# CS-7641
# Assignment 2- Randomized Optimization

*Nayeem Aquib*

## 1 INTRODUCTION

In this report, four local random search algorithms- Randomized Hill Climbing, Simulated annealing, a genetic algorithm, and MIMIC, will be used for three optimization problems- one max problem, deceptive trap, and four peaks using two different datasets. The first problem will highlight advantages of the genetic algorithm, the second of simulated annealing, and the third of MIMIC. Both of these datasets were acquired from Kaggle. One is a dataset of videogame sales across different platforms and different regions. Another is a dataset of college applicants with different metrics from their applications including the decision of the college.

## 2 OPTIMIZATION PROBLEMS

### 2.1 OneMax Problem

#### 2.1.1 Description

The goal is to maximize the number of ones in a binary string. Binary strings are used to represent candidate solutions, where each bit can be either 0 or 1. The fitness of a candidate solution is equal to the number of ones in the binary string. Therefore, the higher the number of ones, the better the fitness.

#### 2.1.2 Why is it interesting

It may sound straightforward, but it's like the "Hello World" of optimization problems. It helps us understand how different strategies work when we're just starting out. Plus, it's a fun way to see how fast we can get to the best solution.

### 2.2 Deceptive Trap Problem

#### 2.2.1 Description

The objective is to maximize the fitness of a binary string where every consecutive group of k bits evaluates to a deceptive trap. In other words, the global optimum is not the sequence of all ones, but rather a sequence where each k-bit group evaluates to zero. Similar to the OneMax problem, binary strings are used as representations. The fitness function is a deceptive trap function. It evaluates each consecutive group of k bits and assigns a fitness value. If all bits in a group are 1, the fitness is 0; otherwise, the fitness is (k - 1). The objective is to maximize this deceptive fitness.

**2.2.2 Why is it interesting**
This problem throws a curveball because the traps trick you into thinking you're going the right way when you're not. It's like trying to solve a puzzle where the pieces keep changing shape. It's interesting because it challenges us to think outside the box and find creative solutions.

## 2.3 Four Peaks Problem

### 2.3.1 Description

The goal is to maximize the fitness of a binary string with both linear and deceptive components, including two local optima. It presents a challenge because it has a combination of a linear component (increasing fitness with the number of ones) and a deceptive component (local optima that are not the global optimum). Binary strings are used as representations, similar to the previous problems. The fitness function is a combination of a linear component and a deceptive component. The linear component rewards the number of ones in the string, while the deceptive component introduces two local optima where the fitness is zero. Therefore, the objective is to find a balance between maximizing the number of ones and avoiding the deceptive optima.

