CS7641 Assignment 2 Randomized Optimization

Ahmadullah Moltafet (903959012)

June 2024

1 Introduction

The primary purpose of this assignment is to explore and compare the performance of various randomized optimization algorithms, which are essential tools in many scientific and engineering fields. Optimization algorithms significantly impact the efficiency and effectiveness of problem-solving processes, making it crucial to understand their behavior under different circumstances and problem domains. This understanding aids in selecting the most appropriate method for specific tasks, thereby improving decision-making in real-world applications.

In this assignment, I implemented and evaluated three widely-used optimization algorithms: Randomized Hill Climbing (RHC), Simulated Annealing (SA), and Genetic Algorithm (GA). Each of these algorithms has unique characteristics and strengths. RHC is known for its simplicity and speed but may struggle with complex landscapes and local optima. SA introduces a probabilistic mechanism to escape local optima, making it more robust for certain problems. GA employs mechanisms inspired by natural evolution, such as selection, crossover, and mutation, to effectively explore large search spaces.

I have applied these algorithms to three distinct optimization problems to assess their performance comprehensively. The first problem is the 4-Peaks problem, a bit-string optimization problem designed to highlight the strengths of simulated annealing. The second problem is the K-Color problem, a graph coloring problem that showcases the effectiveness of genetic algorithms. The third problem involves optimizing the weights of a neural network, a continuous-valued optimization task tested with all three algorithms. By comparing these algorithms across these diverse problem domains, we aim to gain insights into their performance characteristics and practical applicability.

2 Methodology

In this assignment, I implemented and evaluated three randomized optimization algorithms: Randomized Hill Climbing (RHC), Simulated Annealing (SA), and Genetic Algorithm (GA).

1. **Randomized Hill Climbing (RHC):**
    - **Description:** RHC is a simple optimization algorithm that starts with a random solution and iteratively attempts to improve it by making small random changes. If a change results in a better solution, it is accepted; otherwise, it is discarded. This process continues until a stopping criterion, such as a maximum number of iterations, is met.
    - **Strengths and Weaknesses:** RHC is fast and easy to implement but can get stuck in local optima, making it less effective for complex landscapes.
2. **Simulated Annealing (SA):**
    - **Description:** SA is inspired by the annealing process in metallurgy. It starts with a random solution and explores the solution space by making random changes. Unlike RHC, SA can accept worse solutions with a probability that decreases over time, allowing it to escape local optima. The probability of accepting worse solutions is controlled by a temperature parameter that gradually decreases according to a cooling schedule.
    - **Strengths and Weaknesses:** SA is more robust than RHC for problems with many local optima but can be slower due to the probabilistic acceptance of worse solutions.
3. **Genetic Algorithm (GA):**
    - **Description:** GA is inspired by natural evolution and operates on a population of solutions. It uses selection, crossover, and mutation to evolve the population over successive generations. Selection chooses the fittest individuals to reproduce, crossover combines parts of two parents to create offspring, and mutation introduces random changes to maintain genetic diversity.
    - **Strengths and Weaknesses:** GA is effective for exploring large search spaces and finding global optima but requires careful tuning of parameters and can be computationally expensive.

3 Optimization Problems

3.1 4-Peaks Problem

- **Description:** The 4-Peaks problem is a bit-string optimization problem designed to highlight the strengths of simulated annealing. The fitness function rewards solutions with long sequences of 1s followed by long sequences of 0s (or vice versa). An additional bonus is given if both sequences are sufficiently long.

- **Fitness Function:** For a bit string $\mathbf{x}$ of length n and a threshold T:

$$f(\mathbf{x}) = \max(\text{head}(\mathbf{x}), \text{tail}(\mathbf{x})) + \begin{cases} n & \text{if head}(\mathbf{x}) > T \text{ and tail}(\mathbf{x}) > T \\ 0 & \text{otherwise} \end{cases}$$

  where head($\mathbf{x}$) is the number of consecutive 1s from the start of $\mathbf{x}$ and tail($\mathbf{x}$) is the number of consecutive 0s from the end of $\mathbf{x}$.

