OPTIMIZING N-QUEENS PROBLEM USING MEMETIC ALGORITHM-A REVIEW

Amardeep Singh[1] Puneet Thapar[2]

[1] *Student, Computer Science Engineering, CTIEMT, Punjab, India*
[2] *Assistant Professor, Computer science Engineering, CTIEMT, Punjab, India*

ABSTRACT

The Problem like N-Queens problem is well known NP hard problems. In this research study, we use the Memetic Algorithm(MA) to optimize N-Queens problem and make complexity on the parameters with respect to number of iterations. The MA is a hybrid algorithm, being a combination of the Genetic Algorithm and a local search algorithm. The MA solves the N-Queens in two stages. In the first stage, the randomly generated solutions are evolved till they become practicable (i.e., the hard constraints are satisfied) and in the second stage, these solutions are further evolved so as to minimize the violations of the soft constraints. In the final stage, the MA produces optimal solutions in which the hard as well as the soft constraints are completely satisfied.

Keyword: - *Queens, Genetic Algorithm, Evolutionary Algorithm, Memetic Algorithm, Optimization, Mutation, Selection, Crossover, EA, GA, MA, Chromosome.*

1. INTRODUCTION

Problems with non-deterministic solutions that run in polynomial time are known as NP-class problems. N-Queens problem which is to place the 'n' numbers of Queens on a chess board so that neighbor Queens cannot rebuttal each other vertically, horizontally and diagonally. Some of the recent AI techniques include the use of genetic programming, evolutionary strategies, simulated annealing, hill-climbing, genetic algorithm, co-operative genetic algorithm, evolutionary programming, learning classifier systems artificial immune system and different versions of evolutionary algorithms.

Genetic algorithms are search algorithms based on the mechanisms of natural selections and natural genetics. The conventional genetic algorithm does not yield satisfactory solutions. Therefore, we believe that a hybrid methodology involving an Evolutionary Algorithm that finds several practicable solutions and a Local Search exploiting the inherent knowledge of the problem to optimize the intermediate practicable solutions is an appropriate tool to grapple this highly complex problem. The hybridization of evolutionary algorithms (EAs) with other techniques can greatly improve the efficiency of search. EAs hybridized with local search techniques are named as Memetic Algorithms(MA). A common approach is to apply the local search to the Genetic Algorithm(GA) population after crossover and mutation, with the aim of exploiting the best search regions. The ideas of memetics within a computational framework is called "Memetic Computation (MC)" [1].

In this study, we use a hybrid algorithm, named Memetic Algorithm(MA), being a combination of GA and local search (LS) to solve the complex n-Queens problem. An important aspect concerning MAs is the trade-off between the exploration abilities of the EA and the exploitation abilities of the local search technique. The Memetic algorithm is a hybrid of the Genetic Algorithm (GA) and Local Search (LS). The GA follows a simple coding scheme and after the recombination operations, LS is applied using problem-specific knowledge. Penalties are imposed for the violation of the hard as well as the soft constraints of the shift schedules. The Memetic Algorithm solves the N-Queens in two stages. In the first stage, the randomly generated solutions are evolved till they become practicable (i.e., the hard constraints are satisfied) and in the second stage, these solutions are further evolved atleast minimizing the violations of the soft constraints. The result is optimal solutions satisfying the n-Queens problem. [1]

2. N-QUEENS PROBLEM (NQP)

The classic combinatorial problem is to place N-Queens on a chessboard so that no two attack each other. In the chess Queens attacking in three directions i.e. horizontally, vertically and diagonally. The problem can be generalized as placing 'n' non attacking queens on an N x N chessboard. Since each queen must be on a different row and column, we can assume that queen "i" is placed in i^{th} column. All solutions to the NQP can therefore be represented as n-tuples (q1, q2, …, qn) that are permutations of an n-tuple (1, 2, 3, …, n). [4]

Position of a number in the tuple represents queen's column position, while its value represents queen's row position (counting from the bottom) using this representation, the solution space where two of the constraints (row and column conflicts) are already satisfied should be searched in order to eliminate the diagonal conflicts. Complexity of this problem is O (n!). The N-Queens problem is a generalization of the 8-Queens problem posed by a German chess player, Max Bezzel in 1848. The objective of the N-Queens problem is to arrange N-Queens so that no queen may attack each queen. Thus each column, row, diagonal, and anti-diagonal must contain one and only one queen.

