An evolution search algorithm for solving *N*-queen problems

Bah-Hwee Gwee and Meng-Hiot Lim

QUERY

1. Please provide Keywords and Biographical notes.

An evolution search algorithm for solving *N*-queen problems

Bah-Hwee Gwee and Meng-Hiot Lim
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

Abstract: This article explores evolution search algorithm for solving the N-queens problem. It will be shown how simple mechanisms of selection, reproduction and mutation can be effective in solving the N-queens problem. Simulation of the search algorithm for N up to 2000 has been achieved on a personal computer. The algorithm is robust and is capable of exploring multiple solutions to the N-queens problem. Solutions beyond the first solution uncovered are achieved without significant additional overhead.

Reference to this article should be made as follows: Gwee, B.H. and Lim, M.H. (2003) 'An evolution search algorithm for solving *N*-queen problems', *International Journal of Computer Applications in Technology*, Vol. 0, No. 0, pp. 1–6.

1 INTRODUCTION

Evolution search algorithms (ESA) have long been adopted to solve a large number of NP-hard problems [1–3]. For example, Voigt [4] made use of an evolution algorithm, replicator network algorithm, to solve the traveling salesman and module placement problem with good results. The search space of the NP-hard optimization problems tends to increase in an exponential manner as the size of the problem increases. One particular example of such a problem is the N-queens optimization problem, which has found many useful practical applications including VLSI routing and testing, communication computing and networking.

The 8-queen problem was proposed in 1848 and investigated by mathematicians including C.F. Gauss in 1850. The idea is to locate 8 queens on an 8 × 8 chessboard in mutually non-attacking positions. Since then, it has been used as a benchmark problem to demonstrate divide-and-conquer methods [5], load balancing algorithms [6], constraint theory [7] and dynamic programming [8]. Parallel algorithms have also been proposed to solve the general *N*-queen problem. In his reported work, Takefuji [9] obtained good results using Hopfield neural network based on a modified hysterisis McCulloch-Pitts transfer function to obtain an optimum solution. Others, experimented with hysterisis neural network [10] and chaotic neural network with self-feedback controlled to escape from a local minimal point [11] on soling up to 200-queen problems.

In general, for any *N*-queen problem of $N > 3$, there are more than one solution that exist. A brute force sequential search with one queen per row but without restricting the configuration to one queen per column will suggest a search space of N^N. However, the search space can be greatly reduced to *N* factorials if we restrict the search to configurations that satisfy the condition that one and only one queen is to be located per column. For instance, if *N* is 70, the reduced search space is close to 12×10^{99}, which is over the limit of what a normal hand held calculator can handle. Table 1 shows the number of possible solutions for problem size of 4 to 12 queens, which can be determined using exhaustive search method. It becomes obvious that as the value of *N* increases, we should expect the number of possible solutions to increase drastically.

At first glance of Table 1, it may project an impression of easing the search. This is clearly not true since as *N* increases, the search space also increases exponentially. A more representative measure to reflect the increasing

Table 1 *Number of solutions in different sizes of N-queen problems.*

N (problem size)	*Number of solutions*	*Search space (N!)*	*Solution density (per 10^6)*
4	2	24	83,333
5	10	120	83,333
6	4	720	5,555
7	40	5,040	7,936
8	92	40,320	2,281
9	352	362,880	970
10	724	3,628,800	199
11	2,680	39,916,800	67
12	14,032	479,001,600	29

