Solution of n-Queen Problem Using ACO

Salabat Khan, Mohsin Bilal, M. Sharif, Malik Sajid, Rauf Baig
National University Of
Computer and Emerging Science
Islamabad, Pakistan
Email: salabat.khan@nu.edu.pk
Telephone: (+92)51-4532308

***Abstract*—In this paper, a solution is proposed for n-Queen problem based on ACO (Ant Colony Optimization). The n-Queen problem become intractable for large values of 'n' and thus placed in NP (Non-Deterministic Polynomial) class problem. The n-Queen problem is basically a generalized form of 8-Queen problem. In 8-Queen problem, the goal is to place 8 queens such that no queen can kill the other using standard chess queen moves. So, in this paper, the proposed solution will be applied to 8-Queen problem. The solution can very easily be extended to the generalized form of the problem for large values of 'n'. The paper contains the detail discussion of problem background, problem complexity, Ant Colony Optimization (Swarm Intelligence) and a fair amount of experimental graphs.**

***Index Terms*—n-Queen Problem, 8-Queen Problem, Heuristic Techniques, Ant Colony optimization (ACO), Swarm Intelligence (SI).**

I. Introduction

The problems in computer science can be grouped in different classes on the basis of time required to find a correct solution. The class 'P' contains all those decision problems for which polynomial time algorithms exist [1]. The problems which can be solved in polynomial time on a non-deterministic computer are placed in 'NP' class. The non-deterministic computer is a theoretical computer that does not exist. This theoretical computer contains infinite amount of resources to spawn as many processes as there are possible solutions to a problem. Note that however, solution to a problem in 'NP' can be verified in polynomial time [1]. In order to tackle the problems in 'NP', heuristic techniques are used.

The n-Queen problem is also an intractable problem i.e. for large values of 'n' the problem can not be solved in polynomial time and thus placed in 'NP' class. There are total $\binom{n^2}{n}$ number of possible arrangements of 'n' queens on the chessboard [2]. For 8-queen problem, there are 4,426,165,368 possible arrangements of queens on board out of which only 92 are correct solutions [3]. The solution to the n-Queen problem proposed here is based on a heuristic known as "Ant Colony Optimization".

The remainder of this paper is organized as follows. In the next section, we review related research. In Section III we present the basics and the background of ant colony optimization meta-heuristic. In Section IV, architecture of the proposed solution will be discussed. Subsequently, in Section V we present some simulation results to show the ability of our approach. Finally, Section VI will conclude this work.

II. Related Work

Previously, lots of work is done on this problem. K. D. Crawford in [2], applied Genetic Algorithm and have discussed two ways to solve n-Queen problem. Ivica et al. provided a comparison of different heuristic techniques in [1]. The techniques include Simulated Annealing, Tabu Search and Genetic Algorithm.

We found no solution to the problem based on Ant Colony Optimization. So, we can say that this is first ever application of ACO to the n-Queen problem. In order to apply ACO, we first organized the search space. We then discussed a few modifications in the calculation of some parameters for ACO. We have added a few constraints in basic ACO as it can not be applied directly to the n-Queen problem. A detail discussion on all these is provided in the subsequent sections.

III. Basics and Background of ACO

The ant colony optimization (ACO) is a meta-heuristic that is inspired by intelligent behaviors of ants. ACO is a multi-agent system; an ant behavior depicts the behavior of an agent in the system. The first ant algorithm was developed by Dorigo [4]. Improvements in algorithm are made in [5],[6].Ants in nature modify there environment by constantly depositing a chemical substance called pheromone. The pheromone is used as an indirect communication among ants and guides them to find shortest path from their nest to a food place. If there are multiple paths to a food place, ants choose one with high concentration of pheromone. In Figure 1, one can logically find that how a shortest path will be concentrated with the high pheromone values as most of the ants on this path will come back quickly.

A. Simple ACO algorithm

Let G = (V, E) be a connected graph where $|V|$shows total number of nodes and total $|E|$number of edges in graph. The simple ant colony optimization meta-heuristic can be used to find the shortest path between a given source node V_s and a given destination node V_d in the graph 'G'[8]. The path length is either given by the number of nodes on the path or summation of cost values on edges constituting the path. Each edge $e(i, j) \in E$ of the graph connecting the nodes V_i and V_j has a variable (artificial pheromone), which is modified by the ants when they visit the nodes[8].