The problem has been studied for over a 19^{th} century by many famous mathematicians such as Gauss, Polya, and Lucas. The N-Queens problem is a classic example used in computer science to demonstrate various algorithms such as evolutionary algorithms, genetic algorithms, backtracking, permutation generation, divide-and-conquer paradigm, and neural networks. The N-Queens problem is classified as non-deterministic polynomial time hard (NP-hard) which is a class of problems that are "at least as hard as the hardest problems in NP". Problem involves placing N queens on an N x N chessboard such that no queen can attack any other Benchmark code versions include finding the first solution and finding all solutions. Input for this System: A Positive integer n. Task for this System: Place n queens on an n by n chessboard so that no two queens attack each other (on same row, column, diagonal), or report that this is impossible. Solving particular problem for the different values of n=1, 2, 3, 4…n. [4]

3. PROCESS OF N-QUEENS

- Suppose you have 8 chess Queens and chess board of size 8*8.
- Queens can be placed on the chess board so no two queens are attacking each other.

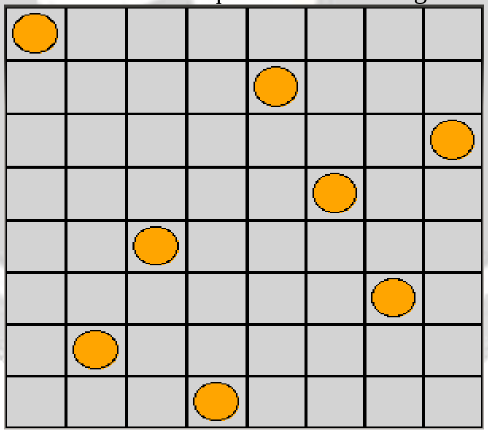

Fig-1: Chess board of size 8*8 with 8 Queens [1]

- Two Queens are not allowed in the same column.
- Two Queens are not allowed in the same column, in the same row.
- Two Queens are not allowed in the same column, in the same row, or along the same diagonal.
- The number of Queens and the size of the board can differ.
- It looks like hard to generate one valid placement.
- But it is easy to check whether a placement is valid or not.
- We will write a program which tries to find a way to place Queens on N x N chess board.
- The program uses a stack to keep track of where each Queens is placed.

- Each time the program decides to place a Queens on the board, the position of the new Queens is stored in a record which is placed in the stack.
- We also have an integer variable to keep track of how many rows have been filled so far.
- Each time we try to place a new Queens in the next row, we start by placing the Queens in the first column.
- If there is a clash with another Queens, then we shift the new Queens to the next column.
- If another clash occurs, the Queens is shifted rightward again.
- When there are no clash, we stop and add one to the value of filled.
- Let's look at the third row. The first position we try has a clash.
- So we shift to column 2. But another clash arises.
- Then we shift to the third column. Yet another clash arises.
- We shift to column 4. There's still a clash in column 4, so we try to shift rightward again.
- When we run out of room in a row: pop the stack, reduce filled by 1 and continue working on the previous row.
- Now we continue working on row 2, shifting the Queens to the right.
- This position has no clash, so we can increase filled by 1, and move to row 3. In row 3, we start again at the first column.

4. OPTIMIZATION

Optimization is an approach to finding the best solution out of a very large set of alternatives (Miettinen, 1999). Whereas single objective optimization employs a single criterion for finding the best solution among a very large set of alternatives, multi-objective optimization employs two or more criteria. The criteria used to compare solutions are known as objectives. As multiple objectives can clash with one another i.e., improving one objective leads to the decline of another there is, generally speaking, no single optimal solution to multi-objective problems. It demonstrates a fundamental issue in multi-objective optimization.

