding symmetry function describing the effect of `bw` $\circ$ `r90` on the assignment $s_{ij} = v$ returns the constraint that $(j, n+1-i) \in U$, if $v = 1$, and if $v = 0$ does nothing.

Hence, given an assignment $s_{ij} = v$, the symmetry function for `bw` $\circ$ `r90` should return a conditional constraint:

$$(|U| = w) \Rightarrow (j, n+1-i) \in U$$

whenever $v = 1$.

In fact, the condition is unnecessary, because if the rest of the constraint were imposed in the case when there are more black queens than white queens, the effect would be to create a solution with more white than black queens, and this is in any case forbidden by the problem constraints. The condition $|U| = w$ can therefore be dropped from the constraints returned by the symmetry functions. This has no effect on the number of fails, or the solutions found, but does reduce the running time.

Note that symmetry breaking in this way is only legitimate because we are actually trying to find solutions with equal numbers of white and black queens. For instance, when $n = 6$, the search does not find all possible distinct solutions with 5 white queens and 6 black queens, but only one of them; the others are ruled out by symmetry-breaking. Suppose that there is a symmetry equivalence class of solutions with 6 queens of one colour and 5 of the other, and suppose that one of the 6 is in a corner square. Hence, none of the 5 are in a corner square, since they are of the opposite colour and would be attacked. The search begins by placing a white queen in the top left corner square; but it cannot find a solution from this equivalence class, because there would have to be 6 white queens, which is forbidden by the constraint that there must be no more white than black queens. On backtracking, the symmetry constraints rule out placing a black queen in any corner square. Hence no solution from the equivalence class will be found. However, 6 corresponding solutions with just 5 white queens and 5 black queens will be found instead.

Table 4 shows the results for this model using ILOG Solver. An asterisk in the column showing the number of solutions found indicates that the number is less than shown in earlier tables; as already described, some of the solutions found have one more black queen than white queens, but can be converted to a set of solutions with equal numbers of each by dropping an extra black queen in all possible ways. The number of solutions found is then exactly as before.

In comparison with the previous models, the number of fails is more than halved for $n = 8$, and the running time is reduced even more, in comparison with the model combining square and line variables (the running times are about 240 sec. with the original model, 160 sec. with the combined model and 72 sec. for the unattacked squares model). The difference is still larger when $n = 9$: the combined model takes over 31 million fails and 5500 sec. to find and prove the optimal solution.

Table 4. Search effort and running time to solve the armies of queens problem optimally and find all non-isomorphic optimal solutions, with the unattacked squares model and SBDS, using Solver.

n	Value	Finding & proving optimal solution			Finding all solutions		
		F	P	sec.	Solutions	F	sec.
2	0	1	2	0.01	1	0	0.01
3	1	5	12	0.01	1	2	0.01
4	2	0	20	0.02	8*	14	0.02
5	4	35	147	0.10	3	164	0.12
6	5	10	1614	1.7	30*	3352	0.61
7	7	75	23671	3.5	19	46333	6.8
8	9	4676	478012	72	53*	960841	140
9	12	2469621	11041681	1700	18	14164002	2200

9 Variable Ordering

The unattacked squares model was derived from trying to solve the problem by hand. This also led to an algorithm for constructing a solution and from that a variable ordering heuristic. The algorithm places a white queen on the square attacking fewest squares that are not already attacked; hence, it tries to keep the number of unattacked squares as large as possible. The algorithm terminates when no more white queens can be placed without reducing the number of unattacked squares below the number of white queens. Often, the solution found is optimal or near optimal. The first white queen placed is in a corner square, and the lexicographic ordering used so far assigns the variable representing the top left corner first. However, after assigning the first variable, the lexicographic ordering diverges from the algorithm. We have therefore experimented with a dynamic variable ordering heuristic that chooses next the variable representing a square which is already attacked itself and where a white queen would attack fewest unattacked squares.

The fewest-unattacked-squares heuristic finds optimal solutions very quickly, but is worse than lexicographic ordering at proving optimality. For instance, when n=8, it finds an optimal solution immediately, with no backtracking, but then takes more than 720,000 fails and 120 sec. to prove optimality. For n=9, again, an optimal solution is found very quickly, in 330 fails, compared to nearly 2.5 million fails for lexicographic ordering; however, it takes 2500 sec. in total and over 13 million backtracks to prove optimality.

