Some New Results for the Queens Domination Problem

P.B. Gibbons and J.A. Webb
Department of Computer Science
University of Auckland
Private Bag 92019
Auckland
New Zealand

Abstract

Computing techniques are described which have resulted in the establishment of new results for the queens domination problem. In particular it is shown that the minimum cardinalities of independent sets of dominating queens for chessboards of size 14, 15, and 16 are 8, 9, and 9 respectively, and that the minimum cardinalities of sets of dominating queens for chessboards of size 29, 41, 45, and 57 are 15, 21, 23 and 29 respectively. As a by-product the numbers of non-isomorphic ways of covering a chessboard of size n with k independent queens for $1 \leq n \leq 15$ and $1 \leq k \leq 8$, as well as the case $n = 16$, $k = 8$, are computed.

Key words

Queens domination problem, dominating set, backtracking, simulated annealing.

1. Introduction

Chessboard problems have been of interest to combinatorialists and puzzle-solvers for centuries. For example, Rouse Ball [3] attributes a problem involving the movement of knights on a chessboard to Guarini in 1512. In fact, the range of problems and extent of research in this area has been quite massive, as described in Rouse Ball [3] and Gardner [10].

A particular class of chessboard problems involves determining the minimum number of chess pieces required to attack all squares of a chessboard, sometimes with additional constraints. In this paper we ask the following questions:

(1) What is the minimum number of queens that can be placed on an $n \times n$ chessboard so as to ensure that each square is attacked by at least one queen? This minimum is called the *domination number* $\gamma(n)$.

(2) What is the minimum number of mutually non-attacking queens that can be placed on an $n \times n$ chessboard so as to ensure that each square is attacked by at least one queen? This minimum is called the *independent domination number $i(n)$*.

While the answers to these questions are currently not known for all values of n, a variety of upper and lower bounds on these numbers have been established, and exact values found for a range of small values of n. In this paper we describe exhaustive and non-exhaustive search techniques that have been used to determine $\gamma(n)$ and $i(n)$ for small values of n. A number of new results are produced from this work. The methods also have the potential to solve many other varieties of chessboard problems.

2. Definitions and previous results

A generalised $n \times n$ chessboard consists of n^2 cells arranged in n rows and n columns. The queen is a chess piece that can move any number of squares in a horizontal, vertical or diagonal direction. A square to which a queen may move is said to be attacked by that queen.

Covering problems involving queens on a chessboard are related to covering problems in graphs in the following way. From an $n \times n$ chessboard form the queens graph Q_n with n^2 vertices corresponding to the chessboard cells and with two vertices adjacent if and only if the cell corresponding to one of the vertices is attacked by a queen placed on the cell corresponding to the other vertex. (Note: similar graphs may be defined for other chess pieces.) Given a graph $G = (V, E)$ a set of vertices D in V is a dominating set if every vertex in $V \setminus D$ is adjacent to at least one vertex in D. The domination number of G, denoted by $\gamma(G)$, is the minimum cardinality of a dominating set in G. A set of vertices S in V is independent if no two vertices in S are adjacent to each other. The independent domination number of G, denoted by $i(G)$, is the minimum cardinality of an independent dominating set in G.

Determining $\gamma(G)$ and $i(G)$ in general is NP-Hard. Determining the answers to questions (1) and (2) in section 1 corresponds to determining $\gamma(G)$ and $i(n)$ for the special class of queens graphs.

The first explicit statement of the queens domination problem (QDP) was due to Abbe Durand in 1861. This work was followed closely by De Jaenisch [7] and the first eight values of $\gamma(Q_n)$ were reported by Rouse Ball [3]. Although there is currently no mathematical proof that these values are correct, they have been verified by computer. Ahrens [2] has discussed the three problems of (1) finding $\gamma(Q_n)$, (2) finding $i(Q_n)$, and (3) finding $\gamma(Q_n)$ but with the additional constraint that every queen must be attacked by at least one other queen. He found that for values of n not exceeding 12, 11, and 9 respectively, the answers to all three questions lie in the interval $[(n-1)/2, (n+2)/2]$.

