Department of Computer Engineering

Bu-Ali Sina University

Evolutionary computing Course

Principle of evolutionary computing , assignment 1

By:

Ehsan Rostami

Academic Supervisor:

Professor Hassan Khotanlou

Fall 2024

1 INTRODUCTION

The N-Queen problem is a classic constraint satisfaction problem where the objective is to place N queens on an N x N chessboard in such a way that no two queens can attack each other. In this problem, a queen can attack another queen if they are in the same row, column, or diagonal. Finding a solution to this problem can be challenging, especially for large values of N.

Genetic Algorithms (GAs) are a class of optimization techniques inspired by the process of natural selection. They are particularly useful for solving complex problems like the N-Queen problem, where traditional methods may struggle. In this report, we utilize GAs to find a valid arrangement of queens on the chessboard, with the goal of minimizing conflicts between queens.

METHODOLOGY

1.1 Chromosome Representation

The chromosome in a genetic algorithm represents a potential solution to the problem. For the N-Queen problem, a chromosome is represented as a list of integers where each index represents a row on the chessboard, and the value at that index represents the column position of the queen in that row. For example, a chromosome for a 4x4 board might be represented as:

$$[1, 3, 0, 2]$$

This means that: - Queen 1 is placed in row 0, column 1 - Queen 2 is placed in row 1, column 3 - Queen 3 is placed in row 2, column 0 - Queen 4 is placed in row 3, column 2

1.2 Fitness Function

The fitness function evaluates the quality of a solution based on the number of non-attacking pairs of queens. A pair of queens attacks each other if they are placed in the same row, column, or diagonal. The fitness function is designed to return a higher value for solutions with fewer attacking pairs. The function can be defined as:

$$\text{fitness} = \frac{1}{1 + \text{number of attacking pairs}}$$

This ensures that solutions with fewer attacks have a higher fitness score.

1.3 Crossover Mechanism

Crossover is a genetic operator used to combine the genetic information of two parent chromosomes to produce offspring. We implement a one-point crossover mechanism, where a random point is chosen in the chromosome, and the segments after that point are swapped between the parents. This results in two offspring chromosomes, which are used for further reproduction.

1.4 Mutation Mechanism

Mutation introduces randomness into the algorithm to maintain genetic diversity and avoid premature convergence. A simple mutation involves randomly changing the position of a queen in one of the rows. This ensures that new genetic variations are explored and can potentially lead to

better solutions.

1.5 Selection Method

The selection process determines which chromosomes will be chosen to create the next generation. In this implementation, we use tournament selection. In tournament selection, a subset of the population is randomly selected, and the best solution from this subset is chosen to become a parent for the next generation.

1.6 Algorithm Workflow

The Genetic Algorithm proceeds as follows:

- Generate an initial population of random chromosomes.
- Evaluate the fitness of each chromosome in the population.
- Use tournament selection to choose parents based on their fitness values.
- Apply crossover and mutation to create offspring.
- Replace the old population with the new offspring.
- Repeat the process until a solution is found or a maximum number of generations is reached.

EXPERIMENTAL RESULTS

For the N-Queen problem, we tested the Genetic Algorithm on various sizes of chessboards, specifically for N = 8, 10, and 12. The following parameters were used for the experiment:

- Population size: 100
- Number of generations: 1000
- Crossover rate: 0.8
- Mutation rate: 0.1

We performed multiple runs of the algorithm, and in all cases, the GA successfully found a solution within a reasonable number of generations. The results for N = 8 and N = 10 are summarized in Table 1.

Table 1: Experimental Results for N-Queen Problem

N (Board Size)	Generations to Solution	Fitness of Best Solution
8	35	1.0
10	765	1.0
12	94	1.0

A visualization of the chessboard for the N = 8 case after finding the solution is shown in Figure 1. The solution is represented with queens (Q) placed on the board.

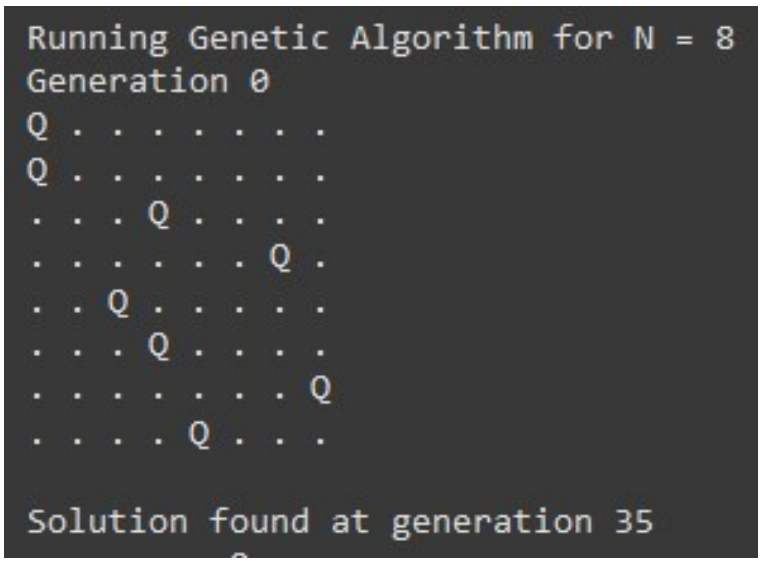

Figure 1: Solution for N = 8 after applying Genetic Algorithm

CONCLUSION

In this report, we successfully applied Genetic Algorithms to solve the N-Queen problem. The algorithm converged to valid solutions efficiently for various board sizes (N = 8, 10, 12). The combination of chromosome representation, fitness evaluation, crossover, mutation, and selection allowed the GA to effectively explore the solution space. The experimental results showed that the GA could find solutions in a reasonable number of generations, and the fitness of the solutions reached 1.0, indicating that no queens were attacking each other. Future improvements could involve fine-tuning the algorithm's parameters or exploring more advanced genetic operators to further optimize the performance.