Given that there is no single optimal solution, rather a crowd of potential solutions with different degrees of tradeoff between the objective, decision makers are subsequently responsible for exploring this set of potential solutions and identifying the solutions to be implemented. While ultimately the selection of the final solution is the responsibility of the decision maker, optimization tools should assist this decision process to the best of their ability. For instance, it may prove useful to identify points of diminishing returns. To perform this type of analysis, it is necessary to provide the decision maker with computation or approximation of these tradeoffs. This strategy of computation or approximating the tradeoffs is known as a fancied optimization. An Optimization problem is the problem to finding the best solution from all practicable solutions. [1]

5. STEPS OF OPTIMIZATION

- Read the problem.
- Reread the problem.
- Draw a graph or picture.
- Identify the given information
- What quantity to be maximized or minimized?
- Find an appropriate equation for what needs to be maximized or minimized, and reduce it to one variable.
- Find the derivative and critical points for your equation.
- Test your critical points and end points (where appropriate). Reread the question and make sure you have answered what was asked.

6. N-QUEENS OPTIMIZATION USING MEMETIC ALGORITHM(MA)

The Memetic Algorithm (MA) we have used to solve the above N-Queens consists of being a combination of Genetic Algorithm and Local search. The MA flowchart is shown in Fig. 1. It describes in detail each of the steps of the MA.

6.1 Random Number Generation
Random number generation is the generation of a sequence of numbers or symbols that cannot be reasonably predicted better than by a random chance, usually through a random-number generator (RNG). A random number is a number chosen as if by chance from some specified distribution such that selection of a large set of these numbers reproduces the underlying distribution. Almost always, such numbers are also required to be independent, so that there are no correlations between successive numbers. Computer-generated random numbers are sometimes called pseudorandom numbers, while the term "random" is reserved for the output of unexpected physical processes.

When used without qualification, the word "random" usually means "random with a uniform distribution." Other distributions are of course possible. For example, the Box-Muller transformation allows pairs of uniform random numbers to be transformed to corresponding random numbers having a two-dimensional normal distribution.

There are a number of common methods used for generating pseudorandom numbers, the simplest of which is the linear congruence method. Another simple and elegant method is elementary cellular automaton rule 30, whose central column is given by 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, ... (OEIS A051023), and which provides the random number generator used for large integers in the Wolfram Language. Most random number generators require specification of an initial number used as the starting point, which is known as a "seed." The goodness of random numbers generated by a given algorithm can be analyzed by examining its noise sphere.

When generating random numbers over some specified boundary, it is often necessary to normalize the distributions so that each differential area is equally populated. For example, picking and from uniform distributions does not give a uniform distribution for sphere point picking.

6.2 Selection & Fitness
Selection is a step of a GA in which individual genomes are chosen from a population for later breeding [using the crossover operator]. A fitness function is a particular type of objective function that is used to summarize, as a single figure of merit, how close a given design solution is to achieving the set aims.

The fitness function is evaluated for each individual, fitness values which are normalized. Normalization means dividing the fitness value of each individual by the sum of all fitness values, so that sum of all resulting fitness values equal to 1. The population is sorted by descending fitness values. Gathered normalized fitness values are computed [the gathered fitness value of an individual is the sum of its own fitness value + the fitness values of all the previous individuals]. The reason that genetic algorithms cannot be considered to be a lazy way of performing design work is precisely because of the effort involved in designing a workable fitness function. Even though it is no longer the human designer, but the computer, that comes up with the final design, it is the human designer who has to design the fitness function. If this is designed badly, the algorithm will either converge on an unreasonable solution, or will have difficulty converging at all.

Moreover, the fitness function must not only correlate closely with the designer's goal, it must also be computed quickly. Speed of execution is very important, as a typical genetic algorithm must be iterated many times in order to produce a usable result for a non-trivial problem.

Fitness approximation may be appropriate, especially in the following cases:

- Fitness computation time of a single solution extremely high.
- Exact model for fitness computation is missing.
- The fitness function is odd or noisy.

6.3 Crossover
Crossover is a genetic operator used to different programming of a chromosomes from one generation to the next generation. It is analogues to reproduction and biological crossover, upon which is Genetic Algorithm are based. Crossover is a process of taking more than one parent solutions and producing a child solution from them. Crossover and mutation are two basic operators of GA.

Performance of GA very depend on them. Type and implementation of operators depends on a problem. There are many ways how to do crossover and mutation. In this chapter are only some examples how to do it for several encoding.