978-1-4244-4873-9/09/$25.00 ©2009 IEEE

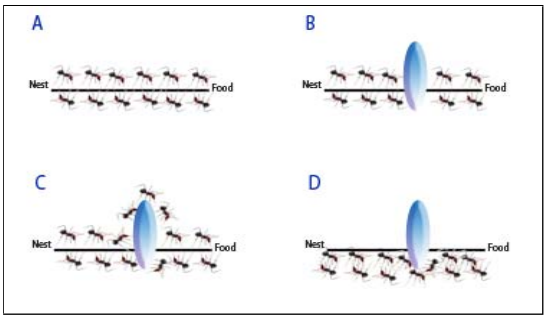

Fig. 1. Ants finding shortest path. a) No obstacle. b) Obstacle Placed. c) Most ants on shortest path a few on longest. d) Ants found shortest path.

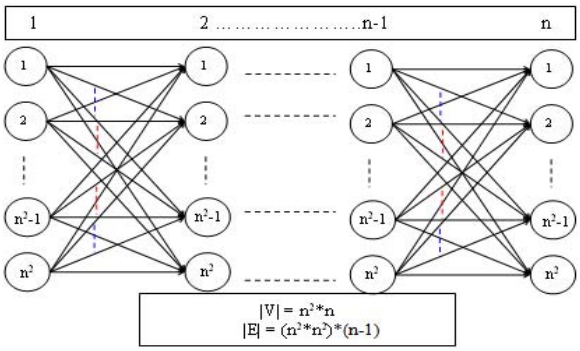

Fig. 2. Search Space for n-Queen, ACO

From a node, when an ant decides which node to move next, it uses 02 parameters to calculate the probability of moving to a particular node; first, distance to that node and second, amount of pheromone on the connecting edge. Let $d_{i,j}$ be the distance between the nodes i and j, the probability that the ant chooses j as the next city after it has arrived at city i where j is in the set 'S' of cities that have not been visited is:

$$p_{i,j} = \frac{[\tau_{i,j}]^{\alpha}.[\eta_{i,j}]^{\beta}}{\sum_{k \in S}[\tau_{i,k}]^{\alpha}.[\eta_{i,k}]^{\beta}} \quad (1)$$

Where $\tau_{i,j}$ is the pheromone value on $edge(i,j)$, $\eta_{i,j}$ is a heuristic value calculated as $\frac{1}{d_{i,j}}$. The parameters α and β are influencing factors of pheromone value and heuristic value respectively.

Some best ants (having good solutions) or all ants modify the pheromone values on the edges added to their tour. One possible modification may be done as:

$$\tau_{i,j} = \tau_{i,j} + \frac{Q}{L} \quad (2)$$

Where Q is some constant and L is the length of the tour, small the value of L high the pheromone value added to the previous pheromone value on edge.

With time, concentration of pheromone decreases due to diffusion affects; a natural phenomenon known as evaporation. This also ensures that old pheromone should not have a too strong influence on the future[7]. This can be done as:

$$\tau_{i,j} = \tau_{i,j}.\rho \ \{\text{where } \rho \text{ will be between 0 and 1}\} \quad (3)$$

IV. N-Queen ACO

N-Queen problem is to place 'n' queens on chessboard such that no queen can kill the other using standard chess queen moves. There are three such situations in which a queen can kill the other; first, if two queens are in the same row, second, if two queens are in the same column and third if one queen is in the diagonal of the other queen.

A. Solution/ Search Space

In order to solve the n-Queen problem, basic ACO is modified and some constraints are added. Moreover, the equations described above are also changed a little bit. The cores of this solution include "formation of search space" & "calculation of heuristic value". In Figure 2, formation of search space is described which makes the search efficient and fast:

The vertices in search space are organized as a grid of n^2 rows * n columns. Every vertex in a column is connected to all the vertices in the next column through directed edges except vertices in n^{th} column. The vertices label shows the chess cell position e.g. label 1 is used as a mapping to cell # [0,0], label 2 is used as a mapping to cell # [0,1] and n^2 is used as a mapping to cell # [n-1, n-1]. For 8-queen problem or 8*8 chess board positions, n will be 8, above gird will have 64*8 vertices, 64*64*7 edges and n^2 i.e. 64 will map to cell # [7,7] at chess board.

