Georgia Institute of Technology

CS 7641: Machine Learning

# Assignment 2 – Randomized Optimization

## 1. Implement four local random search algorithms on three optimization problems

## 2. Find good weights for a neural network

Name: Tianyu Yang
GTid: 903645962
Date: 2020/10/11

## 1. Abstract

In this Randomized Optimization project, we will finish two tasks. The first one is to implement four local random search algorithms, including randomized hill climbing, simulated annealing, a genetic algorithm and MIMIC. They are all popular optimization algorithms in machine learning. Here we will run the algorithms in three optimization problems, such as four peaks, one max and Knapsack. What's more, I will apply the optimization algorithms to the problem in my first assignment which is to predict the result of the NBA games using a Neural Networks model to find optimal weights.

## 2. Introduction

In this project, I will cover three optimization problems to evaluate four optimization algorithms. Mlrose is a popular Python package which is to apply some of the most common randomized optimization and search algorithms. It also has some default optimized problems over both discrete and continuous-valued parameter spaces so that we can use it in this project. Iteration for each setting in grid search is 5. We will compare with these four algorithms from different factors, such as the fitness performance and searching time.

The next part of this project is to find how Randomized Hill Climbing, Simulated Annealing and a Genetic algorithm applied on my first assignment datasets, NBA games to find the best optimal weights for the neural networks model. I use two different activation function for the training model, Relu and Sigmoid. To analyze the performance of our optimization algorithm, we use training and testing fitting time to judge the performance of randomized optimization. What's more, I also calculate the Jaccard Index of the evaluation and get the table for the accuracy score and F1 score for each method.

The last part is the conclusion section. In this section, I will analyze the result from the program and summary the results of the comparison of the four randomized optimization algorithms.

## 3. Methodology

### 3.1 Randomized Hill Climbing

Randomized Hill Climbing is a popular randomized optimization algorithm which is used for getting the best global optimum. The method is to move towards the next higher elevation neighbor until it reaches the peak. It is a kind of mathematical optimization for local search which is to randomize from its starting position and then attempt to find another position by making an increasing change to the problem. After that, if that change has a better solution to the problem, this algorithm will continuously change the increasing value to make it closer to the problem until there is no further improvement it can make the solution.

### 3.2 Simulated Annealing

The simulated annealing is another optimization algorithm used for approximating the global optimum for the machine learning model. In details, it uses a metaheuristic way to analyze an optimization problem in a large searching space. This algorithm considers the status of current state and the next state and then moving forward. It is often used when the searching space is discrete, such as the travelling salesman problem in Python's mlrose package. The algorithm will repeatedly the iterative process until the model has reached the state that the global optima or the given computation budget is exhausted.

### 3.3 A Genetic Algorithm

A parallel genetic algorithm is the optimization algorithm that can avoid the problems inherent in many traditional approaches. It provides an effective way for searching globally for the best fitting model. Though it is hard to find precious value for the best parameters of optimization, it is well suited to search for the region of parameter space which contains the global minimum. This algorithm will start from a large amount of candidate solutions and then, with the iterations by eliminating non-adequate parts of the solutions, remaining the best fit parts of fitting optimum to make it a better continuously solution.

### 3.4 MIMIC

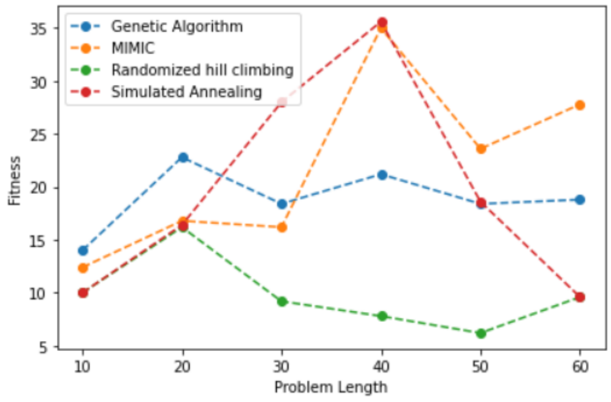
Mutual-Information-Maximizing In -put Clustering (MIMIC) is an optimization algorithm which is to find the best optimum by estimating the probability densities. The kernel of this algorithm is to communicate information for the cost function of the model and then operating from one iteration to the next iteration directly. The MIMIC factorizes the joint probability distribution in a chain-like model representing successive dependencies between variables.