### 2.3.2 Why is it interesting

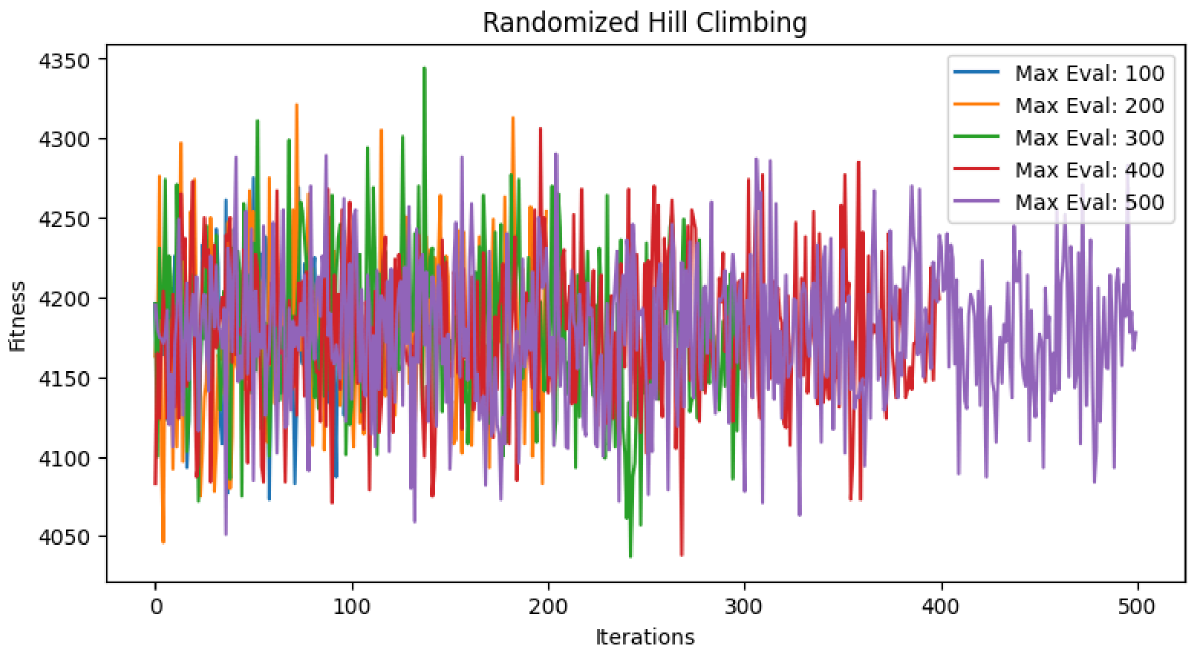
This problem has a balancing act between the linear and the deceptive component which I find useful to learn to strengthen the foundation.

# 3 THE OPTIMIZATION ALGORITHMS ANALYSIS

## 3.1 OneMax Problem

### 3.1.1 Randomized Hill Climbing

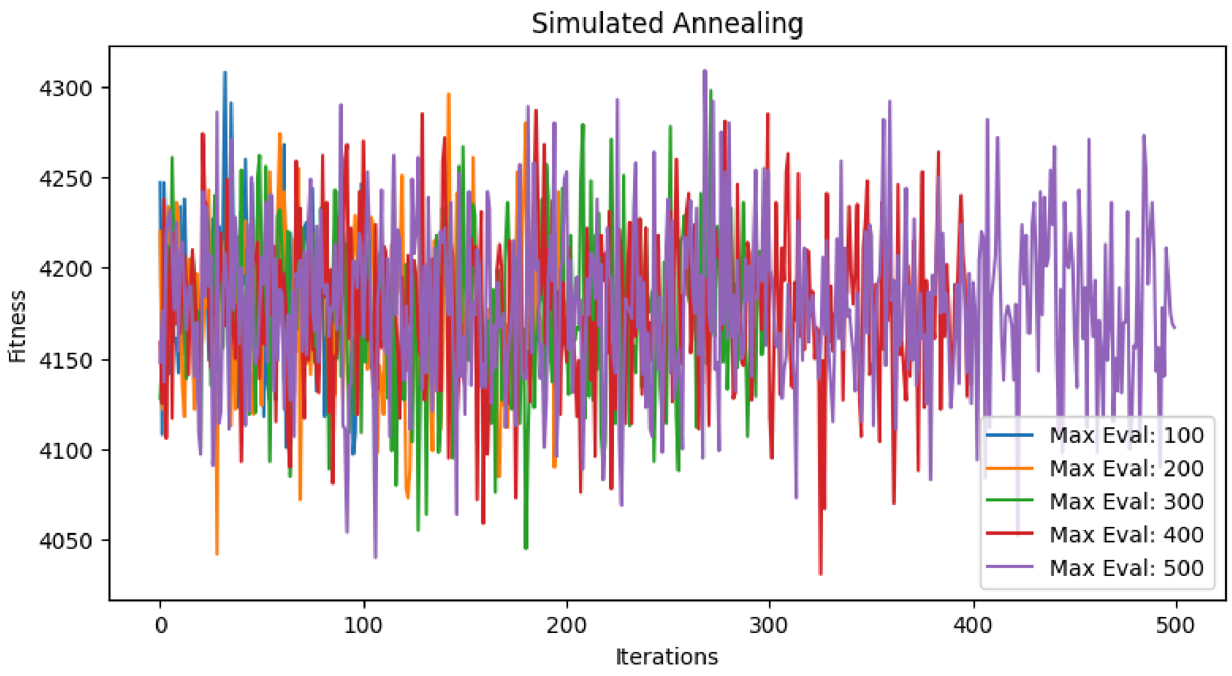
I first started with the OneMax problem. For max_evaluation and max_generation value of 1000 it took 21 seconds on the college admissions dataset and 62 seconds on the video game sales dataset to run. Then I experimented with limiting the maximum evaluations to different numbers(100, 200, 300, 400, 500) and I got the following result.