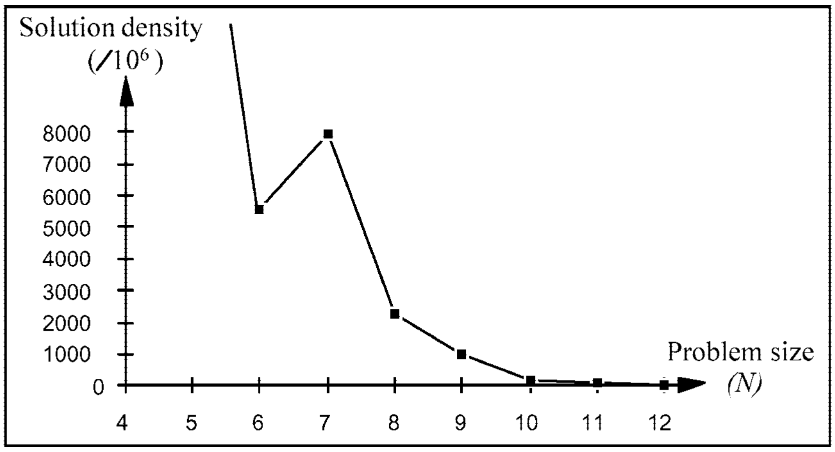

Figure 1 *Relationship between the solution density and the problem size.*

difficulty of the problem as N increases is the *solution density* – the number of solutions with respect to the search space.

The relationship between the solution density and the problem size is shown graphically in Figure 1. It shows clearly that the solution density (per million), decreases significantly as N increases. From the figure shown, if N is 7 then the chances of hitting a solution is close to 8000 for every million trials, assuming a truly random search approach. As N increases to 12, the chances of finding an optimum solution reduce to 29 for every million trials. Hence, it becomes obvious that the degree of difficulty of the search increases with N. It becomes apparent that any algorithm that relies on random walk through the search space can be easily incapacitated as the size of the problem increases.

2 CODING AND FITNESS MEASURE

A typical N-queen configuration can be represented as an N-permutation string structure of cardinality N. Each integer (1 to N) represents the column position of a queen on the chessboard. Using such a representation scheme, the equations to describe the constraints of the N non-attacking queen's configuration are

$$q_i \neq q_j \quad \text{for } i,j = 1 \text{ to } N;\ i \neq j \tag{1}$$

$$|q_i - q_j| \neq |i - j| \quad \text{for } i,j = 1 \text{ to } N;\ i \neq j \tag{2}$$

$$q_1 + q_2 + \cdots + q_N = N(N+1)/2 \tag{3}$$

Equation (1) states that no two queens should occupy the same column while Equation (2) states that no two queens can be placed along the same diagonal. The inclusion of Equation (3) together with Equation (1) will pose the restriction for q_i to take on values from 1 to N only. Based on this formulation, each chromosome or string can represent one possible configuration of the placement of queens. The alleles of the genes are denoted by integers from the set $\{1,2, \ldots, N\}$, which are interpreted as column positions of the queens. The position of each gene within the chromosome indicates the corresponding row that a queen is located.

For an evolution search algorithm to work there is a need for a *fitness function,* which will serve as a measure of goodness of the "quality" of the gene and the chromosome. Besides serving as indicator to signal when an optimum solution is reached, the fitness of the genes serves as an important criterion for selection and reproduction. For any N-queen configuration that is not optimal, there are at least two queens that have been placed in mutually attacking positions, which are referred to as conflicts. A quantitative measure of the quality of a chromosome for the N-queen problem can thus be obtained by considering the queens that are in conflict. Therefore, the fitness function of an N-queen configuration is

$$F(Q_i) = F(q_{i1}) + F(q_{i2}) + \cdots + F(q_{iN}) \tag{4}$$

where,

$$F(q_{ij}) = \begin{cases} 0 & \text{if } q_{ij} = q_{ik} \text{ and} \\ & \quad |q_{ij} - q_{ik}| = |j - k| \\ & \quad \text{for } k = 1 \text{ to } N;\ k \neq j \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

Equation (4) states that the fitness of a chromosome is obtained by summing the fitness of all its component genes. The fitness of each gene is dependent on the positions of all the other queens. A gene is considered fit if it is not in conflict with any other queen as described by Equation (5). From a mathematical view point, the goal of the search is therefore to maximize the $F(Q_i)$ function.