## 4. Result

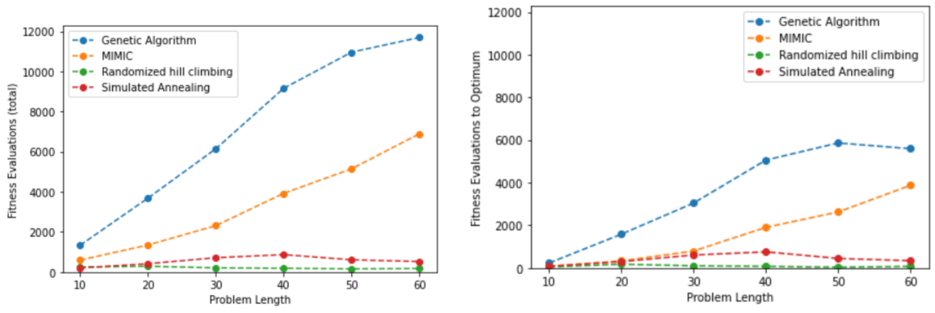
### 4.1 Four Peaks Problem

This four peaks problem is a problem which is to find the highest peak from a four peak. Some optimization algorithms can be appropriate to find the local maxima of this problem. We set the percentage into 0.4 which is an appropriate value to find the local minimum and let genetic algorithm just find the global optimum. We compare four algorithms in one figure from different aspects. Also, we adjusted the length of problem from 10 to 50 so that we can get the comparison of the changes of problem length.

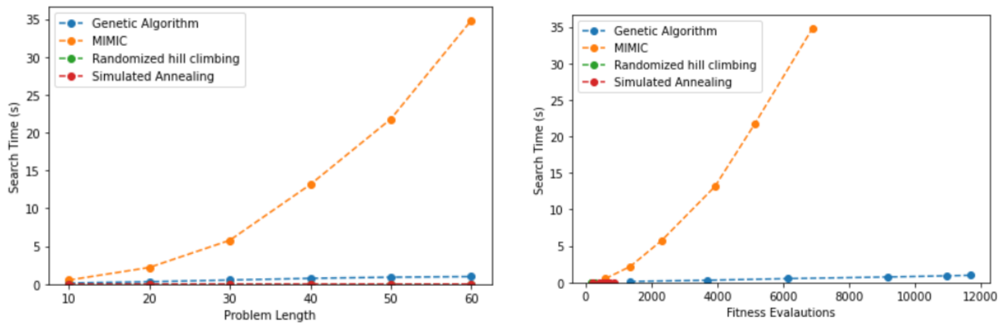
Firstly, compared with four algorithms in four peaks problem. From the figure below, we can conclude that for simulated annealing and MIMIC, when length = 40, it reaches the best fitness value. But Randomized hill climbing algorithm performed not as well as other three algorithms.



Besides, we also compare the fitness evaluations and fitness evaluation optimum to each algorithm, from the figure below, we can conclude that Genetic Algorithms have the best global optimum in the four optimization algorithms no matter the length of problem sets we used.



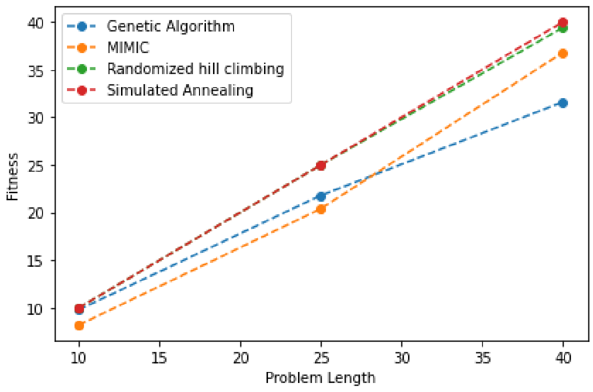
What's more, I also draw the relationship between searching time with problem length and fitness evaluation. From the figure, it is clear that MIMIC has the best searching time whatever the problem length is and whatever the fitness evaluations are.