Here each line seems to reach a plateau. This is probably because the algorithm quickly finds a local optimum and then makes no further progress. This is problematic if the search space has many local maximas because then it can get stuck at a local optima. Also it doesn't show a clear trend of improvement over several iterations. This is probably because it depends on randomly generated solutions that can be better or worse than the current solution.

### 3.1.2 Simulated Annealing

In this case experimenting with different max evaluations gave me the following result.



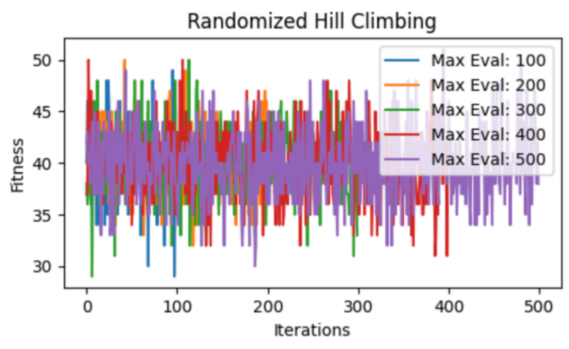
The fitness levels here vary across different runs but don't show significant overall improvement as the maximum number of evaluations increases. This is probably due to possible convergence to local optima or that the temperature reduction in simulated annealing is not allowing the algorithm to escape local optima effectively.

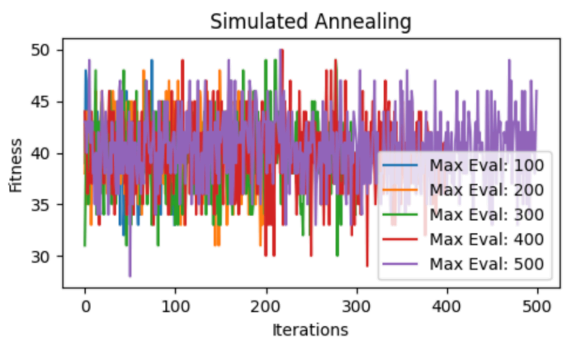
### 3.1.3 Genetic Algorithm and MIMIC

In the case of genetic algorithm and MIMIC, I experimented with different population sizes over a fixed number of iterations(100) and I got the following result which was very surprising since it suggests that either it is converging to local optima extremely quickly.

## 3.2 Deceptive Trap Problem

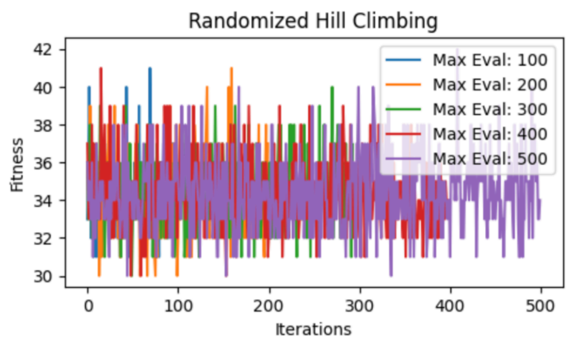
I, then, worked with the deceptive trap problem where for the same number of max evaluations with 4 neighbors it took 280 seconds to run on the college admission dataset and 567 seconds to run on the video game sales dataset. After some experimentation with max validation and population size I got the following result.

