Swapping Algorithm and Meta-heuristic Solutions for Combinatorial Optimization n-Queens Problem

Neil Vaughan[*]
Faculty of Science and Technology
Bournemouth University (BU)
Bournemouth, United Kingdom
nvaughan@bmth.ac.uk

***Abstract*—This research proposes the swapping algorithm a new algorithm for solving the n-queens problem, and provides data from experimental performance results of this new algorithm. A summary is also provided of various meta-heuristic approaches which have been used to solve the n-queens problem including neural networks, evolutionary algorithms, genetic programming, and recently Imperialist Competitive Algorithm (ICA). Currently the Cooperative PSO algorithm is the best algorithm in the literature for finding the first valid solution. Also the research looks into the effect of the number of hidden nodes and layers within neural networks and the effect on the time taken to find a solution. This paper proposes a new swapping algorithm which swaps the position of queens.**

Keywords—n-queens, solution, meta-heuristic, neural network.

I. Introduction

Problems within combinatorial optimization consist of finding optimal arrangement from a finite set. In many such problems, exhaustive search would not be achievable. The set of feasible solutions is discrete or can be reduced to discrete, and in which the goal is to find the best solution. Some common problems involving combinatorial optimization are shown in Table I, which gives a non-exhaustive list of combinatorial optimization problems.

The n-queens problem is known to have a finite number of solutions – it happens to be 92 on a board where n = 8, but the interesting aspect from a computational complexity viewpoint is which algorithmic approach is most efficient at finding either any one solution, or the total number of all solutions. This n-queens problem can be used as a test to analyse the suitability of various intelligent algorithms in general for solving combinatorial optimization or other problems.

The n-queens problem is an example which is well known in the field of artificial intelligence. Due to the uncomplicated structure of this problem, it has been used as a springboard to test out or develop new intelligent problem solving approaches.

The n-queens problem has been attributed to several real-world applications, although the problem may initially have been proposed purely as a mathematical problem. These include scheduling and assigning tasks in a practical way, preventing deadlock on shared computer resources, traffic control, system design, robot placement for maximum sensor coverage, constraint satisfaction and other practical applications. These applications span a wide area including physics, computer science and industry which indicates why the n-queens problem is popular [1].

TABLE I. NON-EXHAUSTIVE LIST OF COMBINATORIAL OPTIMIZATION

Combinatorial optimization
Assignment problem
Closure problem
Constraint satisfaction problem
Cutting stock problem
Integer programming
Knapsack problem
Linear programming
Minimum spanning tree
Nurse scheduling problem
Traveling salesman problem
Vehicle routing problem
Vehicle rescheduling

The n-queens problem belongs to the class of constraint satisfaction problems and is known as an NP-hard problem. For dimensions of n x n it has a search space of n!.

The largest n-queens problem solved to date is for n=26 in which has 2.23 x 10^{16} solutions and took 271 days to solve on parallel supercomputers in 2009 [2].

TABLE II. NUMBER OF SOLUTIONS FOR INITIAL N-QUEENS PROBLEMS

Chess board size (n)	4	5	6	7	8	9	10	11	12
Unique Solutions	1	2	1	6	12	46	92	341	1787
Total Solutions	2	10	4	40	92	352	724	2680	14200

Table II shows the number of possible solutions for chess boards of various sizes. The total number of solutions for each board is shown in the bottom row. The number of unique solutions is also shown, unique means excluding any solutions which are a mirror image or a rotation of another solution, which could be seen as duplicates.

There is currently no known formula for finding the number of solutions from n (Table II). A variant chess related problem, called the bishops problem has a known formula which is 2n-2 [3].

There are several other related combinatorial optimisation problems (Table I).

A. N-Queens Problem definitions

The goal of the non-attacking n-queens problem is to place n queens on a n x n chessboard, in order that they cannot attack each other. This is a classical configuration problem, but it can be also formulated as a combinatorial optimization problem [3].