Until quite recently no non-trivial upper and lower bounds for the QDP were known. Welch (private communication to Cockayne [6]) was the first to derive an upper bound for QDP. This is shown in the following theorem, the proof of which can be found in Cockayne [6]:

Theorem 2.1 (Welch)

Let $n = 3q+r$ where $0 \leq r < 3$. Then $\gamma(Q_n) \leq 2q+r$.

From this it follows that $\gamma(Q_n) \leq 2n/3 + r/3$ where $r = n \bmod 3$. Spencer (private communication to Cockayne [6]) has shown that $\gamma(Q_n) \geq (n-1)/2$, thereby establishing, for all n, the lower bound observed by Ahrens [2] for small n.

A large number of refinements have followed, most of which were aimed at special cases of n (cf. Grinstead, Hahne and Van Stone [12], and Fricke et al [9]). Perhaps the most significant of these refinements is for the special case of $n = 4k+1$. Here the following lower bound has been established by Weakley [16]:

Theorem 2.2 (Weakley)

$\gamma(Q_{4k+1}) \geq 2k + 1$ for all $k \geq 0$.

Weakley also exhibited cases where $\gamma(Q_{4k+1}) = 2k + 1$ for $k = 3, 4, 5, 6,$ and 8. This was extended to include the cases for $k = 9, 12, 13,$ and 15 when Burger, Mynhardt and Cockayne [5] constructed symmetric dominating sets. The value $\gamma(Q_{18}) = 9$ was determined by Alice McRae [9] who used a genetic algorithm to exhibit a covering of an 18×18 chessboard by 9 queens, thereby achieving the lower bound of Spencer.

The current best upper bound for $\gamma(Q_n)$ has been established by Burger, Cockayne and Mynhardt[4]:

Theorem 2.3 (Burger, Cockayne and Mynhardt)

$\gamma(Q_n) \leq 31n/54 + O(1)$.

There are also bounds for the independent queens domination problem (IQDP). Clearly $\gamma(Q_n) \leq i(Q_n)$. In addition the first eight values of $i(Q_n)$ were attributed by Ball [3] to De Jaenisch [7]. While there is currently no mathematical proof these values have been confirmed by computer. The values of $i(Q_n)$ for $1 \le n \le 25$ have been investigated by Weakley [16].

The first upper bound for $i(Q_n)$ was given by Cockayne [6]:

Theorem 2.4 (Spencer and Cockayne)

$$i(Q_n) \le 0.705n + 0.895.$$

This was refined by Grinstead, Hahne and Van Stone [12] to:

Theorem 2.5 (Grinstead, Hahne and Van Stone)

$$i(Q_n) \le 2n/3 + O(1).$$

A summary of the current information we have on the values of $\gamma(Q_n)$ and $i(Q_n)$ for small values of n is given in Table 2.1. A summary of what is currently known about $\gamma(Q_{4k+1})$ is given in Table 2.2.

n	1	2	3	4	5	6	7	8	9	10	11	12	13
$\gamma(Q_n)$	1	1	1	2	3	3	4	5	5	5	5	6	7
$i(Q_n)$	1	1	1	3	3	4	4	5	5	5	5	7	7

n	14	15	16	17	18	19	20	21	22	23	24	25
$\gamma(Q_n)$	≤8	≤9	≤9	9	9	≤10	≤11	11	≤12	≤13	≤13	13
$i(Q_n)$	≤8	≤9	≤9	9	≤10	≤11	≤11	11	≤13	≤13	≤13	13

Table 2.1

k	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
n	9	13	17	21	25	29	33	37	41	45	49	53	57	61	65
$\gamma(Q_n)$	5	7	9	11	13	≤ 17	17	19	≤ 23	≤ 25	25	27	29	31	≤ 35

Table 2.2

It can be seen from Table 2.1 that there are only three values of n (viz. 4, 6 and 12) for which it is known that $\gamma(Q_n) < i(Q_n)$. Fricke et al [9] have made the following conjecture:

Conjecture 2.5 (Fricke et al)

For n sufficiently large, $\gamma(Q_n) = i(Q_n)$.

3. Non-existence by exhaustive search

Table 2.1 indicates that the first unknown value of $i(Q_n)$ is $n = 14$ where $i(Q_{14}) = 7$ or 8. To settle this case we need to either exhibit an independent dominating set of cardinality 7, or show that one does not exist. This can be achieved by an exhaustive backtrack search.