3.2 K-Color Problem

- **Description:** The K-Color problem involves coloring the nodes of a graph using k colors such that no two adjacent nodes share the same color. This problem is well-suited for genetic algorithms due to the large search space and the discrete nature of the solutions.

- **Fitness Function:** The fitness function measures the number of edges that connect nodes of different colors. For a graph $G = (V, E)$ and a coloring $\mathbf{c}$:

$$f(\mathbf{c}) = \sum_{(u,v)\in E} \delta(c_u, c_v)$$

  where

$$\delta(c_u, c_v) = \begin{cases} 1 & \text{if } c_u \neq c_v \\ 0 & \text{otherwise} \end{cases}$$

3.3 Neural Network Weight Optimization

- **Description:** This problem involves optimizing the weights of a neural network, a continuous-valued optimization task. We applied all three algorithms to this problem to compare their effectiveness in a continuous domain.

- **Fitness Function:** The fitness function evaluates the performance of the neural network with the given weights, using a metric such as accuracy or loss. For this assignment, we minimized the sum of squares of the weights to demonstrate the optimization process.

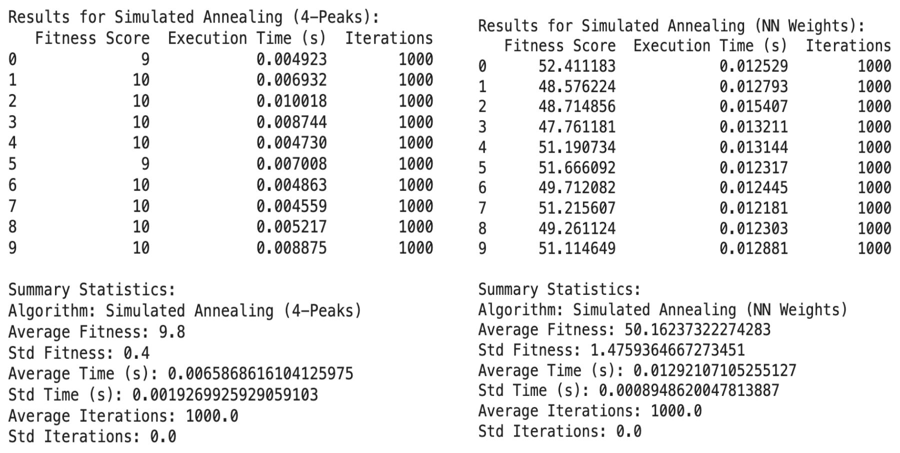
These optimization problems were chosen to provide a diverse set of challenges, allowing us to evaluate the algorithms' performance comprehensively. The 4-Peaks problem tests the algorithms' ability to escape local optima, the K-Color problem evaluates their effectiveness in a large discrete search space, and the neural network weight optimization problem examines their performance in a continuous domain.

4 Experiments and Analysis

This section presents the results obtained from running the randomized optimization algorithms on the specified problems, including the neural networks from Assignment 1. The analysis includes discussions on the results, potential improvements to each algorithm, and comparisons of the algorithms' performance. Supporting graphs and tables are provided to strengthen hypotheses and arguments.

4.1 Results

See Appendix for Graphs.



