

The n-Queens Problem

Craig Letavec and John Ruggiero
 Department of Economics and Finance
 School of Business Administration
 300 College Park
 University of Dayton
 Dayton, OH 45469-2240, USA

craig.letavec@notes.udayton.edu
 ruggiero@udayton.edu

The n -queens problem, originally introduced in 1850 by Carl Gauss, may be stated as follows: find a placement of n queens on an $n \times n$ chessboard, such that no one queen can be taken by any other. While it has been well known that the solution to the n -queens problem is n , numerous solutions have been published since the original problem was proposed. Many of these solutions rely on providing a specific formula for placing queens or transposing smaller solutions sets to provide solutions for larger values of n (Bernhardsson, 1991 and Hoffman *et al.*, 1969).

Empirical observations of smaller-size problems show that the number of solutions increases exponentially with increasing n (Sosi and Gu, 1994). Alternatively, search-based algorithms have been developed. For example, a backtracking search, will systematically generate all possible solution sets for a given $n \times n$ board (Bitner and Reingold, 1975 and Purdom and Brown, 1983). In practice, however, backtracking approaches provide a very limited class of solutions for large size boards because it is difficult for a backtracking search to find solutions that are significantly distinct in the solution space (Sosi and Gu, 1994). Several authors have proposed other efficient search techniques to overcome this problem. These methods include search heuristic methods (Kale, 1990) and local search and conflict minimization techniques (Sosi and Gu, 1994). Recently, advances in research in the area of neural networks have led to several papers proposing solutions to the n -queens problem via neural networks (Shagrir, 1992 and Mandziuk, 1995). Specifically, the use of Hopfield networks has been applied to the n -queens

problem by Mandziuk (1995). The Hopfield neural network is a simple artificial network which is able to store certain patterns in a manner similar to the brain in that the full pattern for a given problem can be recovered if the network is presented with only partial information. The ability of neural networks to adapt and learn from information has applications in optimization problems beyond the n -queens problem.

Finally, the problem has been stated as an integer programming similar to the assignment problem. See Foulds and Johnston (1984) for numerous chess-related programming models including a discussion of the n -queens problem. In this column, we provide an Excel-based application (see <http://131.238.53.216/eco493/chess/chess5.xls>) with visual basic to allow users to solve the problem for any chessboard of size 4 through size 10. The spreadsheet allows visualization of the queens with formatted cells producing a chessboard.

The n-QueensS Mathematical Program

Following Hoffman, *et al.* (1969), we define the chessboard as an $n \times n$ matrix of square elements for $n \geq 4$; each square is identified as an ordered pair (i,j) , where i and j are the row and column numbers of the square, respectively. Given each ordered pairs, we can identify:

1. Row k ($k = 1, \dots, n$) consists of all ordered pairs (k,j) $j = 1, \dots, n$.
2. Row k ($k = 1, \dots, n$) consists of all ordered pairs (k,j) $j = 1, \dots, n$.
3. Minor diagonal k ($k = 2, \dots, 2n$) consists of all ordered pairs (i,j) such that $i + j = k$.
4. Major diagonal k ($k = 1-n, \dots, n-1$) consists of all ordered pairs (i,j) such that $i - j = k$.

For a given chess board of size n , let $d_{ij} = 1$ if a queen occupies (i,j) and 0 otherwise $i, j = 1, \dots, n$. Then a solution to the n -queens problem is obtained from the following mathematical program:

$$\begin{aligned}
 & \text{Max} \sum_{i=1}^n \sum_{j=1}^n d_{ij} \\
 & \sum_{j=1}^n d_{ij} \leq 1 \quad \forall i = 1, \dots, n \\
 & \sum_{i=1}^n d_{ij} \leq 1 \quad \forall j = 1, \dots, n \\
 & \sum_{i=1}^n \sum_{j=1}^n d_{ij} \leq 1 \quad \forall k = 2, \dots, 2n \\
 & \sum_{i=1}^n \sum_{j=1}^n d_{ij} \leq 1 \quad \forall k = 1-n, \dots, n-1 \\
 & d_{ij} \in (0,1) \quad \forall i, j = 1, \dots, n
 \end{aligned}$$

The objective function has no practical use since it

is known a priori that the value is n for $n \geq 4$. In essence, this program uses the mathematical programming search routine to find a solution. Note that additional constraints can be added to force a queen onto a particular square (ordered pair). In this case, the objective function may not achieve a value of n . Recognizing that the decision variables are binary, the first constraint ensures that at most one queen will be placed in each row. The second constraint ensures that at most one queen will be placed in each column. We note that the first two constraints will hold with equality for $n \geq 4$ since the solution is n . The third and fourth constraints force at most one queen on the diagonals.