Backtracking is one of the oldest methods for exhaustive search and has been well described in the literature. A summary of the backtracking method as applied to the construction of combinatorial structures can be found in Gibbons [11]. Key features affecting the efficiency of a backtrack search are the early detection of partial solutions which cannot be extended to a complete solution, and the recognition of partial solutions that are equivalent (or isomorphic) to partial solutions generated earlier in the search. We describe the feasibility and isomorph rejection checks that can be used in a backtrack search for the independent queens domination problem.

We begin by considering the well-known n-queens problem: "Can n mutually non-attacking queens be placed on an $n \times n$ chessboard?". Since a valid solution must contain exactly one queen in each column, we can represent a solution as a vector $(x_1, \ldots, x_{n-1}, x_n) \in \{1 \ldots n\}^n$ where x_i is the row number of the queen in the *ith* column. We define a set $\{f_1, \ldots, f_{n-1}, f_n\}$ of feasibility functions as follows:

$\forall k \in \{1, \ldots, n-1, n\}$, $f_k(x_1, \ldots, x_{k-1}, x_k) =$ true
  iff $\forall\, 0 < i < j < k+1 \quad x_i \neq x_j$ (no queens in same row) **and**
  $|x_i - x_j| \neq |i-j|$ (no queens in same diagonal)

or $f_1(x_1) =$ true
For $k>1$, $f_k(x_1, \ldots, x_{k-1}, x_k) = f_{k-1}(x_1, \ldots, x_{k-2}, x_{k-1})$
  and $\forall\, i < k$, $x_i \neq x_k$ (no queen in same row as *kth* queen)
  and $\forall\, i < k$, $|x_i - x_k| \neq |i-k|$ (no queen in same diagonal as *kth* queen)

The above formulation of the feasibility functions requires a search to determine the value of $f_k(x_1, \ldots, x_{k-1}, x_k)$. This can be avoided by maintaining the following data structures:

array *row[1 .. n]*: *row[i]* = true iff no queen in row i
array *down[1-n .. n-1]*: *down[i-j]* = true iff no queen in down diagonal $i-j$
array *up[2..2n]*: *up[i+j]* = true iff no queen in up diagonal $i+j$

We can now define the feasibility functions as:

$f_1(x_1)$ = true
For $k>1$, $f_k(x_1, ..., x_{k-1},x_k) = f_{k-1}(x_1, ..., x_{k-2},x_{k-1})$
and $row[x_k]$ **and** $down[x_k - k]$ **and** $up[x_k + k]$

Fitting these definitions into the general backtracking algorithm framework outlined in Gibbons [11], we extend a partial solution $(x_1, ..., x_{k-1},x_k)$ $(k<n)$ only if $f_k(x_1, ..., x_{k-1},x_k)$ = true. Since we are interested only in the existence of a non-attacking arrangement of n queens, we would continue the search until either a solution is obtained, or until the search is complete.

Now suppose we wish to generate *all* possible solutions to the n-queens problem. Among all such solutions, many will be equivalent in the sense that they can be generated from others by a series of rotations and/or reflections. A simpler, but equivalent, task therefore would be to generate just a complete set of non-equivalent solutions. This will allow us to incorporate some powerful isomorph rejection techniques along the way. Suppose we have constructed a partial solution, and find that it is equivalent to a partial solution that has already been generated earlier in the search. Then there is no need to further consider the current partial solution, since it, and all its extensions, are equivalent to partial solutions that have already been fully considered. That is, we can reject the current partial solution, pruning it and all its extensions from the search tree.