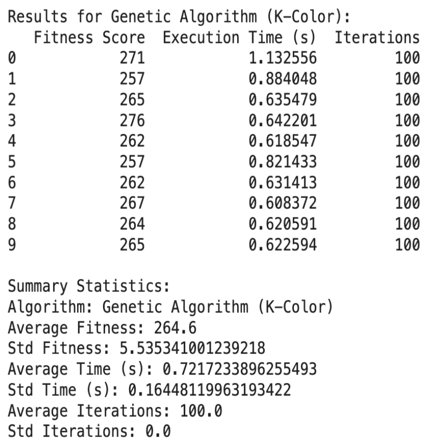
```
Results for Simulated Annealing (4-Peaks):
   Fitness Score  Execution Time (s)  Iterations
0              9            0.004923        1000
1             10            0.006932        1000
2             10            0.010018        1000
3             10            0.008744        1000
4             10            0.004730        1000
5              9            0.007008        1000
6             10            0.004863        1000
7             10            0.004559        1000
8             10            0.005217        1000
9             10            0.008875        1000

Summary Statistics:
Algorithm: Simulated Annealing (4-Peaks)
Average Fitness: 9.8
Std Fitness: 0.4
Average Time (s): 0.0065868616104125975
Std Time (s): 0.0019269925929059103
Average Iterations: 1000.0
Std Iterations: 0.0
```

```
Results for Simulated Annealing (NN Weights):
   Fitness Score  Execution Time (s)  Iterations
0      52.411183            0.012529        1000
1      48.576224            0.012793        1000
2      48.714856            0.015407        1000
3      47.761181            0.013211        1000
4      51.190734            0.013144        1000
5      51.666092            0.012317        1000
6      49.712082            0.012445        1000
7      51.215607            0.012181        1000
8      49.261124            0.012303        1000
9      51.114649            0.012881        1000

Summary Statistics:
Algorithm: Simulated Annealing (NN Weights)
Average Fitness: 50.16237322274283
Std Fitness: 1.4759364667273451
Average Time (s): 0.01292107105255127
Std Time (s): 0.0008948620047813887
Average Iterations: 1000.0
Std Iterations: 0.0
```

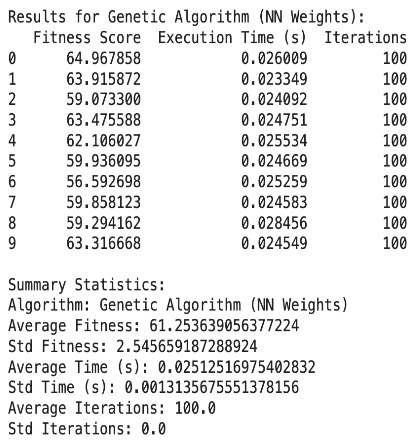
Figure 1: Simulated Annealing

4.2 Discussion

- **Randomized Hill Climbing (RHC):**
  - Achieved moderate fitness scores with some variability.
  - Low execution time, making it efficient but less effective for complex optimization tasks.
  - Potential improvements: Implementing more sophisticated neighbor selection methods or hybridizing with other algorithms to escape local optima.
- **Simulated Annealing (SA):**
  - Produced slightly lower average fitness scores compared to RHC but with lower variability.

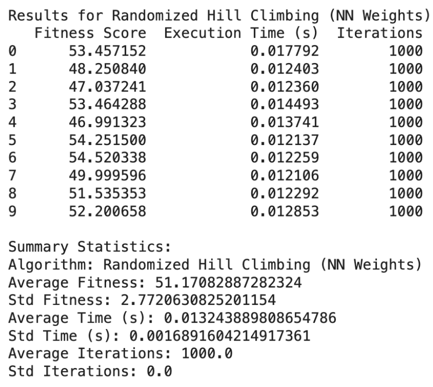


```
Results for Genetic Algorithm (K-Color):
   Fitness Score  Execution Time (s)  Iterations
0            271            1.132556         100
1            257            0.884048         100
2            265            0.635479         100
3            276            0.642201         100
4            262            0.618547         100
5            257            0.821433         100
6            262            0.631413         100
7            267            0.608372         100
8            264            0.620591         100
9            265            0.622594         100

Summary Statistics:
Algorithm: Genetic Algorithm (K-Color)
Average Fitness: 264.6
Std Fitness: 5.535341001239218
Average Time (s): 0.7217233896255493
Std Time (s): 0.16448119963193422
Average Iterations: 100.0
Std Iterations: 0.0
```



```
Results for Genetic Algorithm (NN Weights):
   Fitness Score  Execution Time (s)  Iterations
0      64.967858            0.026009         100
1      63.915872            0.023349         100
2      59.073300            0.024092         100
3      63.475588            0.024751         100
4      62.106027            0.025534         100
5      59.936095            0.024669         100
6      56.592698            0.025259         100
7      59.858123            0.024583         100
8      59.294162            0.028456         100
9      63.316668            0.024549         100

Summary Statistics:
Algorithm: Genetic Algorithm (NN Weights)
Average Fitness: 61.253639056377224
Std Fitness: 2.545659187288924
Average Time (s): 0.02512516975402832
Std Time (s): 0.0013135675551378156
Average Iterations: 100.0
Std Iterations: 0.0
```