In order to simplify the things, we will consider n equal to 8 i.e. we will now talk about 8-queen problem, as from the search space above it is evident that solution to 8-queen problem is easily extendible to n-queen problem.

B. Constraints added in basic ACO

An ant can only move from left to right. Once a node is selected at certain column, it can pick the next node to move at, only from the column, next to the current column. Tour ends over last column.

An ant during a tour visits only 'n' nodes (8 for 8-queen). The label of nodes in the tour can not be the same. The node in a tour represents a cell of chessboard where a queen is to be placed. This restriction is added as if two nodes have same labels then 02 queens will be placed in same cell which is

illegal. This essentially means that no two nodes in a tour will be in the same row of the search space.

C. ACO parameters initialization and equations

Initially, all the edges are assigned with the same small pheromone values. Arbitrary number of ants (swarm size) is created as multi-agent system. Pheromone value is modified on the basis of fitness value of a solution found by an ant.

Fitness Value: The fitness represents the number of positions at chessboard that satisfy the game constraint i.e. queens if placed on these positions will not kill any other queen at chessboard. In case of 8-queen problem the possible range of fitness values is 0-8.

Heuristic Value and Evaporation: The probability equation is same as described above but the calculation of heuristic value is changed. Now, in Equation (1), heuristic value is not based on the distance between two vertices rather number of contradictions if node j is selected as next node. Contradictions represent the positions at chessboard where if queens are placed will kill each other. Note that node j is a chess position at which a queen will be placed. The probability equation for selecting a node is:

$$p_{i,j} = \frac{[\tau_{i,j}]^{\alpha}.[\eta_{i,j}]^{\beta}}{\sum_{k \in S}[\tau_{i,k}]^{\alpha}.[\eta_{i,k}]^{\beta}} \quad (4)$$

Further, value of α and β is initialized with a random number generated in range 0-2. The evaporation is done on constant rate in our solution, decaying the pheromone value on all the edges is done after the completion of a single iteration. A single iteration is completed when all the ants complete their tour. The evaporation is done by subtracting the current pheromone value of an edge from a small constant.

V. Simulation Results

The ACO is a probability based algorithm, so, the results it would generate will be different if run multiple times on the same instance of a problem. So, in order to generate simulations results, we run it 50 times for 8-queen problem and then take the average of the generated output. In Figure 3, average fitness is drawn against maximum iterations allowed. Note that, if we keep increasing the maximum iterations allowed, we also get improving average fitness. In Figure 4, relationship between average iterations taken and max iteration is shown. We used two termination criterions; one application is terminated if we get maximum fitness i.e. 8 in case of 8-queen problem, second, if max allowed iterations are over. It's possible that we get the correct solution before the maximum iterations are over. So, the average number of iterations taken, shows the early convergence of the algorithm in certain tests out of 50 total tests. Some parameters are kept constant as swarm size (number of ants) = 10, alpha and beta = 1 for the figures 3, 4 and 5.

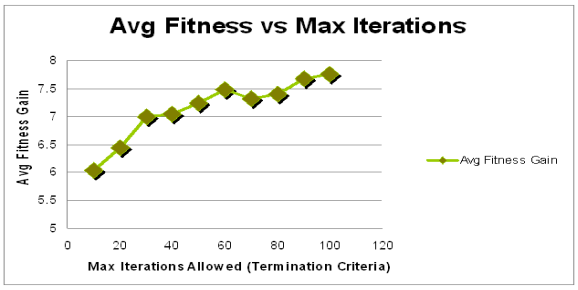

Fig. 3. Avg Fitness vs Max Iterations

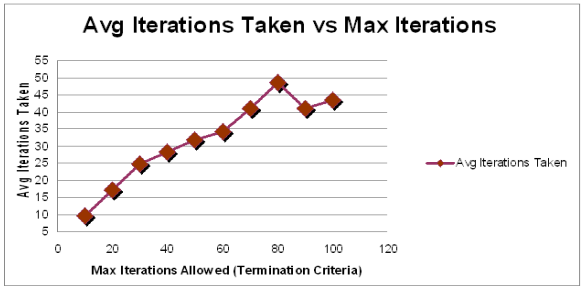

Fig. 4. Avg Iterations Taken vs Max Iterations

In Figure 5, perfect convergence is shown against maximum iterations allowed. Perfect convergence mean that the correct solution is found (fitness = 8). The behavior is self descriptive that as long as we keep increasing the number of maximum iterations allowed, the perfect convergence is also increased.