6.3.1 Single point crossover

A single point crossover on both parents organisms strings is selected. One crossover point is selected, binary string from beginning of chromosome to the crossover point is copied from one parent, and the rest is copied from the second parent 11001011+11011111 = 11001111 as shown in chart 2.

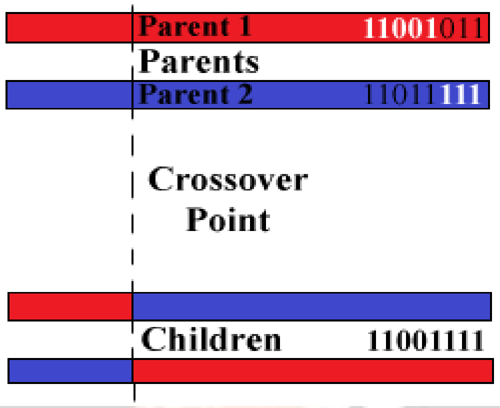

Fig -2: Single Point Crossover [5]

6.3.2 Two point crossover

Two point crossover calls for two points to be selected on the parent organisms string. Everything between the 2 points is swapped between the parents organisms. Two crossover points are selected, binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent 11001011 +11011111= 11011111 as shown in chart 3.

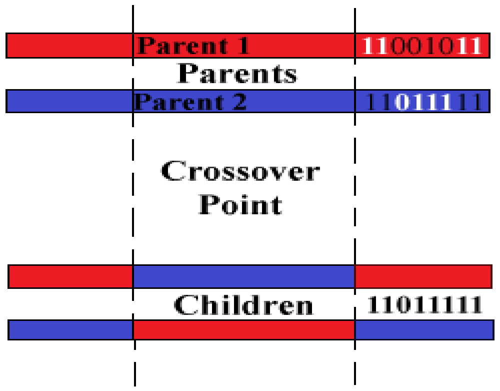

Fig -3: Two Point Crossover [5]

6.3.3 Uniform crossover

Bits are randomly copied from the first or from the second parent 11001011 + 11011101 = 11011111.

6.3.4 Arithmetic crossover

Some arithmetic operation is performed to make a new off-spring 11001011+11011111= 11001001.

6.3.5 Tree parent

The child is derived from tree randomly chosen parents. Each bit of the first parent is compared with same bit of the second parent, when these bits are the same it is used in the offspring. In both parent one crossover point is selected, parents are divided in that point and exchange part below crossover point to produce new offspring.

6.4 Mutations

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next generation. The solution may change completely from the previous solutions. Hence mutation give a perfect and better solution to genetic algorithm. Mutations are done by randomly selecting a bit and converting it into zero if it is non-zero and vice-versa. Bit inversion - selected bits are inverted 11001001 => 10001001. Mutations has various types:

- Bit string mutation
- Flip bit
- Boundary
- Non-uniform
- Uniform
- Gaussian
- Shrink.

6.5 Local Search

Local search is a heuristic method for solving computationally hard optimization problems. This can be used on problems that can be formulated as finding a solution maximizing a criterion among a number of candidate solutions.

This algorithms move from solution to solution in the space of candidate solutions (the search space) by applying local changes, until a solution deemed optimal is found or a time bound is elapsed. For given problem instance π:

- Search space S (π) (solution set S 0 (π) ⊆ S (π))
- Neighborhood function N (π): S (π) 7→ 2 S (π)
- Evaluation function f (π): S 7→ R
- Set of memory states M (π)
- Initialization function init: ∅ 7→ P(S (π) × M (π))
- Step function step: S (π) × M (π) 7→ P(S (π) × M(π))
- Termination predicate terminate: S (π) × M (π) 7→ P({>, ⊥}).

7. GENETIC ALGORITHM

Genetic Algorithm are search algorithms based on the mechanisms of natural selection and natural genetics. Genetic algorithms (GA) are randomized search and optimization techniques that are motivated by the principals of natural selection and evolution processes. The GA was developed by John Holland in 1975, his colleagues, and his students at the University of Michigan. He was inspired by the concept of Darwinian‘s principle of survival of the fittest individuals and natural selection and developed the theory of genetic algorithm. Since then many researchers are using this algorithm to solve various optimization problems effectively.