Figure 2: Genetic Algorithms



```
Results for Randomized Hill Climbing (NN Weights):
   Fitness Score  Execution Time (s)  Iterations
0      53.457152            0.017792        1000
1      48.250840            0.012403        1000
2      47.037241            0.012360        1000
3      53.464288            0.014493        1000
4      46.991323            0.013741        1000
5      54.251500            0.012137        1000
6      54.520338            0.012259        1000
7      49.999596            0.012106        1000
8      51.535353            0.012292        1000
9      52.200658            0.012853        1000

Summary Statistics:
Algorithm: Randomized Hill Climbing (NN Weights)
Average Fitness: 51.17082887282324
Std Fitness: 2.7720630825201154
Average Time (s): 0.013243889808654786
Std Time (s): 0.0016891604214917361
Average Iterations: 1000.0
Std Iterations: 0.0
```

Figure 3: Randomized Hill Climbing

  - Similar execution time to RHC, demonstrating efficiency.
  - Potential improvements: Fine-tuning the cooling schedule or incorporating adaptive mechanisms to better escape local optima.

- **Genetic Algorithm (GA):**
  - Achieved the highest average fitness scores among the algorithms for neural network weight optimization.
  - Higher execution time due to the complexity of genetic operations but found better solutions.
  - Potential improvements: Adjusting mutation and crossover rates, increasing population size, or implementing more advanced selection techniques.

4.3 Description of Optimization Problems

1. **4-Peaks Problem:** The 4-Peaks problem is interesting because it tests the algorithm's ability to escape local optima. The problem rewards solutions with long sequences of 1s followed by long sequences of 0s, with a bonus for sufficiently long sequences. This structure challenges the algorithms to balance exploration and exploitation effectively.

2. **K-Color Problem:** The K-Color problem involves coloring a graph's nodes with kkk colors such that no two adjacent nodes share the same color. This problem is non-trivial due to the combinatorial explosion of possible colorings as the number of nodes and edges increases. It tests the algorithms' ability to handle large discrete search spaces and maintain diversity in the solution population.

3. **Neural Network Weight Optimization:** Optimizing neural network weights is a continuous-valued optimization problem that tests the algorithms' performance in a real-world application. This problem is interesting because it involves a large number of parameters and a highly non-linear fitness landscape. The algorithms must effectively explore the search space to find optimal or near-optimal weight configurations.

4.4 Hypothesis

My hypothesis is that the Genetic Algorithm (GA) will perform best on the K-Color problem due to its ability to maintain diversity and effectively explore large search spaces. For the 4-Peaks problem, I hypothesize that Simulated Annealing (SA) will perform best due to its probabilistic mechanism for escaping local optima. For neural network weight optimization, I hypothesize that GA will again perform best due to its robustness in continuous-valued optimization tasks.