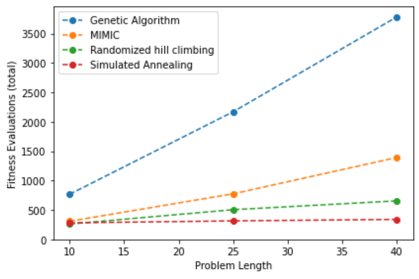
## 4.2 One Max Problem

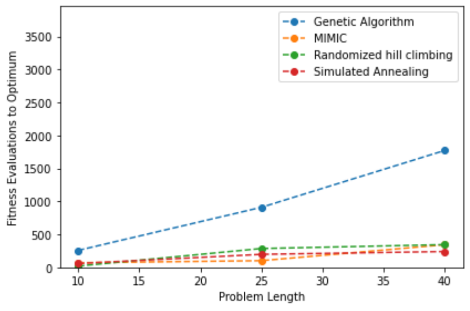
One Max problem is another popular optimization problem in mlrose package. This problem is to evaluate the fitness of an n-dimensional state vector x = [$x_0$, $x_1$, … $x_{n-1}$] as: $Fitness(x) = \sum_{i=0}^{n-1} x_i$. In this problem, in order to get better results, we use the problem sets from 10 to 40.

Firstly, compared with four algorithms in one max problem. From the figure below, we can conclude that for simulated annealing and Randomized hill climbing, when length = 40, it reaches the best fitness value. Last but not least, different from four peaks problem, with the increase of problem length, all of the four algorithms' fitness increases.

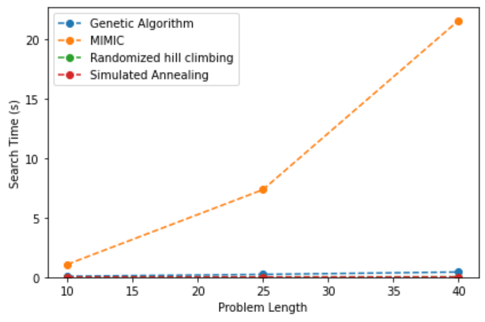


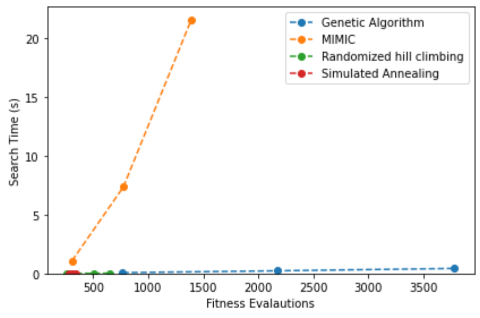
Besides, we also compare the fitness evaluations and fitness evaluation optimum to each algorithm, from the figure below, we can conclude that Genetic Algorithms have the best global optimum in the four optimization algorithms no matter the length of problem sets we used.





What's more, I also draw the relationship between searching time with problem length and fitness evaluation. From the figure, it is obvious that MIMIC has the best searching time whatever the problem length is and whatever the fitness evaluations are.





## 4.3 Knapsack Problem

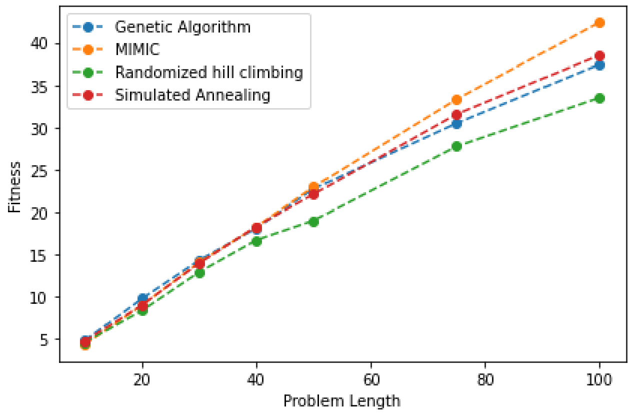
This Knapsack problem is a classical optimization problem in machine learning. Given a set of n items, where item I has known weight $w_i$ and known value $v_i$, and maximum knapsack capacity W. The Knapsack fitness function evaluates the fitness of a state vector x = $[x_0, x_1, \ldots x_{n-1}]$ as

$$Fitness(x) = \sum_{i=0}^{n-1} v_i x_i, \text{ if} \sum_{i=0}^{n-1} w_i x_i \leq W, \text{ and } 0, \text{ otherwise},$$