To recognise equivalent partial solutions we need to specify precisely the equivalence operations acting on the chessboard. Assume that the chessboard squares are labelled (i,j), $i=0,1,...,n-1$, $j=0,1,...,n-1$, with i being the row index and j being the column index. The top left square is labelled $(0,0)$ and the bottom right square labelled $(n-1,n-1)$ (as in Figure 4.1). Then the equivalence operations are as follows:

Operation	**Notation**	**Square mapping**
Identity	I	$(i,j) \rightarrow (i,j)$
Rotation anticlockwise by 90	C_1	$(i,j) \rightarrow (n-1-j,\ i)$
Rotation anticlockwise by 180	C_2	$(i,j) \rightarrow (n-1-i,\ n-1-j)$
Rotation anticlockwise by 270	C_3	$(i,j) \rightarrow (j,\ n-1-i)$
Reflection about up diagonal	D_1	$(i,j) \rightarrow (n-1-j,\ n-1-i)$
Reflection about down diagonal	D_2	$(i,j) \rightarrow (j,i)$
Vertical reflection	R_1	$(i,j) \rightarrow (n-1-i,j)$
Horizontal reflection	R_1	$(i,j) \rightarrow (i,n-1-j)$

This set of operations forms a group (the dihedral group) under the action of composition. We say that configurations A and B are equivalent if B can be obtained from A by a sequence of operations from the above set.

Actions in the group are applied as partial solutions are constructed, and can be used to reject partial solutions that are equivalent to other partial solutions that have already been considered earlier in the search. Since we are generating partial solutions systematically in increasing lexicographical order, a partial solution is identified as having been generated earlier in the search if it is lexicographically less than the current partial solution. Note that the technique of isomorph rejection is also useful in an existence search, since equivalent non-solutions are culled out also in the search process.

Equivalence checking can be added to the feasibility check for a partial solution. If this is done to the n-queens enumeration algorithm then only non-equivalent solutions are produced by the algorithm.

We now return to the original problem of covering an $n \times n$ chessboard by a set of k $(\leq n)$ independent queens. A solution to this problem can be obtained by modifying the above algorithm for the n-queens problem. A solution to the k independent queens problem will necessarily have one queen in each of k columns. We can therefore proceed to enumerate each of the ${}^{n}C_{k}$ k-column subsets in increasing lexicographical order, and for each k-subset perform a backtrack search for an arrangement of k independent queens placed in the k columns so as to cover all squares of the chessboard. The same feasibility checks can be applied as with the n-queens algorithm, with the additional check that an arrangement of k-queens covers all squares. Isomorph rejection is again a powerful tool, but needs to be used with some care.

Suppose we are backtracking to find all arrangements of k queens using the column subset $(c_1, c_2, \dots , c_k)$ $(c_1<c_2< \dots < c_k)$, and have generated a partial feasible solution P involving $h \leq k$ queens placed in columns $c_1, c_2, \dots , c_h$. For each of the 7 non-identity transformations obtain an equivalent partial solution P' involving the h columns $d_1, d_2, \dots , d_h$, $d_1<d_2< \dots < d_h$. We carry out the following isomorph rejection checks on P':

(i) If $(d_1, d_2, \dots , d_h) < (c_1, c_2, \dots , c_k)$ reject P as it and all extensions to it are equivalent to partial solutions considered during the enumeration of an earlier k-subset of columns.

(ii) If $(d_1, d_2, \dots , d_h) > (c_1, c_2, \dots , c_k)$ we cannot reject P at this stage as P' possibly belongs to a future k-subset of columns. We must continue to extend P.

(iii) If $(d_1, d_2, \dots, d_h) = (c_1, c_2, \dots, c_k)$ in general we cannot reject P as the extension of P may produce an equivalent partial solution involving a k-subset of columns not yet investigated. However, under special circumstances we can consider a rejection. If we know that under the equivalence operation currently being used any extension of P uses a set of columns lexicographically $\leq (c_1, c_2, \dots, c_k)$. then we can reject P if P' is lexicographically less than P, since P and all extensions to it are equivalent to partial solutions considered either earlier in the search of the current k-subset of columns, or in the search of an earlier k-subset of columns. We know that the column subset will be preserved under the vertical reflection R_1, and we know that the column subset will not increase if $c_h \geq n-1$. There are other conditions where this constraint is true (for example involving the horizontal reflection R_2) but we did not consider it cost effective to carry out all these checks.

The algorithm was implemented in the language C on a DEC ALPHA 3000/600 workstation, producing the results in Table 3.1. New results from Table 3.1 are that $i(Q_{14}) = 8$, $i(Q_{15}) = 9$, and $i(Q_{16}) = 9$. A side-effect of the searches is the enumeration of all non-equivalent ways of covering a chessboard of size n with k independent queens for $1 \leq n \leq 15$ and $1 \leq k \leq 8$, as well as for $n = 16$, $k = 8$.