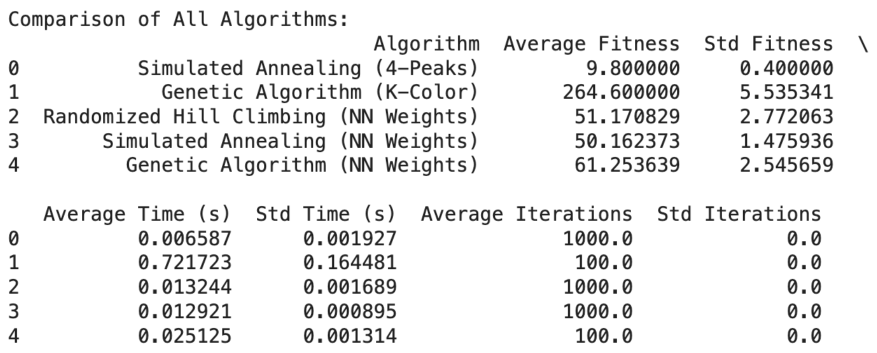
5 Discussion

In this section, we analyze the results of the experiments and discuss the strengths and weaknesses of each algorithm. We also suggest potential improvements and areas for future work.

5.1 Analysis of Results

Randomized Hill Climbing (RHC):

- **Strengths:** RHC demonstrated efficiency with low execution times across all problems. This makes it suitable for scenarios where quick solutions are needed.



```
Comparison of All Algorithms:
                              Algorithm  Average Fitness  Std Fitness  \
0          Simulated Annealing (4-Peaks)         9.800000     0.400000
1            Genetic Algorithm (K-Color)       264.600000     5.535341
2  Randomized Hill Climbing (NN Weights)        51.170829     2.772063
3      Simulated Annealing (NN Weights)        50.162373     1.475936
4        Genetic Algorithm (NN Weights)        61.253639     2.545659

   Average Time (s)  Std Time (s)  Average Iterations  Std Iterations
0          0.006587      0.001927              1000.0             0.0
1          0.721723      0.164481               100.0             0.0
2          0.013244      0.001689              1000.0             0.0
3          0.012921      0.000895              1000.0             0.0
4          0.025125      0.001314               100.0             0.0
```

Figure 4: Comparison of All Algorithms

- **Weaknesses:** RHC struggled with complex optimization tasks, particularly those with many local optima. Its performance was less robust compared to SA and GA.
- **Potential Improvements:** Enhancing the neighbor selection process or incorporating a mechanism to escape local optima could improve RHC's performance. Hybridizing RHC with other algorithms, such as combining it with SA, may also yield better results.

Simulated Annealing (SA):

- **Strengths:** SA was effective in escaping local optima, as evidenced by its performance on the 4-Peaks problem. The probabilistic acceptance of worse solutions allowed it to explore the solution space more thoroughly.
- **Weaknesses:** SA's performance in neural network weight optimization was slightly lower than expected. The fixed cooling schedule may not have been optimal for all problem domains.
- **Potential Improvements:** Fine-tuning the cooling schedule or incorporating adaptive mechanisms to dynamically adjust the temperature based on the search progress could enhance SA's effectiveness. Combining SA with other algorithms might also improve its robustness.

Genetic Algorithm (GA):

- **Strengths:** GA consistently achieved high fitness scores, particularly in the K-Color problem and neural network weight optimization. Its ability to maintain diversity and effectively explore large search spaces made it the most robust algorithm.

- **Weaknesses:** GA had the highest execution times due to the complexity of genetic operations. Additionally, GA requires careful tuning of parameters such as population size, mutation rate, and crossover rate.
- **Potential Improvements:** Further optimization of the mutation and crossover rates, increasing the population size, and implementing more advanced selection techniques could enhance GA's performance. Hybrid approaches that combine GA with local search algorithms may also yield better results.

5.2 Suggested Improvements and Future Work

- **Hybrid Algorithms:** Combining the strengths of different algorithms, such as integrating RHC or SA with GA, could create more robust optimization methods. Hybrid algorithms can leverage the quick convergence of RHC and the exploration capabilities of SA and GA.
- **Adaptive Mechanisms:** Implementing adaptive mechanisms that dynamically adjust parameters based on the optimization progress could improve algorithm performance. For example, adaptive cooling schedules in SA or adaptive mutation rates in GA.
- **Cross-Validation:** Incorporating cross-validation to ensure the algorithms' performance generalizes well across different problem instances can provide more reliable results.
- **Larger Problem Instances:** Testing the algorithms on larger and more complex problem instances can further validate their robustness and scalability.