7.1 Steps of Genetic Algorithm

- Instantiation
- Fitness function
- Size of population
- Elitism rate
- Mutation rate
- Number of iterations.

In case of GA usually we get optimal solution for a particular problem, but in case of N-Queens problem we need a superior solution which can be given by implementing certain technique such as local search based MA.

8. APPLICATIONS OF N-QUEENS

There are variety of N-Queens Applications that is deal with the daily life and real world problems. Some of these are given below**:**

- Image Processing.
- Deadlock Prevention.
- Very large scale integration(VLSI) Testing.
- Traffic control.
- Register Allocation.
- Motions estimation.

9. CONCLUSIONS

The N-Queens Problem (NQP), like the well known Travelling Salesman Problem (TSP), is an NP-hard problem. With this intuition, we have a Memetic Algorithm (GA + local search) to solve the NQP. The Hybrid MA, being a combination of GA and LS to solves the NQP in two phases. In the first stage, the randomly generated solutions are evolved till they become practicable (i.e., the hard constraints are satisfied) and in the second stage, these solutions are evolved so as to minimize the violations of the soft constraints.

10. REFERENCES

[1] Er. Vishal Khanna, Er. Sarvesh Chopra, Review on N-Queen Optimization Using Tuned Hybrid Technique, International Journal Of Engineering Sciences & Research Technology (Thomson Reuters), pp.62-68, Vol.6, Issue.2, 2017.

[2] Yangming Zhou, Jin-Kao Hao, and B´eatrice Duval, Oppositionbased Memetic Search for the Maximum Diversity Problem, IEEE Transactions on Evolutionary Computation, pp.1-15, 2017.

[3] Adam Santos, Reginaldo Santos, Moisés Silva, Eloi Figueiredo, Claudomiro Sales, and João C. W. A. Costa, A Global Expectation–Maximization Approach Based on Memetic Algorithm for Vibration-Based Structural Damage Detection, Ieee Transactions On Instrumentation And Measurement, pp.110, 2017

[4] Yifeng Zeng, Xuefeng Chen, Yew-Soon Ong, Jing Tang and Yanping Xiang, Structured Memetic Automation for Online Human-like Social Behavior Learning, Ieee Transactions On Evolutionary Computation, pp.1-14, 2017.

[5] A.H. Beg, Md Zahidul Islam, Advantages and Limitations of Genetic Algorithms for Clustering Records, 2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA), pp.2478-2483, 2016.

[6] Sarkan Guldal and Veronica Baugh, ―N-Queens Solving Algorithm by Sets and Backtracking‖, IEEE Southeast Conference, pp.125-129, 2016.

[7] B Documentaries, ―Full Solution of N-Queens Problem O Reilly‖, http://oreillynQueensproblem.blogspot.in, 3-Sept2016.

[8] Tad Gonsalves and Kohei Kuwata, ―Memetic Algorithm For The Nurse Scheduling Problem‖, International Journal of Artificial Intelligence and Application, pp. 43-52, 2015.

[9] Soham Mukherjee, Santanu Datta, Pramit Brata Chanda and Pratik Pathak, Comparative Study Of Different Algorithms To Solve N Queens Problem, International Journal in Foundations of Computer Science & Technology, Vol.5, Issue.2, pp.15-27, March 2015.

[10] Amarbir Singh and Sandeep Singh Dhillon, ―A Comparative Study of Algorithms for N-Queens Problem‖, International Journal of Advance Foundation and Research in Science and Engineering , Vol.1, Special Issue, pp.1-4, 2015.

[11] Soham Mukherjee, Santanu Datta, Pramit Brata Chanda and Pratik Pathak, ―Comparative Study of Different Algorithms To Solve N-Queens Problem‖, International Journal of Foundations of Computer Science and Technology, Vol.5, Issue.2, pp.15-27, 2015.

[12] Ahmed S. Farhan , Wadhan Z. Tareq and Fouad H. Awad, ―Solving N-Queens Problem using Genetic Algorithm ‖, International Journal of Computer Applications, Vol.122, Issue.12, pp.11-14, 2015.

[13] Vikas Thada and Shivali Dhaka, ―Performance Analysis of NQueens Problem using Backtracking Algorithm Techniques ‖, International Journal of Computer Applications, Vol.102, Issue.7, pp. 26-29, 2014.