4. Existence by probabilistic search

The previous section deals with a situation where we show that an upper bound of u queens is tight by establishing the non-existence of a solution with $u-1$ queens. In this section we show that a lower bound of l queens is tight by producing a solution with l queens. To cope with larger cases than the ones investigated in section 3 where we are confident that a solution exists we use a probabilistic (non-exhaustive) approach.

Various probabilistic techniques have been successful in solving structural existence problems of this type (for a summary see Gibbons [11]). One method which been particularly popular is that of simulated annealing (Aarts and Korst [1], Metropolis et al [15], van Laarhoven [14], and Kirkpatrick, Gelatt Jr. and Vecchi [13]). While a complete description is given in Gibbons [11] we provide a brief summary here.

The search operates within a set Σ of feasible solutions. A cost $c(i)$ is associated with each $i \in \Sigma$, and the task is to find an optimal solution with overall minimum cost. For each $i \in \Sigma$ we define a set T_i of transformations each of which can be used to change i into a closely related feasible solution j. The set of solutions that can be reached from i by applying a transformation from T_i is called the neighbourhood $N(i)$ of i. The simulated annealing algorithm is related to that of hill-climbing. Starting with a randomly chosen current feasible

n	k	# of non-equivalent solutions	Time
1	1	1	-
2	1	1	-
3	1	1	-
4	2	0	-
4	3	2	-
5	2	0	-
5	3	2	-
6	3	0	-
6	4	17	-
7	3	0	-
7	4	1	-
8	4	0	-
8	5	91	0.1 sec
9	4	0	0.1 sec
9	5	16	0.4 sec
10	4	0	0.4 sec
10	5	1	1.6 secs
11	4	0	1.0 secs
11	5	1	5.9 secs
12	6	0	1.4 mins
12	7	105	3.7 mins
13	6	0	5.3 mins
13	7	4	19.6 mins
14	7	0	1.5 hours
14	8	55	4.5 hours
15	8	0	23.5 hours
15	9	1314	58.4 hours
16	8	0	113.7 hours

Table 3.1

solution i we generate a random $j \in N(i)$ and compute the cost difference $\Delta_{ij} = c(j) - c(i)$. If $\Delta_{ij} \leq 0$ we move to j as the current feasible solution. If $\Delta_{ij} > 0$ we accept j as the current feasible solution with probability $e^{-\Delta_{ij}/T}$ where T is a control parameter known as the temperature. This acceptance rule is known as the Metropolis criterion. Initially the control parameter is given a large value and a sequence (or Markov chain) of trials is generated. After each such sequence, the control parameter is lowered. After an appropriate stopping condition is met, the final configuration is taken as an approximation to the optimal solution.

Suppose T_k is the value of the control parameter and L_k the length of the Markov chain generated at the *kth* iteration of the simulated annealing algorithm. Then a pseudo-Pascal description of the simulated annealing algorithm is as follows:

```
Set $T_1$, $L_1$ and $\alpha$;  $k:=1$;   Generate an initial feasible solution $i$;
**repeat**
        **repeat** $L_k$ times
        **begin**
                Randomly generate $j$ from $N(i)$;
                $\Delta_{ij} = c(j) - c(i)$;
                **if** $\Delta_{ij} \leq 0$ **or** $random[0,1) < e^{-\Delta_{ij}/T_k}$ **then** $i := j$;
        **end**;
        $T_{k+1} := \alpha T_k$;
        Set $L_{k+1}$;
        $k := k+1$;
**until** stop condition is met
```

In the above algorithm *random[0, 1)* returns a random number generated from a uniform distribution on the interval *[0, 1)*.