3 THE ALGORITHM

The basic mechanisms at work in the algorithm are selection, reproduction and mutation. This section outlines in details the evolution search algorithm for solving the *N*-queen problem. The general search algorithm as shown below consists of two stages:

```
Procedure: N-queen search
    t = 0
    Generate initial population
Stage 1{selection and reproduction}
    begin
        evaluate $F_t(Q)$;
        sort according to fitness;
        for (i = 1 to p/2) do
        $Q_{(p/2)+i} = R_1(Q_j)$;
        t = t + 1;
    end;
Stage 2 {selection and mutation}
    while (t < generation_limit)
    begin
        for (i = 1 to p) do
        begin
            evaluate $F_t(Q_i)$;
            if $F_t(Q_i) < N$
                identify weak genes of $Q_i$;
                mutate with probability m;
            if $F_t(Q_i) = N$
                $Q_i$ is optimum solution!
        end;
    t = t + 1;
    end;
```

In Stage 1, the fitter half of the population survives into the next generation. Those that survive reproduce offspring to replace the weaker half of the population. The single parent reproduction operator R_1 used in this algorithm can be written as:

$$R_1(A) = \uparrow(a_1, a_2, \ldots, a_N) \tag{6}$$

where A is a chromosome, a_i is the allele of gene i and "↑" is a shift operation. A suitable form of shift operation that has been used effectively is the shift-by-one operation. For example, if $A = \{1,3,5,2,4)$ which denotes a 5-queen configuration, $R_1(A)$ becomes (2,4,1,3,5). The main idea of the $R_1(A)$ is to generate a clone that has similar structure to A. The main goal of Stage 1 of the algorithm is to sieve out weak structures so that the search may focus more on chromosomes that are relatively fit.

At Stage 2 of the algorithm, chromosomes are reproduced and modified using mutation type operations. As an illustration, consider a chromosome representing the configuration of 8 queens:

$$A = (\underline{3},7,\underline{5},1,\underline{6},\underline{4},\underline{8},\underline{2}) \tag{7}$$

From the condition laid out by Equation (2), it can be deduced that the first, third, fifth, sixth, seventh and the eighth queens are in conflict. This is indicated in Equation (7) by the underscored alleles of *A,* which has a fitness *F(A*) equal to 2. Making use of the similarity template [12] as a notation to classify similarity in features, we can write one of the representative schemata of *A* to be:

$$H_A = (*,7,*,1,*,*,*,*) \tag{8}$$

where * = {2,3,4,5,6,8}. We will refer to such a schema as a class of *permutation constrained schemata.* If the entire search space is considered as a hyperplane, then H_A can be viewed as the sub-hyperplane where one or more of the optimum solutions lied within it. The permutation constrained schema of Equation (8) is representative of 720 chromosomes with the defined features of $q_2 = 7$ and $q_4 = 1$. While fit genes survive and weak genes will be removed during mutation, the probability of *A* remaining unchanged into the next generation will therefore be 1/720. Hence it is clear from here that the fitter the chromosome, the greater is the probability of it retaining most of its structural features into the next generation with little or no alterations. Furthermore, by retaining features that contribute positively to its fitness, the tendency for parents to reproduce fitter offspring is enhanced.

To further explain the evolutionary mechanisms, we make use of two examples in the following discussion. Consider the case where $A = (\underline{7},1,3,5,8,\underline{2},4,6)$. Figure 2(a) illustrates why chromosome *A* is considered stuck at a false peak. Notice from Figure 2(a) that q_1 and q_6 are in mutually attacking positions. The fitness of chromosome *A* is 6. Reproduction and mutation may yield $(\underline{2},\underline{1},3,\underline{5},\underline{8},\underline{7},4,6)$, which will introduce additional conflicts and causes the fitness to drop to 3. The chromosome *A* is then required to start from the point where the fitness is 3 and will evolve progressively from then on. However, there is also a 0.5 probability that the chromosome survives into the next generation without any alteration.