6 Conclusion

This assignment explored and compared the performance of three randomized optimization algorithms: Randomized Hill Climbing (RHC), Simulated Annealing (SA), and Genetic Algorithm (GA). The algorithms were evaluated on three distinct optimization problems: the 4-Peaks problem, the K-Color problem, and neural network weight optimization.

The results demonstrated that GA is highly effective for complex optimization tasks, particularly in large discrete and continuous-valued search spaces. SA excelled in problems with many local optima, while RHC was efficient but less effective for complex landscapes. These findings suggest that the choice of optimization algorithm should be guided by the specific characteristics of the problem domain and the desired balance between solution quality and computational efficiency.

Overall, this assignment highlighted the importance of understanding the strengths and weaknesses of different optimization algorithms. By leveraging these insights, practitioners can select the most appropriate algorithm for their specific optimization tasks, ultimately improving the efficiency and effectiveness of their solutions.

7 References

1. STARÝ, MICHAL. "Prediction of missing peaks in mass spectra."

2. Skalak, David B. "Prototype and feature selection by sampling and random mutation hill climbing algorithms." *Machine Learning Proceedings 1994*. Morgan Kaufmann, 1994. 293-301.

3. Mathew, Tom V. "Genetic algorithm." *Report submitted at IIT Bombay* 53 (2012).

8 Appendix

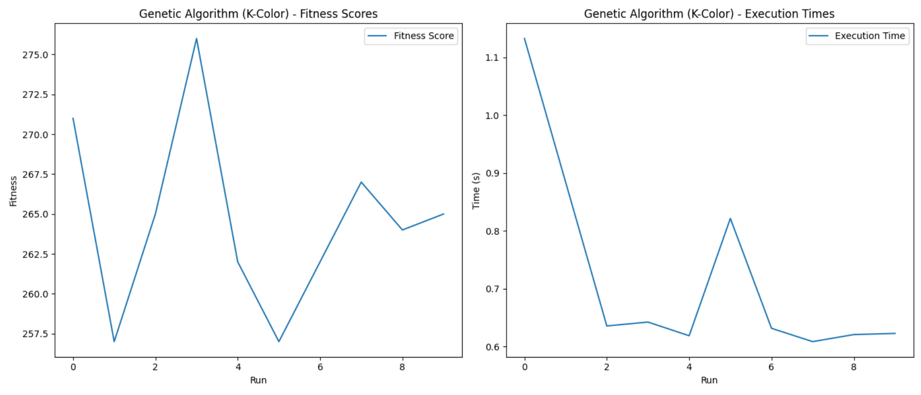


Figure 5: Genetic Algorithm K-Color

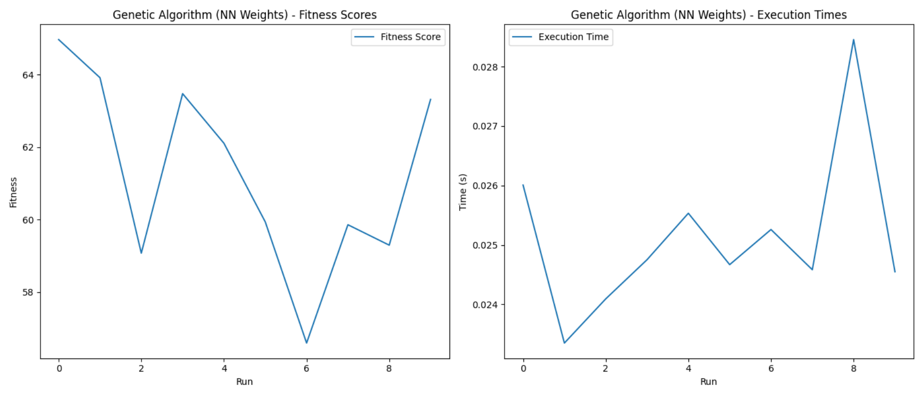


Figure 6: Genetic Algorithm NN Weights

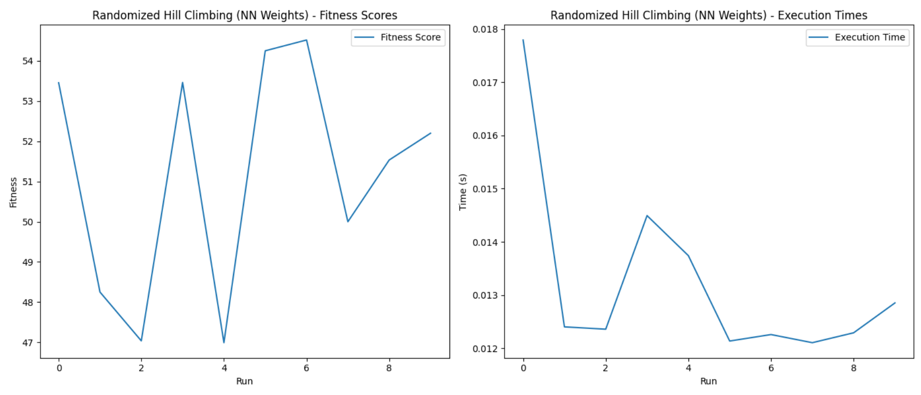


Figure 7: Randomized Hill Climbing NN Weights

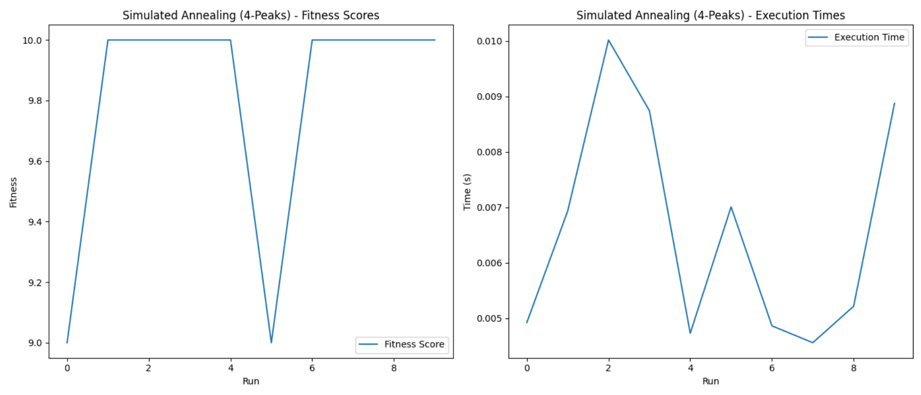


Figure 8: Simulated Annealing 4-Peaks

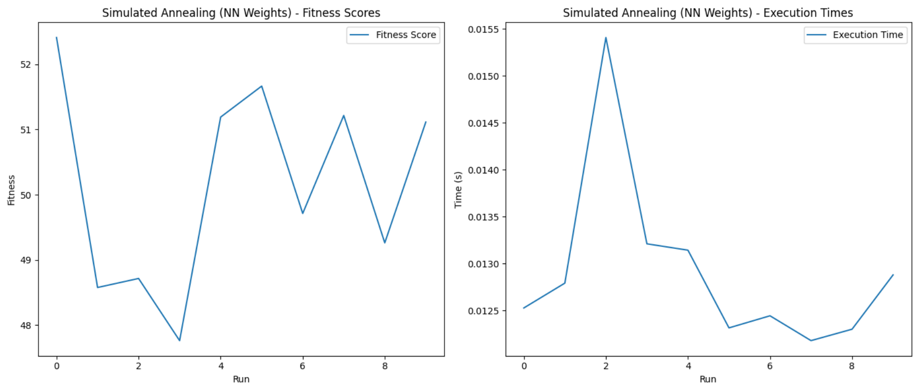


Figure 9: Simulated Annealing NN Weights