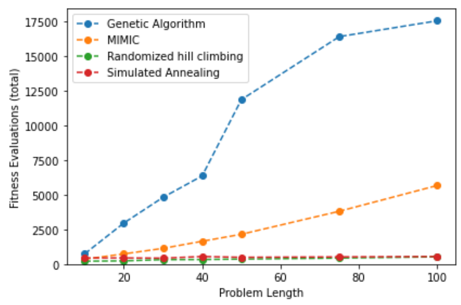
.

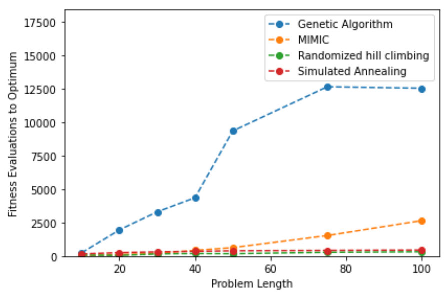
Each element is given with a mass and a value, we need to determine the number of each of the element in this vector to let the total weight no more than the threshold value and make the total weight as large as possible. The length of this problem sets is from 10 to 100.

Firstly, compared with four algorithms in four peaks problem. From the figure below, we can conclude that for simulated annealing and MIMIC, when length = 100, it reaches the best fitness value. But Randomized hill climbing algorithm performed not as well as other three algorithms.

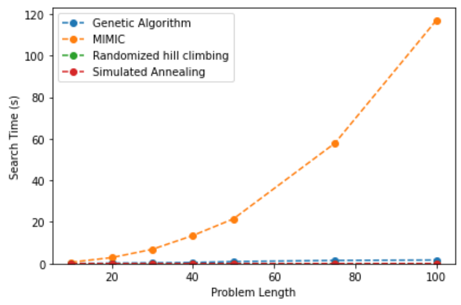


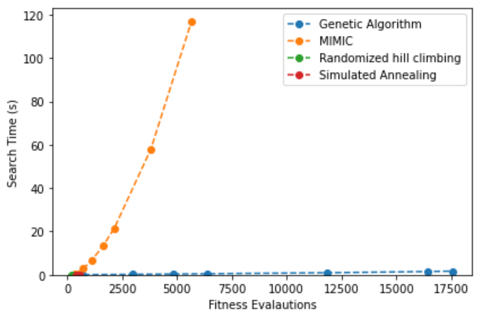
Besides, we also compare the fitness evaluations and fitness evaluation optimum to each algorithm, from the figure below, we can conclude that Genetic Algorithms have the best global optimum in the four optimization algorithms no matter the length of problem sets we used. MIMIC also has a good performance comparing to the other two algorithms.





What's more, I also draw the relationship between searching time with problem length and fitness evaluation. From the figure, it is clear that MIMIC has the best searching time whatever the problem length is and whatever the fitness evaluations are.





## 4.4 Find a good weight for a Neural Network

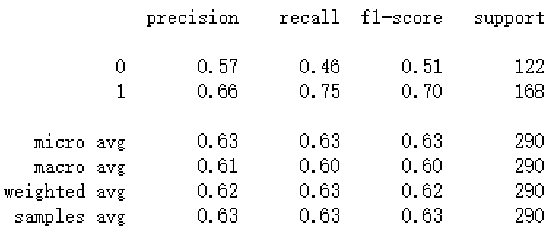
The last part of this project is to find a good weight for a Neural Network with optimization algorithms. The datasets I used the NBA games for 2019-2020 seasons. The datasets include all matches' information for each team. As we know, percentage of shooting, rebounds and assists might affect the result of each match. Therefore, I am trying to use Neural Networks to build a model to predict the result of a game. This dataset contains 965 rows of data. Binary number 1 or 0 represents the win or loss of each game for the home team.