The performance of the algorithm will be influenced by the following factors:

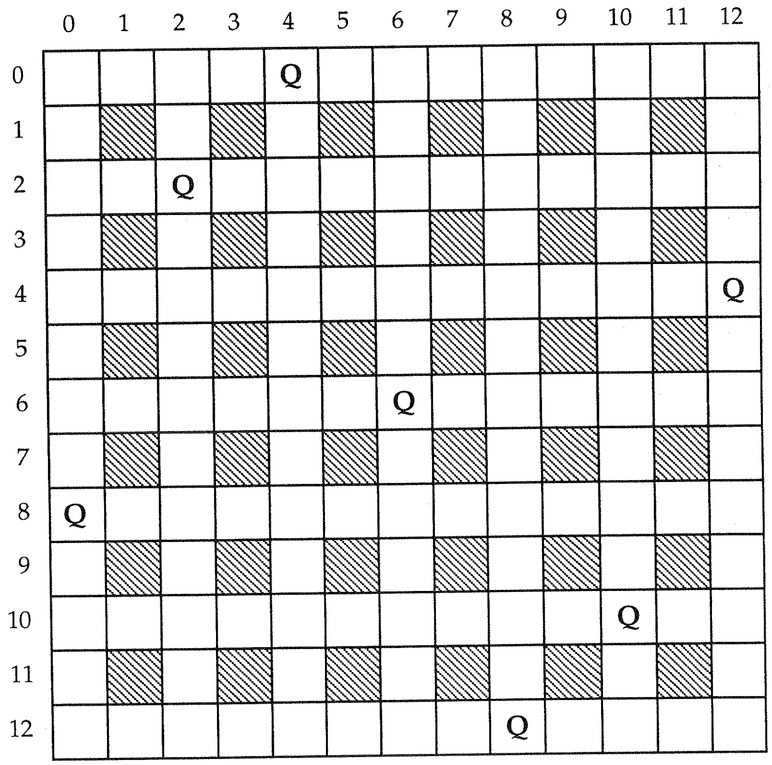
(i) The value of T_1.
(ii) The value of the control decrement α.
(iii) The Markov chain length L_k for each k.
(iv) The choice of stopping condition.

The specification of these parameters constitutes a cooling schedule.

This algorithm will now be applied to the production of dominating sets of not necessarily independent queens for the case $n = 4k+1$. Here Weakley [16] has shown that $\gamma(Q_{4k+1}) \geq 2k+1$ for all integer values k. So if we can build a dominating set of $2k+1$ queens we will establish that $\gamma(Q_{4k+1}) = 2k+1$.

For example, with $k = 3$ a dominating set of size 7 is shown in Figure 4.1. On this chessboard the rows and columns have been numbered from 0 to 12, and so each square can be classified as odd-odd, odd-even, even-odd and even-even according to the parity of its corresponding row-column components. Notice that the dominating set in the above case has the property that there is exactly one queen on each of the $2k+1$ even rows and columns. So to check that the set does in fact dominate the board we need only check that each of the (shaded) odd-odd squares is dominated diagonally by a queen. If we restrict our attention to this type of solution then we can compress the even rows and columns to lines and represent the covering of Figure 4.1 as shown in Figure 4.2.

In this representation queens are restricted to the $2k+1$ horizontal and vertical lines, and must dominate diagonally all the $2k \times 2k$ squares. Such an arrangement can be represented by a vector $S = (x_0, x_1, \ldots, x_{2k})$ where the queen on vertical line i is placed on horizontal line x_i. This representation allows the formulation of a simulated annealing search for a dominating set. Note that the non-existence of such a solution does not necessarily preclude the existence of a dominating set of size $2k+1$ of another form.



▨ odd-odd square Q occupied square

Figure 4.1 Covering of $4k+1 \times 4k+1$ board where $k = 3$

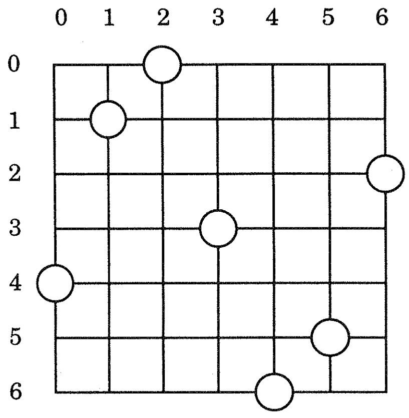


Figure 4.2 Compressed representation of covering of Figure 4.1

We begin by randomly placing *2k+1* queens on the *2k+1* horizontal and vertical lines. The cost of this solution is the number of uncovered squares. We then randomly choose two queens at positions (x_1, y_1) and (x_2, y_2) and investigate replacing them at positions (x_2, y_1) and (x_1, y_2). This interchange preserves coverage of all horizontal and vertical lines. The new configuration is accepted according to the Metropolis criterion. An example of such an interchange is shown in Figure 4.3:

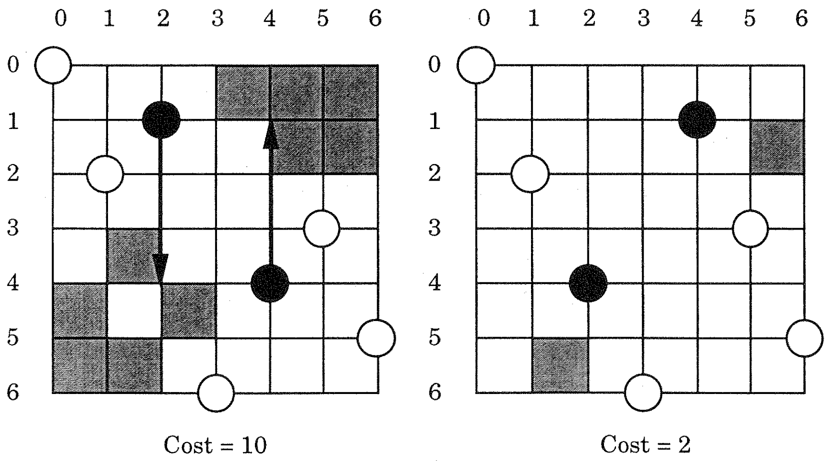


Figure 4.3 A pairwise queen exchange, before and after

For our stopping condition we set a limit of the number of consecutive Markov chains in which the maximum and minimum cost for each Markov chain remains unchanged. This limit is termed the freezing factor.

Determining the optimal values of parameters in the cooling schedule is crucial to the development of an effective algorithm. While some theoretical guidelines are available (cf Aarts and Korst [1]), in practice these values must be set empirically by experimentation (cf Elliott and Gibbons [8]). For this application we used the following ranges of values in the cooling schedule:

Initial temperature	1.0 - 5.0
Control decrement α	.995 - .9995
Markov chain length	1000 - 5000
Freezing factor	12

Table 4.1

One final modification must be mentioned. In many optimisation problems we are happy to accept a good approximation to an optimal solution. In this case however an approximate covering is of no use - we are only interested in an arrangement which has cost *0* and thus covers all squares on the board. The simulated annealing algorithm must therefore be repeatedly run until an optimal solution is found.

Using the simulated annealing algorithm the new results in Table 4.2 were obtained. Each solution is represented by a vector $(x_0, x_1, \ldots, x_{2k})$ where the queen on vertical line i is placed on horizontal line x_i.

k	n	Dominating set of size 2k+1
7	29	(10,3,6,11,14,1,5,13,9,7,2,4,12,0,8)
10	41	(14,11,6,19,16,1,7,9,2,12,18,8,4,13,20,3,0,15,17,5,10)
11	45	(14,5,2,13,20,7,15,3,0,10,22,19,11,12,8,21,4,1,16,9,6,17,18)
14	57	(18,2,22,9,6,27,10,7,0,25,24,14,17,23,26,12,15,3,8,13,28,1,20,5,19,21,4,11,16)

Table 4.2

Typical run times to generate these solutions on a DEC ALPHA 3000/600 workstation are displayed in Table 4.3.

k	n	Run time
7	29	10 secs
10	41	1 minute
11	45	50 minutes
14	57	83 minutes

Table 4.3

In these solutions the placement of queens on Q_{4k+1} with exactly one queen on each even row and each even column is similar to that of Burger, Mynhardt and Cockayne [5], but differs by not requiring a queen on the central square, and by not requiring that the queens placement be symmetric in any way. This relaxation was essential in finding solutions, since it has been shown by exhaustive search in [5] that no such (symmetric) solutions exist for $k = 7, 10, 11,$ and 14.

5. Conclusions

The application of dominating queens has provided a very nice example of when exhaustive and non-exhaustive search techniques are useful.