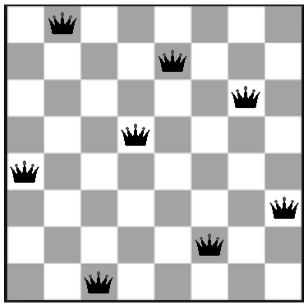

Fig. 1. One solution for the non-attacking n-queens problem

II. Existing Solutions

Various existing approaches to solve n-queens have used a variety of meta-heuristic approaches. Heuristic algorithms have been presented and compared, using such techniques as simulated annealing, tabu search, and genetic algorithms [4]. Recent methods involve particle swarm optimization, dynamic load distribution, ant colony optimization, gravitational search algorithms, and other recently developed techniques [5].

A. Genetic algorithms

Genetic algorithm approaches have proven applicable to a wide range of constraint satisfaction problems, including n-queens problem. But holism and random choices cause problem for genetic algorithm in searching large state spaces. So, the efficiency of this algorithm would be demoted when the size of state space of the problem grows exponentially. The weakness can be lessened by using local search algorithm like minimal conflicts algorithm. Minimal conflicts algorithm is trying to provide partial view for genetic algorithm by locally searching the state space. This may cause genetic algorithm to take longer steps toward the solution. Modified genetic algorithm, is the result of collaboration between genetic algorithm and minimal conflicts algorithm.

Thanks to its robustness and its capacity of adaptability to a wide variety of areas, genetic algorithms (GA) [6] are one of the most used meta-heuristics in the literature. Nowadays, a lot of research studies related to this kind of algorithm can be found.

B. Evolutionary algorithms

Recent evolution algorithms aims at the n-queens problem have been enhanced by the adoption of some quantum concepts such as quantum bits and states superposition. The use of the quantum interference has allowed this hybrid approach to improve efficiency and good results by Draa et al. (2011) [7].

C. Imperialist Competitive Algorithm (ICA)

Recently ICA was compared with Cooperative CSO and was found to have less number of fitness function evaluations than CPSO by Kalejahi et al. (2015) [6].

D. Other methodological algorithms

An approach to solving the *n*-Queens Problem is to use vector spaces. Although the situations in which this approach can be used are limited, it shows promise for wider generalization, or as an intermediate step that yields a partial solution. Finite fields can be used for generating solutions, and it can be combined with other methodologies to quickly produce solutions to the *n*-Queens Problem demonstrated by Pope & Sonnier, (2014) [8].

III. Swapping algorithm

This research has developed a new algorithm for solving n-queens problem. The aim of this algorithm is to identify a first solution as quickly as possible and with low number of moves. A solution is in the standard sense, consisting of the non-attacking problem, which is placing all queens on the board with no attacking on diagonal, horizontal or vertical lines.

The first step of the algorithm is to place initially all queens on the board in a semi-random fashion. This step ensures that exactly one queen is placed in each column and one queen in each row. At this stage there may be several attacks present on diagonal lines, but none on horizontal or vertical lines.

A quicker initialization method is to simply align queens along a diagonal from corner to corner. This can be achieved by setting for each queen the row is equal to the column. This requires no randomization and eliminates attacking on rows and columns.

Because the queens are each initialized on their own row and their own column, the algorithm never has to check for any horizontal or vertical attacking, but only diagonals need to be checked, which reduces computations and increases algorithm speed significantly.

The next stage is to detect which of the queens are attacking on diagonal lines. If some queens are found to be attacking, one is picked at random and swapped with another queen.

To identify which queen to swap with, a search is conducted amongst all queens to check whether any queen exists where a swap would eliminate the present diagonal attack. If there is, then the preferable swap is made. However if not, a swap is still made, but with a queen picked randomly.

The author wishes to acknowledge Bournemouth University for funding.

A swap process consists of simply swapping the row of the two queens, however each queen keeps their initial column, throughout the entire algorithm.

Swapping is the only method by which queens are moved in this algorithm, so this ensures no two queens are ever on the same row or column, so only diagonal attacks need to be detected.

The method used to check if a solution has been found, is to loop through every queen identifying any diagonal attacks. If a queen is in an adjacent column to another queen, it cannot be also in an adjacent row. If a queen is two columns from another queen it cannot be two rows away. Therefore a check can identify whether any two queens are positioned whereby the difference between their columns equals the difference between their rows.

IV. Results of the random swapping algorithm

An experimental test was completed to identify the number of swaps required to find a valid solution on an 8x8 board.

