OPEN ENDED LAB

Artificial Intelligence

Third Year-Computer and Information Systems Engineering

Batch: 2022

Group Members:

FAHAD RASHEED ……………………………………….. (CS-22114)

MUHAMMAD AWAB SHUJA ……………………………. (CS-22132)

ABDUL AHAD …………………………………………… (CS-22139)

Submitted to: Miss Hameeza Ahmed

Submission Date: 25/11/2024

Table Of Contents

Abstract:

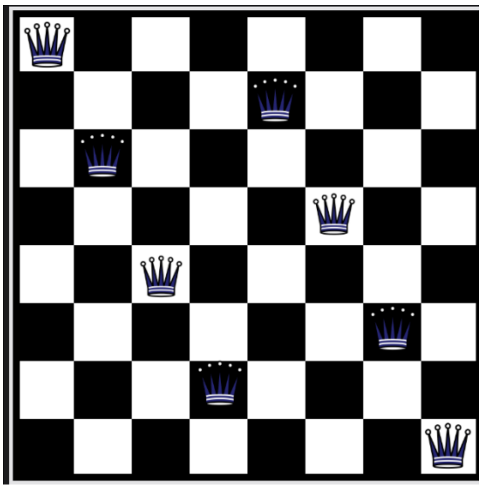
The N-Queen Problem is a classic combinatorial optimization challenge where N queens must be placed on a $N \times N$ chessboard so that no two queens threaten each other. This paper explores the application of a Genetic Algorithm (GA) to solve this problem. GAs are metaheuristics inspired by natural selection, employing operations such as selection, crossover, and mutation to iteratively improve solutions. This approach is particularly suited to the N-Queen Problem due to its ability to navigate large search spaces effectively. The report demonstrates how GAs can evolve valid board configurations over generations and highlights the flexibility and efficiency of this approach for solving complex constraint problems.

Introduction:

The N-Queen Problem requires placing N queens on an $N \times N$ chessboard such that no two queens are in the same row, column, or diagonal. The challenge lies in finding a configuration that satisfies these constraints, especially as N increases, which results in an exponentially growing search space. Traditional brute-force methods become computationally prohibitive for larger N, making it necessary to adopt heuristic approaches such as Genetic Algorithms. The goal of this study is to explore how GAs can solve the N-Queen Problem efficiently and to demonstrate their capability to handle complex combinatorial optimization problems.

Objective:

The goal is to place eight queens on an 8x8 chessboard so that no two queens threaten or attack each other. To prevent the queens from attacking one another, no two queens should be in the same row, column, or diagonal.

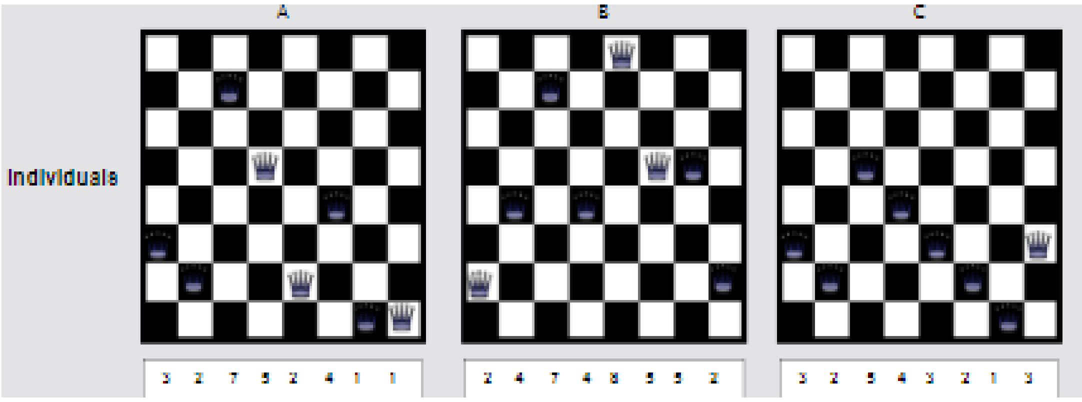


Approach:

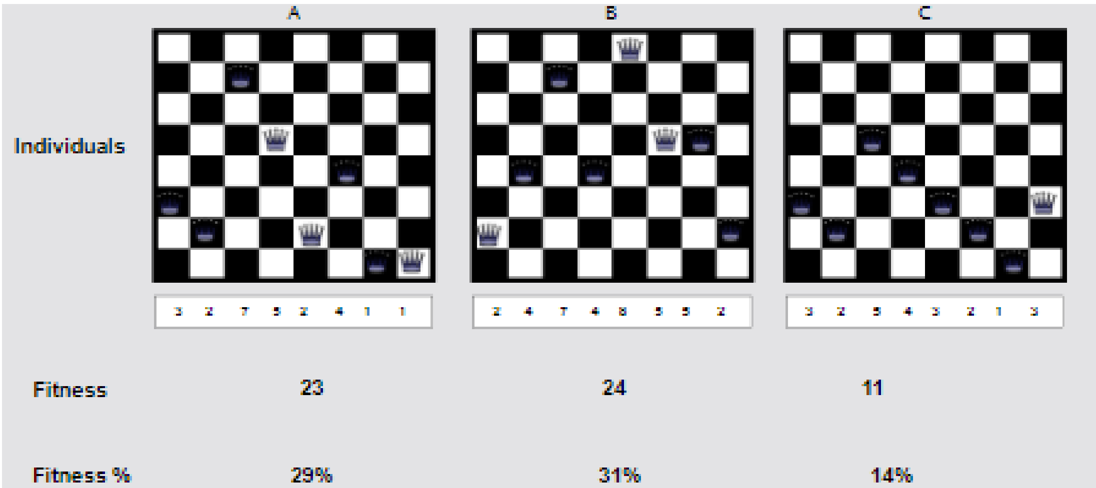
We'll using Genetic Algorithm for solving N-Queen Problem. A Genetic algorithm is an informed search heuristic greatly inspired by the biological principles of natural selection and genetics. It iteratively enhances a population of potential solutions by simulating the natural evolution process, which includes mutation, crossover, and selection. The process continues until a satisfactory solution or the maximum number of generations is reached.

Steps in the Genetic Algorithm:

1. Initialization: Create a randomized population of potential solutions. These arrays represent the queen's positions on the chessboard. The index represents the column, and the row represents the value.



2. Fitness Evaluation: Calculate each potential solution's fitness. The number of non-attacking pairs in the 8-Queens puzzle determines its fitness function.



The fitness is calculated by adding the number of non-attacking pairs for each queen. For instance, for board A, the queen positions are [3, 2, 7, 5, 2, 4, 1, 1], the queen at index 1, value 3 has 6 non-attacking pairs, the queen at index 2, value 2 has 5 non-attacking pairs, the queen at index 3, value 7 has 4 non-attacking pairs and it goes on for the remaining queens till the 8th index. The number of non-attacking pairs for each queen in the board is added and fitness is evaluated. Based on those fitness values, probability values are calculated, which are the weighted average of the fitness values and the cumulative probability values.

Selection: Select parents from the current population to create the next generation. Standard selection methods include Roulette Wheel Based Selection, Tournament Selection, and Rank Based Selection. For our implementation, we have used a Roulette Wheel Selection based on the cumulative probability values calculated in the previous step.

3. <u>Crossover:</u> Perform crossover between pairs of parents to create new offspring. Crossover involves exchanging genetic information to create a child solution. The crossover point is randomly generated and at that point, population values between two populations are swapped as seen below.

   Before crossover:

   Suppose that the third element of the array is set as the crossover point. It means all the elements highlighted in blue will remain the same whereas the grey color of both arrays, A and B will exchange the rest of the elements with one another to produce diverse offspring.

3	2	7	5	2	4	1	1

A before Crossover

2	4	7	4	8	5	5	2

B before Crossover

After Crossover:

3	2	7	4	8	5	5	2

A after Crossover

2	4	7	5	2	4	1	1

B after Crossover

4. <u>Mutation:</u> Bring minor changes or mutations to the existing offspring solutions to maintain diversity and explore the search space. In our problem, mutation occurs as a random value swap for another value.

3	2	7	9	8	5	5	2

A after crossover and mutation

2	4	7	5	2	4	8	1

B after crossover and mutation

Why use a Genetic Algorithm for the N-Queen Problem?

I. <u>Efficient Search in Large Solution Spaces</u>: The N-Queen Problem's search space grows factorially with N. A GA's population-based approach allows for parallel exploration of multiple solutions, improving the likelihood of finding valid configurations.

II. <u>Constraint Handling</u>: By encoding solutions as chromosomes, GAs inherently satisfy some constraints. The fitness function then ensures solutions evolve to minimize or eliminate violations of other constraints.

III. Robustness Against Local Optima: The stochastic nature of crossover and mutation helps GAs escape local optima, enabling exploration of diverse regions of the solution space.

Output:

For 4 Queens:

```
Generation 46's maximum fitness score is 6 for the population, [1, 3, 0, 2]
Solution found in generation 46!
```

For 8 Queens:

```
Generation 297's maximum fitness score is 28 for the population, [2, 5, 7, 0, 4, 6, 1, 3]
Solution found in generation 297!
```

Conclusion:

The application of Genetic Algorithms to the N-Queen Problem demonstrates the algorithm's efficacy in solving complex combinatorial challenges. By leveraging biologically inspired operators, GAs can efficiently navigate large solution spaces, evolving populations toward optimal configurations. This approach is scalable and adaptable, capable of addressing the increasing complexity of the problem as *N* grows. The findings affirm that GAs are not only effective for the N-Queen Problem but also broadly applicable to other optimization and search problems with similarly complex constraints.