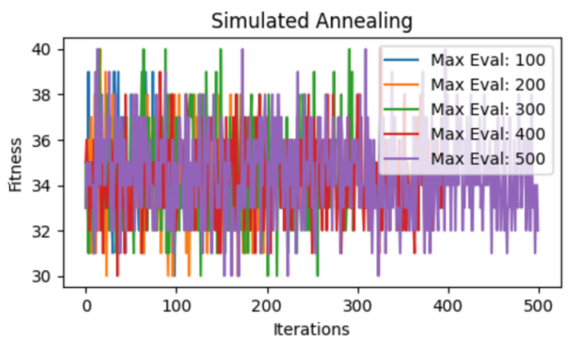




### 3.3 Four Peaks Problem

Finally, for the four peaks problem, with the same number of max evaluations, it took 142 seconds to run on the college admission dataset and 273 seconds to run on the video game sales dataset. After some experimentation with max_eval I got the following graphs:





## 4 Time and Space Complexity

### 4.1.1 Randomized Hill Climbing

Space Complexity: O(n), where n is the length of the bitstring (the size of the problem).

Time Complexity: O(max_evaluations * n), where max_evaluations is the maximum number of evaluations allowed and n is the length of the bitstring. In each evaluation, a neighbor is generated, which requires O(n) operations, so the overall time complexity is linear in the number of evaluations and the size of the problem.

### 4.1.2 Simulated Annealing

Space Complexity: O(n), similar to Randomized Hill Climbing.

Time Complexity: O(max_evaluations * n), similar to Randomized Hill Climbing. However, the cooling schedule might lead to fewer evaluations being performed, potentially reducing the total time.

### 4.1.3 Genetic Algorithm:

Space Complexity: O(n * population_size), where n is the length of the bitstring and population_size is the size of the population.

Time Complexity: O(max_generations * population_size * n), where max_generations is the maximum number of generations, population_size is the size of the population, and n is the length of the bitstring. Each generation involves operations on the entire population, including crossover and mutation, which requires O(n) operations per individual.

### 4.1.4 MIMIC Algorithm

Space Complexity: O(n * population_size), similar to the Genetic Algorithm.

Time Complexity: O(max_generations * population_size * n * log(population_size)), where max_generations is the maximum number of generations, population_size is the size of the population, and n is the length of the bitstring. The time complexity includes building the probability distribution and sampling from it, which requires sorting the population and constructing histograms, resulting in a higher time complexity compared to the Genetic Algorithm.

## 5 CONCLUSION

Overall, it seems that in case of OneMax problem the algorithms capable of exploring the search space efficiently, such as Genetic Algorithm and MIMIC, can quickly converge to the optimal solution by incrementally improving candidate solutions. But Randomized Hill Climbing may struggle to escape local optima, especially if the search space is large. Simulated Annealing may also require careful tuning of parameters to balance exploration and exploitation effectively.

In case of the deceptive trap problem, the Genetic Algorithm and MIMIC, with their population-based approaches and diversity maintenance mechanisms, did great and that is probably due to their ability to explore multiple solutions simultaneously. Randomized Hill Climbing and Simulated Annealing may struggle to overcome the deceptive nature of the fitness landscape, potentially getting stuck in suboptimal solutions.
In case of the four peak problem, the Genetic Algorithm and MIMIC, with their ability to

maintain diversity and explore the search space effectively, can overcome the challenges posed by the deceptive optima and converge to the global optimum.

So after the experimentation and analysis my general feeling is that while Randomized Hill Climbing can be a good choice for a quick and dirty optimization, it might not be the best choice for finding the global optimum, especially for problems with many local optima. Genetic algorithm on the other hand can require some fine tuning, but can be very effective at finding the overall best solution.