Let us consider another case where $B = (\underline{1},3,5,7,2,4,6,\underline{8})$ as shown in Figure 2(b). The chromosome is considered trapped at a false peak because the reproduction and mutation plan will naturally force the chromosome to dive

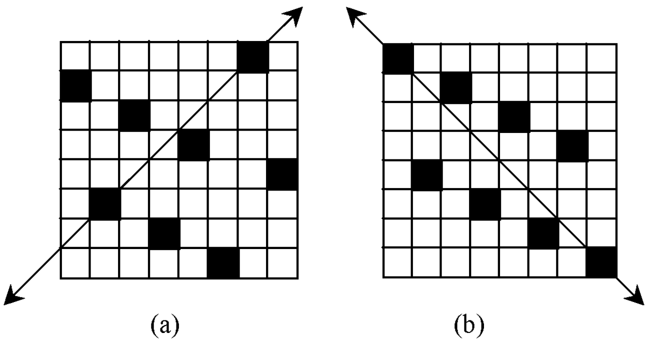

Figure 2 *Illustration of local maxima 8-queen configuration.*

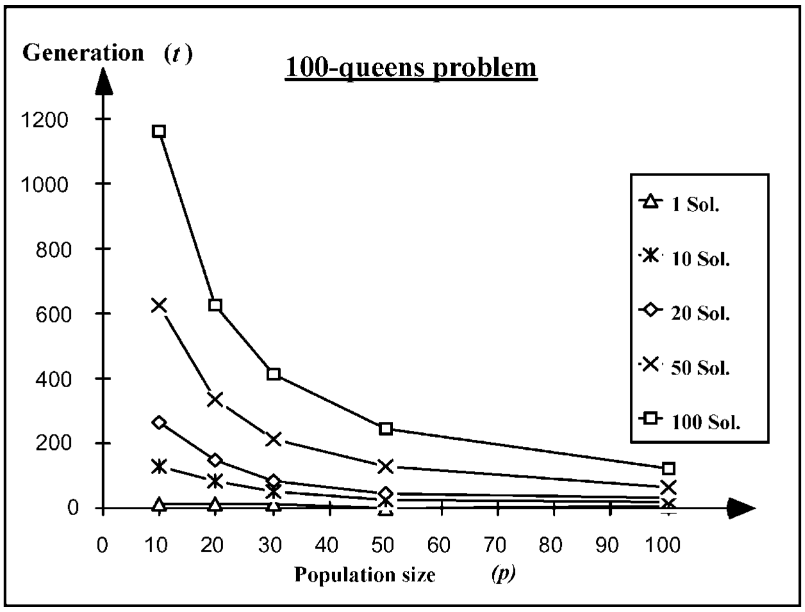

Figure 3 *The number of generations for obtaining solutions of 100-queens problem based on different population sizes.*

into another local maxima (8,3,5,7,2,4,6,1) where the first and the last gene are in conflict with each other. The chromosome will continue to oscillate between the 2 states if no external disturbances are added. It can be noted from Figure 2(b) that the two conflicting queens are located at the corner and interchange their position will not have any improvement. In order to solve this problem, external disturbance is introduced to shake the oscillatory structure. This is done by randomly selecting one or more genes to be labeled as *pseudo weak genes*. For example, the fourth gene of the chromosome (1,3,5,7,2,4,6,8) is labeled as a weak gene even though it is not in conflict with any other queens. With the disturbance caused by the pseudo weak gene, the mutation operator will have more choices to relocate the conflicting queens to better position during the reproduction process. The structure of chromosome that is stuck at local maxima will then be broken up. It will then evolve towards the near optimal region.