Since this heuristic is so poor at proving optimality, it seemed worthwhile to try exactly the opposite heuristic, i.e. choose the square where a white queen will attack *most* unattacked squares. Not surprisingly, this takes much longer to find the optimal solution (though not as long as lexicographic ordering for the larger values of n), but it is overall much faster than either the fewest-unattacked-squares heuristic or lexicographic ordering. The results are shown in Table 5. For 8×8 and 9×9, this heuristic runs more than 10 times faster than lexicographic ordering.

Table 5. Search effort and running time to solve the armies of queens problem optimally. (The unattacked squares model with the most-unattacked-squares heuristic and SBDS, using Solver.)

n	Value	F	P	sec.
2	0	1	2	0.01
3	1	5	12	0.01
4	2	4	20	0.02
5	4	0	42	0.03
6	5	44	628	0.13
7	7	581	3779	1.12
8	9	2953	44276	11.7
9	12	44778	374800	116
10	14	15690	5891793	2460
11	17	235248	82758262	37100

The 10 × 10 and 11 × 11 problems can now be solved, although the latter takes more than 10 hours running time. The fewest-unattacked-squares heuristic again finds optimal solutions for both these problems very quickly. We have also found all optimal solutions: in both cases, there are solutions with an extra black queen, so that the number returned has to be adjusted to give the true number of solutions with equal sized armies. After adjustment, there are 405 solutions to the 10 × 10 problem and 714 to the 11 × 11 problem.

Since the most-unattacked-squares heuristic performs so much better than either of the other variable orderings considered, and yet is not especially good at finding optimal solutions, it is worth trying to explain why it does well. Figure 2 shows the first solution found by this heuristic in solving the 11 × 11 problem, and the first optimal solution found. The first solution found has only

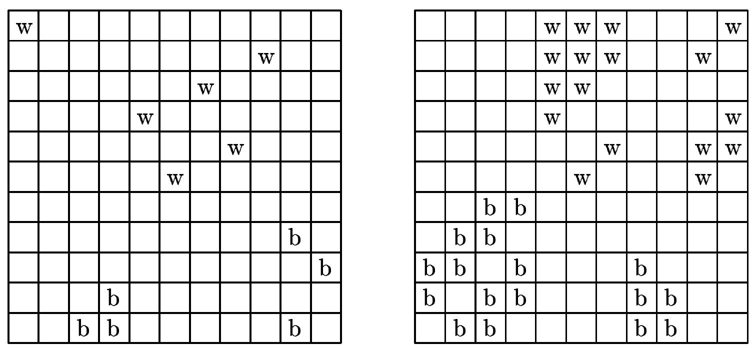

Fig. 2. Equal sized armies of queens on a 11 × 11 board. Left, 6 white queens attack all but 6 squares on the board. Right, an optimal solution with 17 queens of each colour.

6 queens of each colour, whereas the optimal solution has 17. The heuristic is biased towards producing solutions with small number of white queens: once an optimal solution has been found, all such assignments become nogoods. Hence the heuristic can prune branches of the search tree when only a few variables have been assigned. On the other hand, a heuristic which tries to place as many white queens as possible before leaving fewer than the optimal number of unattacked squares (as the fewest unattacked squares heuristic does) will tend to prune the search much lower down the tree.

Although both heuristics could be used with the earlier models, they are expensive to implement, since the information on unattacked squares is not readily available. Here, we compute $|A_{ij} \cap U|$ for each unassigned variable s_{ij}, and choose the variable for which this is smallest or largest, depending on the heuristic.

10 Discussion

The peaceable armies of queens problem is a difficult optimisation problem that was hard to solve using an integer programming model. The constraint programming models considered here have all done reasonably well in solving the 8 × 8 problem; even so, problems larger than 10 × 10 are taking a very long time to solve, even for the best model we have found. Related problems have been investigated by Velucchi [6], and the optimal values for the armies of queens problem up to 10 × 10 can be extracted from his results. This suggests that constraint programming is competitive with other methods that have been tried for this problem. However, the problem has no practical importance and it is the experience of trying to solve it that is useful, rather than the solutions themselves.

Starting from a basic constraint programming model with no symmetry breaking, we have shown that the time to solve the 8 × 8 problem can be reduced from 2100 sec. to 12 sec. (using ILOG Solver), a more than 100-fold improvement. The results are summarized in Table 6.