Solution for $n = 10$

A solution for a chessboard of size $n = 10$ obtained from the spreadsheet is shown in the following diagram:

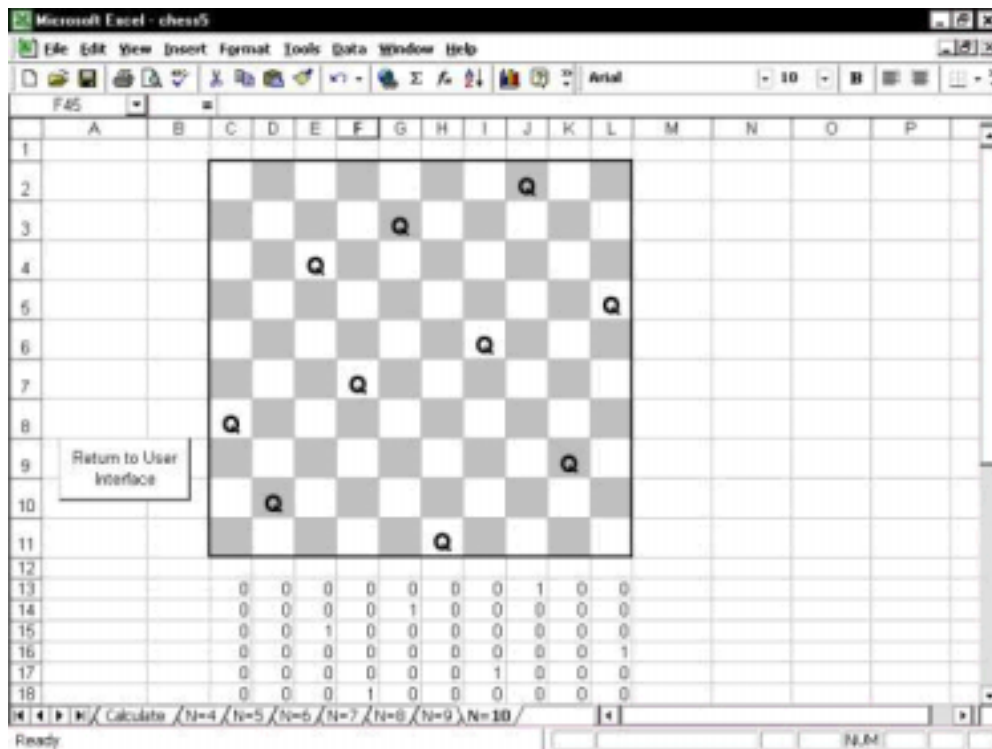


Figure 2

References

- Bernhardsson, B. (1991), "Explicit solutions to the n-queens problems for all n," *ACM SIGART Bulletin*, Vol. 2, No. 7.
- Bitner, J.R. and E.M. Reingold (1975), "Backtracking programming techniques," *Communications of the ACM*, Vol. 18, No. 11, pp. 651-56.
- Foulds, L.R. and D.G. Johnston (1984), "An application of graph theory and integer programming: Chessboard non-attacking puzzles," *Mathematics Magazine*, Vol. 57, No. 2, pp.95-104.
- Hoffman, E.J., J.C. Loessi, and R.C. Moore (1969), "Constructions for the solution of the n queens problem," *Mathematics Magazine*, pp.66-72.
- Kale, L.V. (1990), "An almost perfect heuristic for the n non-attacking queens problem," *Information Processing Letter*, Vol.34, pp.173-78.
- Mandziuk, J. (1995), "Solving the n-queens problem with a binary Hopfield-type network. Synchronous and asynchronous model," *Biological Cybernetics*, Vol. 72, No. 5, pp. 439-46.
- Purdom, P.W. and C.A. Brown (1983), "An analysis of backtracking with search rearrangement," *SIAM Journal of Computing*, Vol. 12, No. 4, pp. 717-33.
- Shagrir, O. (1992), "A neural net with self-inhibiting units for the n-queens problem," *International Journal of Neural Systems*, Vol. 3, No. 3, pp. 249-52.
- Sosi, R. and J. Gu (1994), "Efficient local search with conflict minimization: A case study of the n-queens problem," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, No. 5, pp. 61-68.