To make a comparison of Neural Networks model, I use two activation function, one is the Relu activation function and another one is Sigmoid activation function. I use the timers to count the running time for training models and testing models to make a comparison with different optimization model. I also calculate the Jaccard index and classification report for the analyzing results. The analyzing result including the test accuracy scores and F1 scores. These parameters are calculated from sklearn.metrics packages in Python. Here are the outputs.

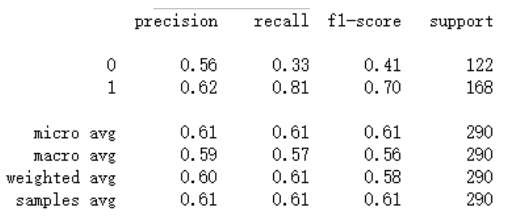
***Normal Neural Networks:***

| | Relu | Sigmoid |
|---|---|---|
| Training time | 0.13168369999999996 | 1.0322408999999997 |
| Testing time | 0.0007014000000005183 | 0.00107699999999995505 |
| Jaccard index | 0.457286432160804 | 0.457286432160804 |

| | | |
|---|---|---|
| Classification report | | |



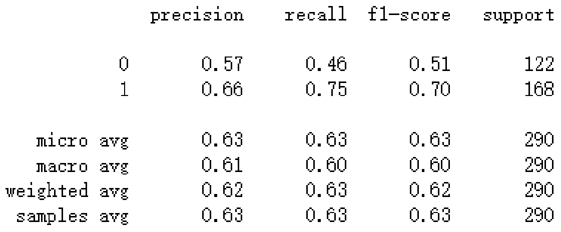
```
              precision    recall  f1-score   support

           0       0.57      0.46      0.51       122
           1       0.66      0.75      0.70       168

   micro avg       0.63      0.63      0.63       290
   macro avg       0.61      0.60      0.60       290
weighted avg       0.62      0.63      0.62       290
 samples avg       0.63      0.63      0.63       290
```



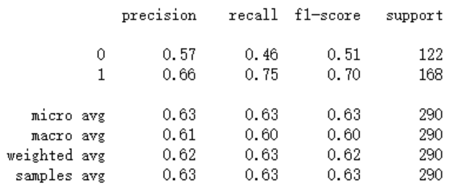
```
              precision    recall  f1-score   support

           0       0.56      0.33      0.41       122
           1       0.62      0.81      0.70       168

   micro avg       0.61      0.61      0.61       290
   macro avg       0.59      0.57      0.56       290
weighted avg       0.60      0.61      0.58       290
 samples avg       0.61      0.61      0.61       290
```

***Randomized Hill Climbing:***

| | Relu | Sigmoid |
|---|---|---|
| Training time | 0.23200109999999974 | 0.5258365000000005 |
| Testing time | 0.0009589999999999321 | 0.0006398999999994714 |
| Accuracy score | 0.6275862068965518 | 0.6275862068965518 |
| F1 Score | 0.6196865203761756 | 0.6196865203761756 |
| Jaccard index | 0.457286432160804 | 0.457286432160804 |
| Classification report | | |



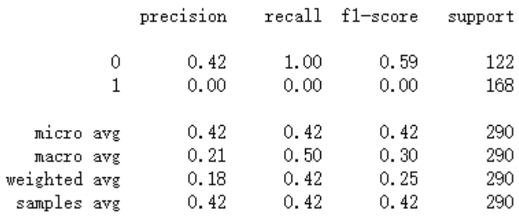
```
              precision    recall  f1-score   support

           0       0.57      0.46      0.51       122
           1       0.66      0.75      0.70       168

   micro avg       0.63      0.63      0.63       290
   macro avg       0.61      0.60      0.60       290
weighted avg       0.62      0.63      0.62       290
 samples avg       0.63      0.63      0.63       290
```



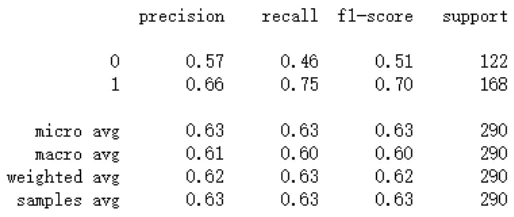
```
              precision    recall  f1-score   support

           0       0.57      0.46      0.51       122
           1       0.66      0.75      0.70       168

   micro avg       0.63      0.63      0.63       290
   macro avg       0.61      0.60      0.60       290
weighted avg       0.62      0.63      0.62       290
 samples avg       0.63      0.63      0.63       290
```