4 SIMULATION RESULTS

Results of our simulation have shown that we are able to obtain a solution to any number of queens subject to the limitation of computer memory. On a personal computer, we were able to simulate an evolutionary search for up to 2000 queens. In all cases that we tried, the first solution was obtainable in less than 30 generations for any problem size. Intuitively, one can assume that with bigger population, the greater is the likelihood of uncovering solutions to the problem. This can easily be confirmed with experiments to show that by increasing the population size, the number of generations required to reach an optimum solution will also be reduced. For this purpose, the results obtained from simulating a 100-queen problem are plotted in Figure 3, which shows the number of generations required to attain 1, 10, 50 and 100 solutions. The average number of generations required (based on five random trials) of obtaining 1, 10, 20, 50 and 100 solutions are shown using different population size.

As shown in Figure 3, we notice the effect of population size becoming more apparent for obtaining large number of different optimum solutions where the performance of the search improves significantly with bigger population size. This can be attributed to the overhead incurred in setting up the search. A reasonable explanation for the observation is that from the initial population, it is necessary to orientate and focus the search in the proper direction. This is consistent with graphical plots, which show that the time required achieving the first optimum solution remains relatively constant. Thus can be attributed to the fix overhead incurred in setting up the search.

A problem size of 1000 has been simulated and the results obtained based on the average of five different random trials are plotted in Figure 4, which shows the number of generations, *t*, required to obtain up to 12 different optimum solutions. From the plot in Figure 4, it can be seen that after the occurrence of the first optimum solution, the time cost of finding subsequent solutions reduces. This further substantiates the point made earlier that after the initial setting up time, the efficiency of the search improves for greater depth of search.

For purposes of illustration, Figure 5 depicts an optimum solution of a 100-queen problem solved using the reported algorithm. The corresponding chromosome representing this solution is genetically coded as follows:

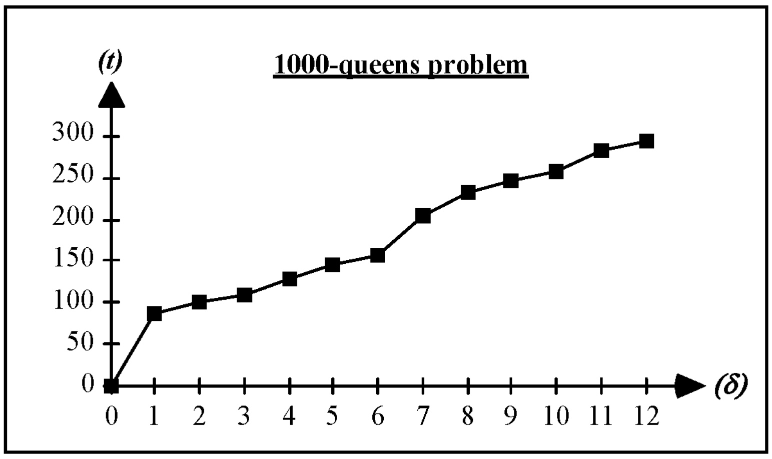

Figure 4 *Simulation of 1000-queen problem ($\delta = 12$).*

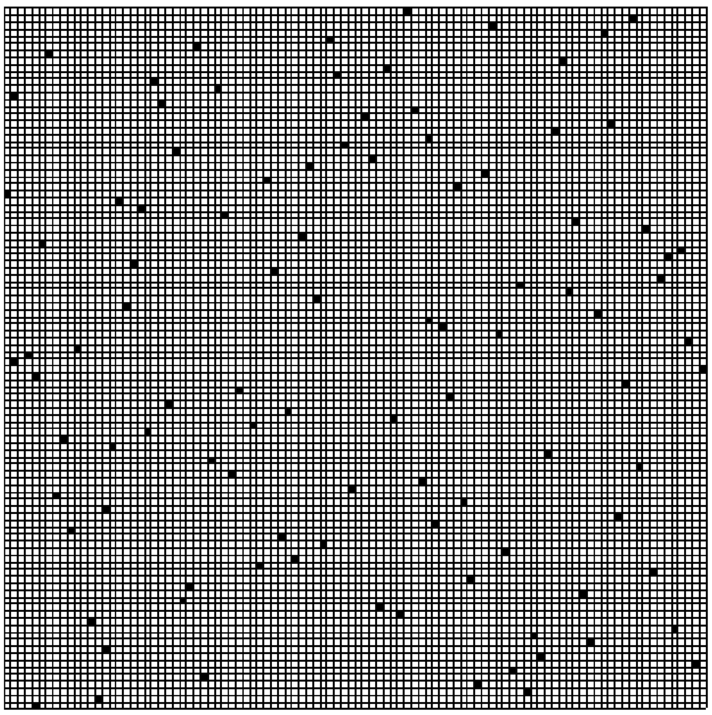

Figure 5 *One of the optimum solutions of a 100-queen problem.*

$Q_{100}(opt)$ = {57 89 69 85 46 27 6 79 54 47 21 30 2 22 58 51 86 78 50 48 24 52 43 68 37 64 0 16 19 31 81 91 42 5 96 94 18 38 93 73 80 44 17 84 60 62 70 97 10 3 1 99 11 88 33 63 23 40 55 35 20 8 15 77 29 90 32 59 49 7 65 13 87 61 9 39 45 71 41 36 92 66 26 82 25 53 56 12 95 75 83 14 76 98 72 28 67 74 34 4}

4 CONCLUSIONS

In this article, we explored the use of evolutionary search technique in solving the *N*-queen combinatorial optimization problem. The proposed optimization algorithm is robust, which it will not be trapped in local maxima and is able to uncover a repertoire of possible solutions for any given problem. The algorithm was simulated extensively with good results obtained for various depth of search. Problems up to 2000 queens have been simulated on a personal computer.

REFERENCES

1 Michalewicz, Z., Deb, K., Schmidt, M. and Stidsen, T. (2000) 'Test-case generator for nonlinear continuous parameter optimization techniques', *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp. 197–215.

2 Yamaguchi, T., Kohata, N., Wakamatsu, Y. and Baba, T. (1998) ‘Chaotic evolutionary processing and its applications’, *IEEE Conference on Fuzzy Systems Proceedings*, Vol. 1, pp. 183–188.

3 Back, T. and Schwefel, H.P. (1996) ‘Evolutionary computation: an overview’, *IEEE International Conference on Evolutionary Computation*, pp. 20–29.

4 Voigt, H.-M. (1989) *Evolution and Optimization*, Akademie-Verlag, Berlin.

5 Abramson, B. and Yung, M. (1989) ‘Divide and conquer under global constraints: a solution to the *N*-queens problem’, *Journal of Parallel & Distributed Computing*, Vol. 6, pp. 649–662.

6 Feng, M.D. and Yuen, C.K. (1994) ‘Dynamic load balancing on a distributed system’, *IEEE Symposium on Parallel and Distributed Processing*, pp. 318–325.

7 Munehisa, T., Kobayashi, M. and Yamazaki, H. (2001) ‘Cooperative updating in the hopfield model’, IEEE Transactions on Neural Network, Vol. 12, No. 5, pp. 1243–1251.

8 Rivin, I. and Zabih, R. (1992) ‘A dynamic programming solution to the *N*-queens problem’, *Information Processing Letters*, Vol. 41, pp. 253–256.

9 Takefuji, Y. (1993) *Neural Network Parallel Computing*, Kluwer Academic.

10 Nakaguchi, T., Jin’no, K. and Tanaka, M. (1999) ‘Theoretical analysis of hysteresis neural network solving N-Queens problems’, *IEEE Symposium on Circuits and Systems*, Vol. 5, pp. 555–558.

11 Ohta, M. (1999) ‘On the self-feedback controlled chaotic neural network and its application to n-queen problem’, *International Joint Conference on Neural Networks*, Vol. 1, pp. 713–716.

12 Goldberg, D.E. (2002) *The Design of Innovation: Lessons From and for Competent Genetic Algorithms*, Kluwer Academic Publishers, MA, Boston.