When the queens were initialized into a diagonal line from corner to corner, the random swapping algorithm took on average 463 swaps to find a solution. This is a high number as the random swapping has no guidance, and each swap may make things worse instead of better.

The average number of required swaps 463 was calculated by running the algorithm indefinitely, until a solution was found. Over a thousand such runs, the average number of swaps required to obtain a valid solution was 463.

This took around 60 milliseconds to find a solution, which was calculated by running a loop requiring the algorithm to find a solution 1000 times, which took 61 seconds on laptop with 2GHz processor and with 6GB RAM.

Another interesting question is, when starting with queens aligned diagonally, what is the minimum number of swaps for a board of size n to find a non-attacking solution. With this algorithm on a size 8 board, occasionally a solution was reached after only 5 swaps (42736815). On other occasions no solutions were found for over 3000 swaps, with an average of 463 swaps being required.

V. Results of the beneficial swapping algorithm

The beneficial swapping algorithm differs from the random swapping algorithm in that swaps only take place if they cause a benefit, by reducing the number of total attacks on the board.

A loop goes through every possible combination of 2 queens on the board, which can be calculated as (n+1)*(n/2) so on a board where n is 8 there are 45 combinations. After each swap is complete, if that causes no reduction effect on total number of attacks, the swap is reversed and the next combination is tried.

Beneficial swapping algorithm took on average 6 milliseconds to find a valid solution from a random starting board. This was measured by repeating the test in a loop of 1000 times, which took 6 seconds. The minimum required number of swaps was 1 and the maximum was 72. Overall the average number of swaps required was 9.712. This is much more efficient than the random swapping algorithm.

When starting with all queens arranged in a diagonal line position from corner to corner, the algorithm found a valid solution in 9 swaps every time, taking 3 milliseconds per solution.

VI. Conclusion

This research has shown that it is feasible to identify solutions to the n-queens problem by the proposed swapping algorithm. This entails keeping each queen in a separate column and swapping the row of two queens. The initial starting position may consist of all queens being arranged in a diagonal line, or a random position, but all queens should have their own row and column. Due to this, no attacks ever occur on rows or columns and the algorithm only has to check for diagonal attacks.

The method for choosing which queens to swap has a big effect on the efficiency of the algorithm. Swapping queens at random takes on average 463 swaps and 60 milliseconds from a diagonal start to find a solution.

When the queens to be swapped were instead selected in an optimized approach, whereby only queens were swapped in which doing so reduced the number of total attacks on the board, this gave a direction to the algorithm and solutions were found much quicker and efficiently. Only 9 swaps were required to complete a solution from a diagonal start position, which took 3 milliseconds. From a random starting position, the algorithm took on average 9.7 swaps which took on average 6 milliseconds with a maximum of 72 swaps.

The algorithm was able to identify a valid solution from any random starting position only by making swaps between queens in various rows.

References

[1] Watkins JJ. (2004). Across the Board: The Mathematics of Chess Problems. Princeton: Princeton University Press. ISBN 0-691-11503-6.

[2] Sloane NJA, Number of ways of placing n nonattacking queens on an n x n board, The On-Line Encyclopedia of Integer Sequences.

[3] Dudeney HE. (1970) "Bishops--Unguarded" and "Bishops--Guarded." §297 and 298 in Amusements in Mathematics. New York: Dover, pp. 88-89 and 96.

[4] Martinjak I, Golub M. (2007), Comparison of Heuristic Algorithms for the N-Queen Problem, Proceedings of the IEEE Int. Conf on Information Technology Interfaces, pp. 759-764.

[5] Pothumani. (2013) Solving N Queen Problem Using Various Algorithms - A Survey, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, No. 2.

[6] Mohabbati-Kalejahi N, Akbaripour H, Masehian E. (2015). Basic and Hybrid Imperialist Competitive Algorithms for Solving the Non-attacking and Non-dominating n-Queens Problems. *Computational Intelligence*. pp. 79-96. Springer International Publishing

[7] Draa A, Meshoul S, Talbi H, Batouche M. (2011). A quantum-inspired differential evolution algorithm for solving the N-queens problem. *Neural Networks*, 1, 12

[8] Pope J, Sonnier D. (2014), A linear solution to the n-Queens problem using vector spaces. *Journal of Computing Sciences in Colleges*, 29(5), pp. 77-83.