For Figure 6 and 7, parameters which are kept constant are as maximum iterations allowed = 20, alpha and beta = 1. The relationship between swarm size and average fitness gain is depicted in figure 6, as expected if we keep increasing the swarm size the average fitness gain will also become better. In Figure 7, as shown prefect convergence also increases if swarm size is increased.

For Figure 8 and 9, parameters which are kept constant are

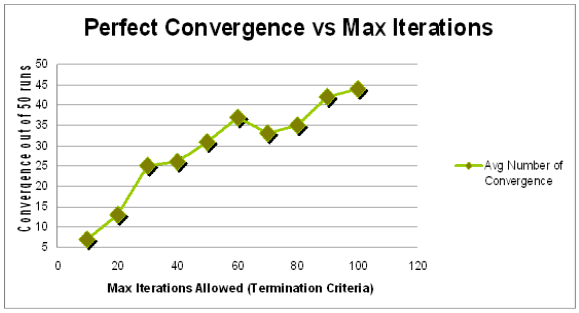

Fig. 5. Perfect Convergence vs Max Iterations

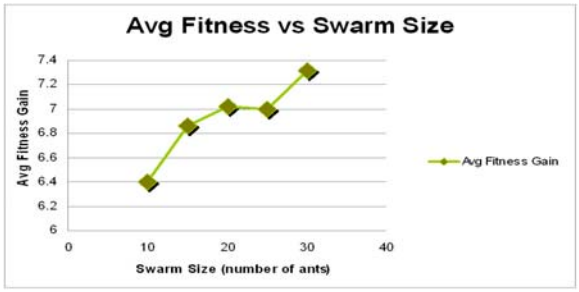

Fig. 6. Avg Fitness vs Swarm Size

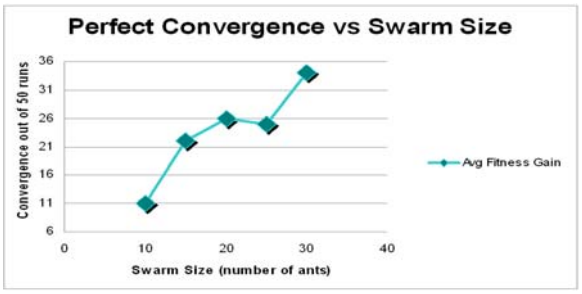

Fig. 7. Perfect Convergence vs Swarm Size

as maximum iterations allowed = 20, swarm size = 10 and beta = 1. The average fitness gain and perfect convergence are found best on alpha value 1.5 as shown in Figure 8 and 9, respectively. Both of curves are declining after the alpha value 1.5, so, it may be considered as a saturation point if the value of beta is 1.

For Figure 10 and 11, parameters which are kept constant are as maximum iterations allowed = 20, swarm size = 10 and alpha = 1. The average fitness gain and perfect convergence are found best on beta value 1.5 as shown in Figure 10 and 11, respectively. Both of curves are declining after the beta value 1.5, so, it may be considered as a saturation point if the value of alpha is 1.