Using an exhaustive approach we were able to settle and confirm values of $i(Q_n)$ for $1 \leq n \leq 16$. A side-effect was to count all possible non-equivalent ways of covering a chessboard of size n with k independent queens for $1 \leq n \leq 15$ and $1 \leq k \leq 8$, as well as for $n = 16$, $k = 8$. Using the dihedral group of rotations and reflections acting on the chessboard is essential in obtaining a workable isomorph rejection strategy. As we were able to use the whole group in this case we were able to implement a total isomorph rejection strategy, so that only non-isomorphic solutions were generated. In many cases involving backtracking the groups are very large so that only a subgroup can be used during the construction. While this often cuts down the search time substantially, the output solutions must later be checked for possible isomorphisms.

With the dominating queens problem the unknown cases were much larger. Here we were confident of achieving a solution corresponding to a lower bound and were able to do so using a (probabilistic) simulated annealing approach. For the case $n = 4k+1$ we were able to determine $\gamma(Q_n)$ for four new values of n, thereby confirming that $\gamma(Q_{4k+1}) = 2k+1$ for $2 \leq k \leq 15$. This provides additional evidence to support a possible conjecture that $\gamma(Q_{4k+1}) = 2k+1$ for all $k \geq 2$.

The techniques described in this paper could easily be applied to many other chessboard problems.

6. Acknowledgements

The authors wish to thank Ernie Cockayne, Christine Mynhardt and Doug Weakley for helpful discussions relating to this research. They are also grateful to an anonymous referee for pointing out many improvements and corrections to a draft of this paper.

References

[1] Emile Aarts, Jan Korst, Simulated annealing and Boltzmann machines - A stochastic approach to combinatorial optimisation and neural computing, (Wiley, 1989).

[2] W. Ahrens, Mathematische Unterhaltungen und Spiele, Liepzig-Berlin, 1910.

[3] W.W. Rouse Ball, Mathematical Recreations and Problems of Past and Present Times, Macmillan, London (1982).

[4] A.P. Burger, C.M. Mynhardt and E.J. Cockayne, Domination and Irredundance in the Queen's Graph, Discrete Math., to appear.

[5] A.P. Burger, C.M. Mynhardt and E.J. Cockayne, Domination Numbers for the Queen's Graph, Bulletin of the ICA 10 (1994), 73 - 82.

[6] E.J. Cockayne, Chessboard Domination Problems, Discrete Math. 86 (1) (1990), 13-20.

[7] C.F. De Jaenisch, Applications de l'Analyse Mathematique an Jenudes Echecs, Petrograd, 1862.

[8] J.R. Elliott and P.B. Gibbons, The Construction of Subsquare Free Latin Squares by Simulated Annealing, Aust. J. of Combinatorics 5 (1992), 209-228.

[9] G.H. Fricke, S.M. Hedetniemi, S.T. Hedetniemi, A.A. McRae, C.K. Wallis, M.S. Jacobson, H.W. Martin, and W.D. Weakley, Combinatorial Problems on Chessboards: A Brief Survey, in "Graph Theory, Combinatorics and Applications", Proc. Seventh Quadrennial Conf. on the Theory and Applications of Graphs, Western Michigan University, Alavi & Schwenk (Eds.), Wiley (1995).

[10] M. Gardner, The Unexpected Hanging and Other Mathematical Diversions, Simon and Schuster, New York (1969).

[11] P.B. Gibbons, Computational Methods in Design Theory, Chapter 9, Section VI, in The CRC Handbook of Combinatorial Designs, C.J. Colbourn and J.H. Dinitz (eds.), CRC Press (1996).

[12] C.M. Grinstead, B. Hahne and D. Van Stone, On the Queen Domination Problem, Discrete Math. 86 (1991), 21-26.

[13] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimisation by simulated annealing, Science 220 (1983), 671-680.

[14] P.J.M. van Laarhoven, Theoretical and computational aspects of simulated annealing, Erasmus University, Rotterdam, Ph.D. thesis (available as a CWI Tract, 1988).

[15] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Physics 21 (1953), 1087-1092.

[16] W.D. Weakley, Domination in the Queen's Graph, in "Graph Theory, Combinatorics and Applications", Proc. Seventh Quadrennial Conf. on the Theory and Applications of Graphs, Western Michigan University, Alavi & Schwenk (Eds.), Wiley (1995).

(Received 29/4/96; revised 19/7/96)