[14] Ellips Masehian, Hossein Akbaripour and Nasrin MohabbatiKalejahi,―Solving the n-Queens Problem Using a Tuned Hybrid Imperialist Competitive Algorithm ‖, The International Arab Journal of Information Technology, Vol.11, Issue.6, pp.550-559, 2014.

[15] Belal Al-Khateeb, Wadhah Z. Tareq, Solving 8-Queens Problem by Using Genetic Algorithms, Simulated Annealing and Randomization Method, 2013 Sixth International Conference on Developments in eSystems Engineering, pp.187, 2013.

[16] Vishal Kesri and Manoj Kumar Mishra, ―A new approach to solve n-Queens problem based on series‖, International Journal of Engineering, Research and Applications, Vol.3, Issue.3, pp.1349-1349, 2013.

[17] Ram Gopal Sharma and Bright Keswani, ―Implementation of NQueens Puzzle using Meta-Heuristic Algorithm (Cuckoo Search)‖ International Journal of Latest Trends in Engineering and Technology, Vol. 2, Issue. 3, pp. 343-347, 2013.

[18] S.Pothumani, ―Solving N-Queens Problem using Various Algorithms-A Survey‖, International Journal of Advance Research in Computer Science and Software Engineering, Vol. 3, Issue. 2, pp. 247-250, 2013.

[19] Kenekayoro Patrick, Comparison of simulated annealing and hill climbing in the course timetabling problem, African Journal of Mathematics and Computer Science Research, Vol. 5, Issue.11, pp.176-178, September 2012.

[20] Farhad Soleimanian, Bahareh Seyyedi and Golriz Feyziour, ―A New Solution for N-Queens Problem using Blind Approaches: DFS and BFS Algorithms‖, International Journal of Computer Applications, Vol.53, Issue.1, pp.45-48, 2012.

[21] Vishal Kesri, Vaibhav Kesri and Prasant Ku. Pattnaik, ―A Unique Solution for N-Queens Problem‖, International Journal of Computer Applications, Vol.43, Issue.12, pp.13-19, 2012.

[22] Baolei Gu,‖ Research and Realization of N-Queens Problem Based on the Logic Language Prolog‖, Springer Computational Intelligence and Intelligent System, Vol.4, Issue.1, pp. 50-56, 2012.

[23] Aftab Ahmed, Attique Shah, Kamran Ali Sani and Abdul Hussain Shah Bukhari,‖ International Journal of Advance Computer Science and Technology‖ Vol.1, Issue.2, pp. 57-63, 2012.

[24] Jun Zhang and Zili Zhang, ―An Algebraic method for the nQueens Problems based on Permutation Operation Group‖, International Journal of Computer Networks and Information Security, Vol.3, pp.19-25, 2011.

[25] Jordan Bell and Brett Stevens, ―A Survey of Known results and research areas for n-Queens‖ Discrete Mathematics Science Direct, Vol.309, Issue.1, pp.1-31, 2008.

[26] H. Ahrabian, A. Mirzaei and A.Nowzari-Dalini,‖ A DNA Sticker Algorithm for Solving N-Queens Problem‖, International Journal of Computer Science and Applications, Vol.5, Issue.3, pp.12-22, 2006.

[27] Chung-Neng Wang, Shin-Wei Yang, Chi-Min Liu and Tihao Chiang, ―IEEE Transactions on Circuits and System for Video Technology‖, Vol.14, Issue.4, pp.429-440, 2004.

[28] Marko Bozikovic, Marin Golub and Leo Budin, ―Solving NQueens Problem Using Global Parallel Genetic Algorithm‖, European Conference Ljubljana Slovenia, pp.11-17, 2003.

[29] Roc Sosic and Jun Gu, ―Fast Search Algorithms for the NQueens Problem‖, IEEE Transactions on Systems, Man, and Cybernetics, Vol.21, Issue.6, pp.1572-1576, 1991.

[30] Rok Sosic and Jun Gu, ―A Polynomial Time Algorithm for NQueens Problem‖, Special Interest Group on Artificial Intelligence, vol.1, Issue.3, pp.7-14, 1990.