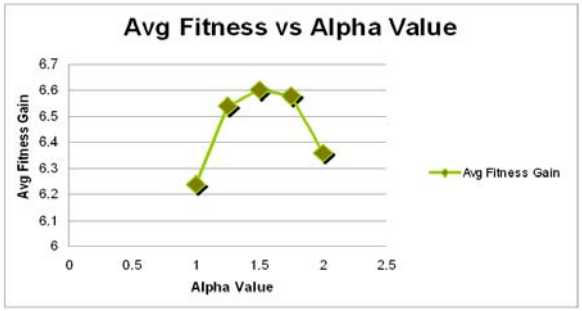

Fig. 8. Avg Fitness vs Alpha Value

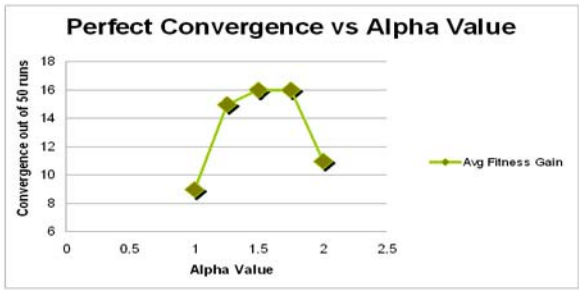

Fig. 9. Perfect Convergence vs Alpha Value

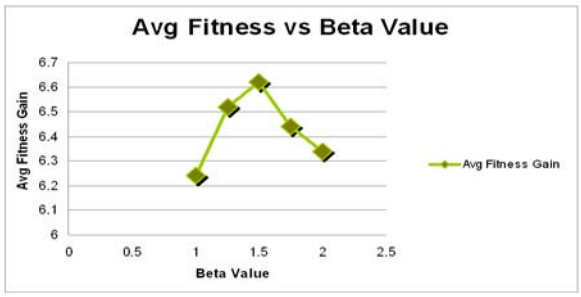

Fig. 10. Avg Fitness vs Beta Value

VI. Conclusion

The solution we proposed is working efficiently for 8-queen problem and it can easily be extended to large values of 'n' because of the simplistic model of the search space. A solution is shown in Figure 12. We found our solution working efficiently for 8-queen if parameters are set as swarm size = 15, alpha = 1, beta = 1.5 or alpha = 1.5 and beta = 1. We conclude that the ACO can provide better solution in reasonable amount of time for combinatorial optimization problems and as future work we will explore its applicability to some other such problems.

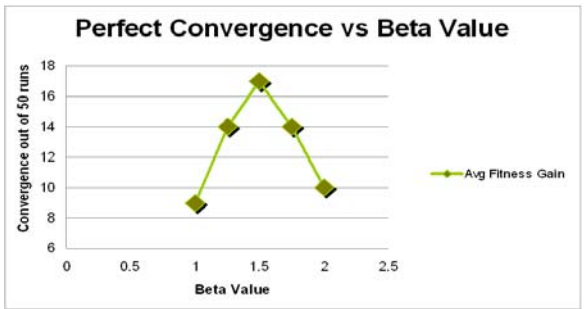

Fig. 11. Perfect Convergence vs Beta Value

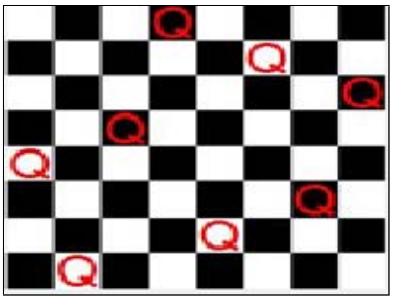

Fig. 12. A solution to 8-Queen Problem

ACKNOWLEDGMENT

The authors would like to acknowledge Higher Education Commission (HEC) of Pakistan and Dr. Rauf Baig for their continuous support. It would have been impossible to complete this effort without their continuous support.

REFERENCES

[1] I. Martinjak and M. Golub, "Comparison of Heuristic Algorithms for the N-Queen Problem", Proceedings of the ITI 2007 29th Int. Conf. on Information Technology Interfaces, June 25-28, 2007.

[2] K. D. Crawford, "Solving the N-Queens Problem Using Genetic Algorithms", In Proceedings ACM/SIGAPP Symposium on Applied Computing, Kansas City, 1992, pages 1039-1047.

[3] "Eight Queen Puzzle", [Online] Available: http://en.wikipedia.org/wiki/Eight _queens_puzzle [Accessed: July. 20, 2009].

[4] M. Dorigo. Optimization, "Learning and Natural Algorithms", PhD thesis, Politecnico di Milano, 1992.

[5] L.M. Gambardella and M. Dorigo, "Ant-Q: A Reinforcement Learning Approach to the TSP", In Proceedings of Twelfth International Conference on Machine Learning, 1995, pages 252-260.

[6] L.M. Gambardella and M. Dorigo, "Solving Symmetric and Asymmetric TSPs by Ant Colonies", In Proceedings of IEEE International Conference on Evolutionary Computation, 1996, pages 622627.

[7] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, Second Edition, John Wiley & Sons, 2007.

[8] M. Gunes, U. Sorges and I. Bouazizi, "ARA - The Ant-Colony Based Routing Algorithm for MANETs", International Workshop on Ad Hoc Networking (IWAHN 2002), Vancouver, British Columbia, Canada, August 18-21, 2002.