Table 6. Performance of different models in solving the 8 × 8 armies of queens problem.

Model	F	P	sec.
Basic model, no SBDS	5270	12002608	2100
Basic model, with SBDS	5270	938652	240
Combined model, with SBDS	5270	945247	160
Unattacked squares model, with SBDS	4676	478012	72
Unattacked squares model, with SBDS & most-unattacked-squares heuristic	2973	44276	12

A major part of the improvement is due to eliminating the symmetry using SBDS. Given an implementation of SBDS, it requires no ingenuity on the part of the user to write the 15 functions to describe the effects of the individual symmetries of the problem. For the 8 × 8 problem, eliminating the symmetry reduces the time to solve the problem optimally from 2100 sec. to 240 sec.; it also allows a set of non-isomorphic solutions to be found, whereas without symmetry breaking, it took too long to find all the possible solutions, which would in any case have been uninformative.

Further reductions in running time are due to remodelling the problem. We have described three different ways of modelling it, starting from a basic model not very different from an integer programming formulation.The combined model introduces ancillary variables (one for each row, column or diagonal) in order to reduce the number of constraints, from $4n^3$ to $4n^2$, approximately. This significantly reduces running time, although the search effort is largely unaffected.

The unattacked squares model has the same number of search variables as the other models, but with fewer possible values, so that the number of possible assignments is reduced. The model also has fewer constraints than the previous models, which probably contributes to the reduction in running time. However, the binary constraints are between an integer variable and a set variable, so that constraint propagation may be more expensive than with binary constraints involving two integer variables.

Devising new models does require ingenuity. The different models we have presented can be seen as viewing the problems at different levels. The basic model expresses that a single white queen and a single black queen are inconsistent if they are on the same row, column or diagonal. The combined model takes the perspective of a line (row, column or diagonal) of the board: any number of queens can be placed on a line provided that they are all the same colour. The unattacked squares model expresses that any number of white queens can be placed anywhere on the board, as long as there are at least as many unattacked squares as white queens. Hence, each model takes a broader view of the problem than the previous model. Moreover, whereas the first two models are only concerned with whether the white and black queens attack each other, the final model also has something of the optimization criterion built into it: not only must the white and black queens not attack each other, but there must be enough of each of them. Trying to view the problem from several different angles is likely to be a fruitful source of ideas for remodelling; we found that constructing solutions by hand facilitated this and gave useful insights into key features of the problem.

The final improvement in modelling the problem came from a variable ordering heuristic. We have presented two: one finds optimal or near-optimal solutions very quickly, but is poor at proving optimality. The other is its exact opposite and takes much longer to find an optimal solution, but then is much better at proving optimality. Although it is intuitively clear that finding optimal solutions and proving optimality are different in nature, it is surprising to see it demonstrated in such a clear-cut way. Again, the first heuristic was inspired by trying to construct solutions by hand. There may be other problems where a good heuristic for proving optimality is the exact opposite of a good heuristic for finding an optimal solution, and this will be investigated further. Variable ordering heuristics have hitherto mainly been investigated in the context of constraint satisfaction rather than optimization: our experience with this problem suggests that variable ordering heuristics for satisfaction problems and for optimization problems may need to be designed separately. For some optimization problems, it may be better to use two different heuristics, the first to find a good solution and the second to improve that solution if possible and to prove optimality.

Acknowledgments

We are extremely grateful to Warwick Harvey for his advice and help in using ECL^iPS^e, and for his implementation of SBDS. The authors are members of the APES research group (`http://www.dcs.st-and.ac.uk/~apes`) and would like to thank the other members. This work is supported by EPSRC grant GR/R29673.

References

1. R. A. Bosch. Peaceably coexisting armies of queens. *Optima (Newsletter of the Mathematical Programming Society)*, 62:6–9, 1999.
2. 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.
3. M. Gardner. Chess queens and maximum unattacked cells. *Math Horizon*, pages 12–16, November 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. B. M. Smith. Dual Models of Permutation Problems. In *Proceedings of CP'01: the 7th International Conference on Principles and Practice of Constraint Programming*, LNCS 2239, pages 615–619. Springer, 2001.
6. M. Velucchi. For me, this is the best chess-puzzle: Non-dominating queens problem. `http://anduin.eldar.org/~problemi/papers.html`. Accessed May 2002.