Evalutionary Computaion

HW1

N-Queen Problem

Ali Eskandarian

40360336

Problem Statement

The N-Queens problem, particularly the 8-Queens variant, presents a classic challenge in combinatorial optimization and artificial intelligence. The objective is to place eight queens on an 8×8 chessboard such that no two queens threaten each other. This means ensuring that no two queens are positioned in the same row, column, or diagonal. The problem can be mathematically defined and analyzed, making it a suitable candidate for evolutionary computation techniques.

Problem Definition

Objective: Place N queens on an n×n chessboard without any two queens attacking each other.
Constraints:

- Each queen must occupy a unique row.
- Each queen must occupy a unique column.
- No two queens can be placed on the same diagonal.

Computational Complexity

The N-Queens problem is known to have 92 distinct solutions for n=8, but as n increases, the complexity grows significantly, with over 22 billion possible configurations for a 25×25 chessboard. The growth of potential solutions emphasizes the need for efficient algorithms to find valid arrangements.

Evolutionary Algorithm Approach

In this assignment, Genetic Algorithm (GA) has implemented to solve the 8-Queens problem. The GA will utilize the following properties:

- Population Initialization: Generate an initial population of potential solutions (i.e., arrangements of queens).
- Fitness Function: Define a fitness function that evaluates how many pairs of queens are attacking each other. The goal is to minimize this value.
- Selection: Use selection methods to choose parent solutions based on their fitness scores.
- Crossover: Implement crossover techniques to combine features of parent solutions and produce offspring.
- Mutation: Introduce mutation to maintain genetic diversity within the population.

Method

Algorithm Implementation

Firstly, The genetic algorithm developed based on specifications in Table 1:

Table 1

Property	Specification
Reoresentation	Permutation
Recombination	Cut-and-Fill Crossover
Mutation	Swap
Parent Selection	Best 2 out of Random 5
Population Size	100
Number of Offsprings	2
Initialization	Random
Termination	Solution found after 10,000 generation

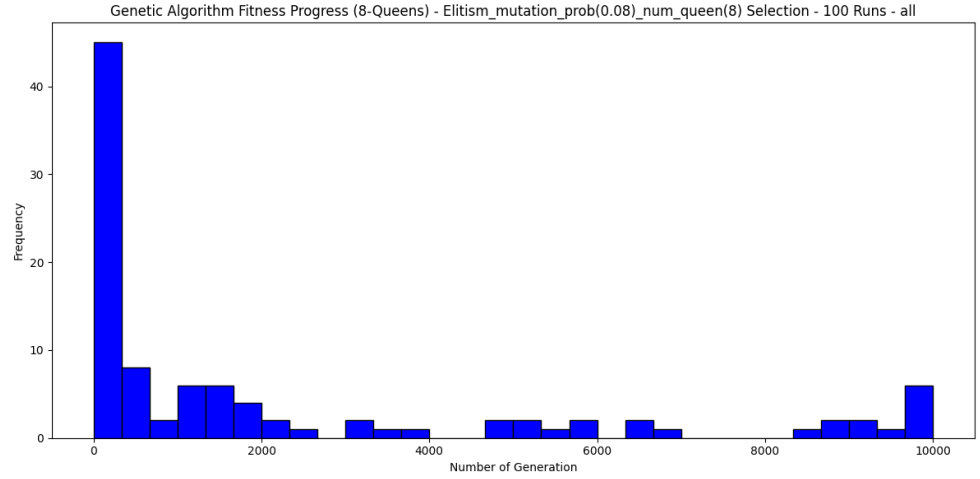
Performance Analysis

We used basic parameters and Elitism Selection to evaluate the algorithm, Solving 8 queen with mutaion probability 0.08 and recombination probability 0.8:

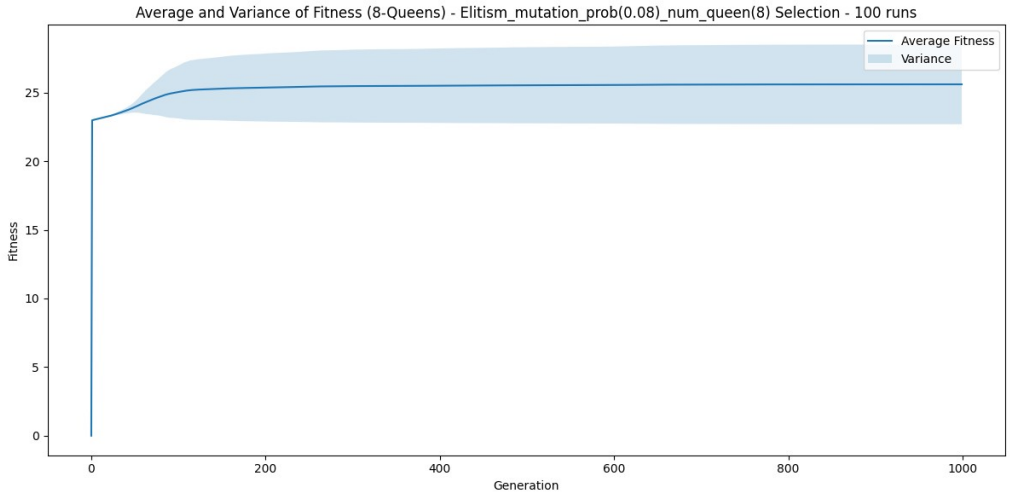
to Evaluate this stochastical model the algorithm runs for 100 times which results are in Table 2:

Average optimal result’s generation	Max optimal result’s generation
2224.28	10000 (Not Found)

Max iteration (to find optimal solution):



Avarage of Solution:



we have a converegence in most cases but still there is anomalies too which will be discussed later in the document.

Parameter Exploration

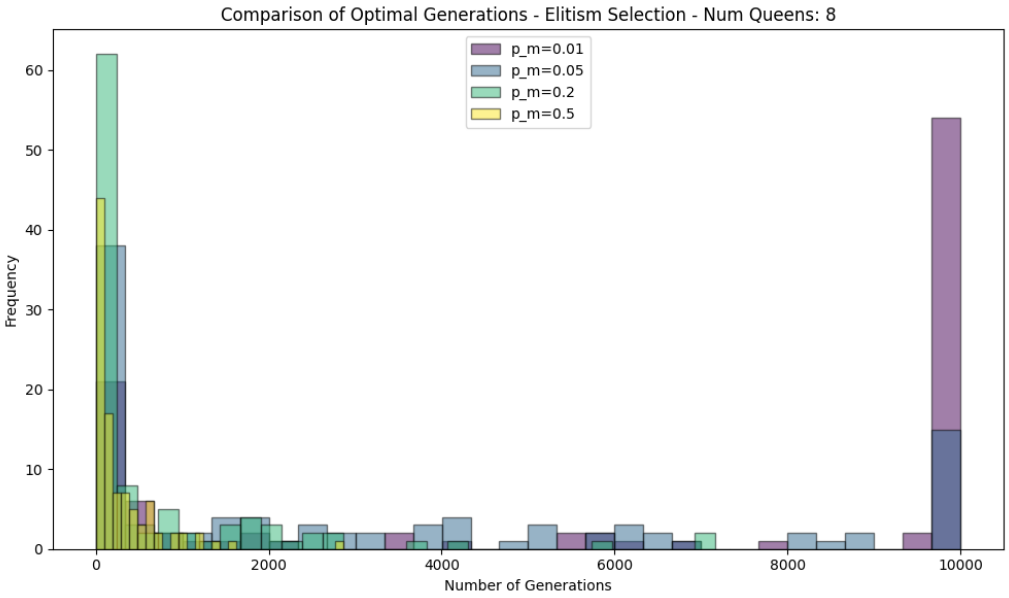
To Explore other parameters, search space has choose as follows:

1. Mutation Probabilities: 0.01, 0.05, 0.2, 0.5
2. Recombination Probabilities: 0.2, 0.4, 0.6, 0.8

results are available in Table 3 are in Elitism method:

Mutation Probability	Average Generation to Optimal	Max Generation to Optimal
0.01	6280.57	10000(Not Found)
0.05	3339.06	10000(Not Found)
0.2	720.53	7166
0.5	285.81	2862

Histograms to shows the performance as visualization:

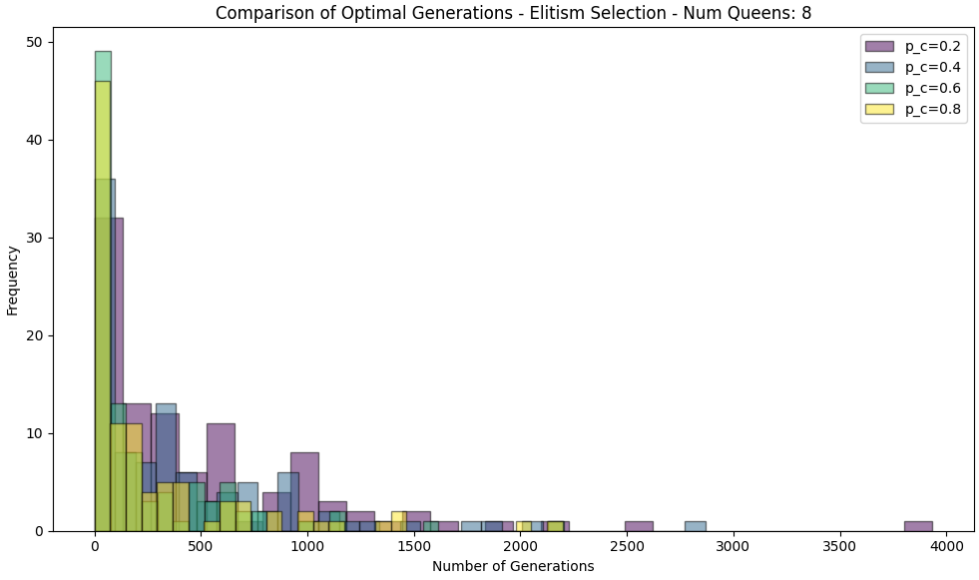


The results indicate that increasing the recombination probability generally leads to improved convergence speed and solution quality. For instance, with a recombination probability of 0.2, the average generation to optimal was 505.05, and the maximum generation to optimal reached 3933. As the probability increased to 0.4, these figures improved to an average of 408.5 and a maximum of 2868. The most notable improvement occurred at a recombination probability of 0.6, where the average generation to optimal dropped significantly to 234.16, and the maximum generation to optimal was 2203. However, at a recombination probability of 0.8, there was a slight increase in both metrics (274.85 average and 2198 maximum), indicating that while higher recombination rates can enhance diversity and exploration, there may be diminishing returns beyond a certain point.

Also we can Explore recombination factor with the best mutation result which is 0.5:

Recombination Probability	Average Generation to Optimal	Max Generation to Optimal
0.2	505.05	3933
0.4	408.5	2868
0.6	234.16	2203
0.8	274.85	2198

Histograms to shows the performance as visualization:



This Results are shown that mutation has way more impact in this problem which is a represented with permutation.

The mutation probability exhibited a more complex relationship with convergence speed and solution quality. At very low mutation probabilities (e.g., 0.01), the average generation to optimal was extremely high at 6280.57, with no solutions found within the maximum limit of 10000 generations. This suggests that insufficient mutation can lead to stagnation and a lack of exploration in the solution space. Increasing the mutation probability to 0.05 still resulted in a high average generation (3339.06) but showed some improvement in finding solutions, albeit still failing to reach optimality within the set limit.

A more effective mutation probability was observed at 0.2, where the average generation to optimal dropped to 720.53, and although the maximum generation remained high at 7166, this indicates a significant improvement in solution quality compared to lower mutation rates. The best performance was achieved with a mutation probability of 0.5, yielding an average generation of only 285.81 and a maximum of 2862. These results highlight that moderate mutation rates can enhance population diversity, enabling faster convergence toward optimal solutions.

In summary, while higher recombination probabilities generally improve convergence speed and solution quality, mutation probabilities must be carefully balanced; too low can hinder progress, while moderate levels appear most effective for fostering diversity and accelerating convergence in solving the N-Queens problem.

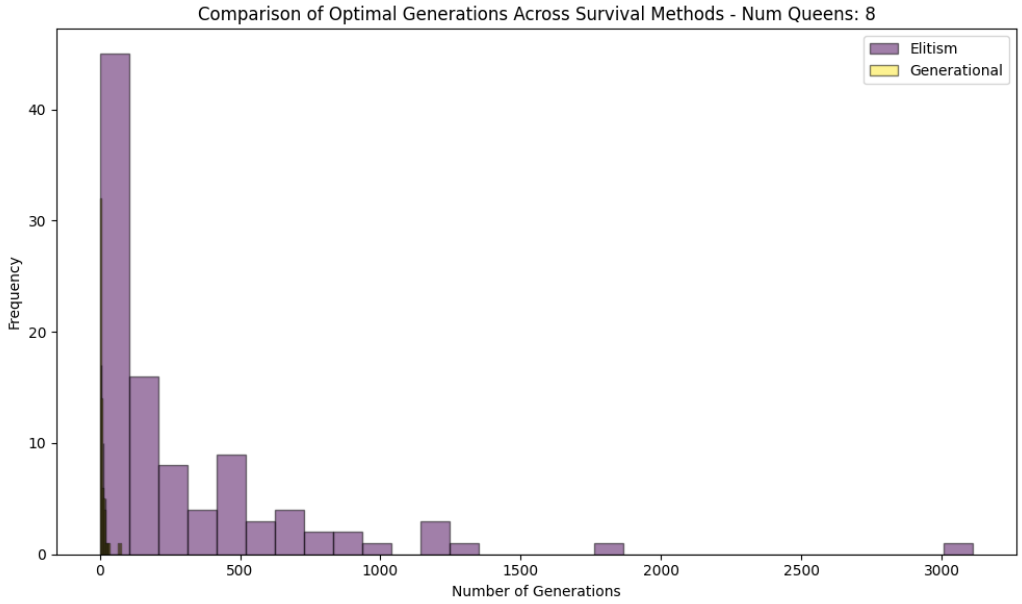
Survival Selection

This experiment has created to explore the selection methods effect on final results, 2 methods has implemented (Generational and Elitism) with previous best results:

- Survival Method: Elitism, Generational

Survival Selection	Average Generation to Optimal	Max Generation to Optimal
Elitism	295.44	3111
Geenerational	9.23	76

Histograms to shows the performance as visualization:



Comparison

The performance of the "Elitism" strategy and the "Generational Replacement" strategy reveals significant differences in their effectiveness for solving the N-Queens problem. The Elitism strategy, which retains a portion of the best-performing individuals from one generation to the next, resulted in an average generation to optimal of 295.44 and a maximum generation to optimal of 3111. In contrast, the Generational Replacement strategy, which replaces the entire population with new offspring at each generation, demonstrated a much more efficient performance with an average generation to optimal of 9.23 and a maximum generation to optimal of 76. These results suggest that Generational Replacement is far superior in terms of convergence speed and overall effectiveness for this particular problem.

Discussion

Analyzing the pros and cons of each survival selection method based on the experimental findings reveals important insights into their operational dynamics. The Elitism strategy, while ensuring that high-quality solutions are preserved across generations, can lead to slower convergence rates, as evidenced by its higher average and maximum generations to optimal. This may result from a lack of diversity in the population, potentially causing premature convergence on suboptimal solutions.

On the other hand, the Generational Replacement strategy fosters greater diversity by introducing entirely new offspring in each generation. This approach not only accelerates convergence but also allows for a broader exploration of the solution space, which is crucial when tackling complex problems like N-Queens. However, one potential downside is that it may sometimes discard high-quality solutions entirely if they are not part of the new offspring.

In conclusion, while Elitism offers stability by retaining top performers, Generational Replacement provides a more dynamic and effective method for finding optimal solutions in this context. The experimental results clearly favor Generational Replacement for its efficiency and adaptability in solving the N-Queens problem.

Scalability Testing

Now with our best results we can test scalability of the algorithm:

Recombination Probability	Average Generation to Optimal	Max Generation to Optimal
8	6.79	41
10	76.1	277
12	475.63	2430
20	2300.1	8700

As we can see algorithm can find the results but due to very high search space it lose its speed for more involved queens.

Critical Evaluation

Strengths and Weaknesses

The implementation of the Genetic Algorithm (GA) for solving the N-Queens problem demonstrates several strengths, particularly in its ability to efficiently explore a vast solution space and converge on optimal or near-optimal solutions. The use of strategies such as Elitism and Generational Replacement allows for a balance between preserving high-quality solutions and introducing diversity, which is crucial for avoiding local optima. Additionally, the flexibility of adjusting parameters like mutation and recombination probabilities enables fine-tuning of the algorithm to improve performance based on specific problem instances. However, the GA also has limitations; for instance, it can be sensitive to parameter settings, and inappropriate choices may lead to slow convergence or failure to find optimal solutions. Furthermore, the reliance on stochastic processes means that results can vary between runs, which may impact the reproducibility of findings.

Improvement Suggestions

To enhance the performance of the GA implementation, several potential improvements can be considered. First, exploring alternative selection methods, such as tournament selection or rank-based selection, could provide more robust mechanisms for choosing parents and maintaining diversity within the population. Additionally, experimenting with different crossover techniques—such as uniform crossover or two-point crossover—might yield better offspring by combining genetic material more effectively. Incorporating adaptive mechanisms that dynamically adjust mutation and recombination rates based on the current state of the population could also improve convergence speed and solution quality. Finally, integrating hybrid approaches that combine GAs with local search techniques (e.g., hill climbing) may further refine solutions by exploiting promising areas of the search space more thoroughly. These enhancements could lead to a more efficient and effective GA tailored to solving complex optimization problems like N-Queens.

Conclusion

The findings from these experiments underscore the effectiveness of evolutionary algorithms, particularly Genetic Algorithms, in solving combinatorial optimization problems like the N-Queens problem. The ability to explore vast solution spaces while balancing exploitation and exploration through mechanisms such as selection strategies and parameter adjustments enhances the algorithm's performance. The superior results achieved with Generational Replacement and optimal mutation and recombination probabilities highlight the importance of adaptive strategies in evolutionary computation. Overall, these experiments demonstrate that evolutionary algorithms are not only capable of finding solutions to complex problems but can also be fine-tuned for improved efficiency and effectiveness, making them a valuable tool in the field of optimization

Code

Code is utilizing OOP and have a solver class which provide functions to implement the GA algorithm