***Simulated Annealing***

| | Relu | Sigmoid |
|---|---|---|
| Training time | 2.3663499000000012 | 2.2413022 |
| Testing time | 0.0012954999999994499 | 0.0010329000000002253 |
| Accuracy score | 0.4206896551724138 | 0.6275862068965518 |
| F1 Score | 0.24914630063608975 | 0.6196865203761756 |
| Jaccard index | 0.2663755458515284 | 0.457286432160804 |
| Classification report | | |



```
              precision    recall  f1-score   support

           0       0.42      1.00      0.59       122
           1       0.00      0.00      0.00       168

   micro avg       0.42      0.42      0.42       290
   macro avg       0.21      0.50      0.30       290
weighted avg       0.18      0.42      0.25       290
 samples avg       0.42      0.42      0.42       290
```

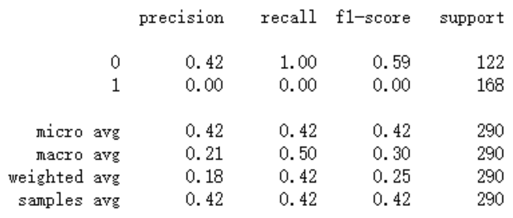


```
              precision    recall  f1-score   support

           0       0.57      0.46      0.51       122
           1       0.66      0.75      0.70       168

   micro avg       0.63      0.63      0.63       290
   macro avg       0.61      0.60      0.60       290
weighted avg       0.62      0.63      0.62       290
 samples avg       0.63      0.63      0.63       290
```

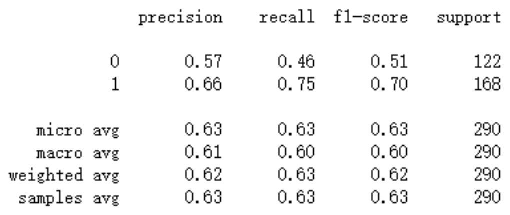
***A genetic Algorithm***

| | Relu | Sigmoid |
|---|---|---|
| Training time | 2.2715651 | 2.2230119000000013 |
| Testing time | 0.0007366000000015305 | 0.0007601000000008185 |
| Accuracy score | 0.4206896551724138 | 0.6275862068965518 |

| | | |
|---|---|---|
| F1 Score | 0.24914630063608975 | 0.6196865203761756 |
| Jaccard index | 0.2663755458515284 | 0.457286432160804 |
| Classification report | | |



```
              precision    recall  f1-score   support

           0       0.42      1.00      0.59       122
           1       0.00      0.00      0.00       168

   micro avg       0.42      0.42      0.42       290
   macro avg       0.21      0.50      0.30       290
weighted avg       0.18      0.42      0.25       290
 samples avg       0.42      0.42      0.42       290
```



```
              precision    recall  f1-score   support

           0       0.57      0.46      0.51       122
           1       0.66      0.75      0.70       168

   micro avg       0.63      0.63      0.63       290
   macro avg       0.61      0.60      0.60       290
weighted avg       0.62      0.63      0.62       290
 samples avg       0.63      0.63      0.63       290
```

Here are the results from the program and later on in the conclusion section, I will make a comparison and analysis from the table above.

## 5. Conclusion

### 5.1 Three optimization problems analysis

In conclusion, from the result we received from thee three optimization problems: Four peaks, one max and Knapsack, we can obtain the results from the following aspect:

***Fitness:***

With the comparison of three problems, I can conclude that different optimization problem has different performance if the length of problem sets is different. MIMC and Simulated Annealing have better performance than the others. But generally, they are almost close to each other. Randomized Hill Climbing is not as well as the other three algorithms.

***Fitness evaluations and fitness evaluation optimum:***

From this aspect, genetic algorithms is significantly better than the other three algorithms, which means that if we want to have a better fitness evaluation mean value, we should choose Genetic algorithms as our first choice no matter what the length of the problem sets.

***Searching time:***

With the comparison of three problem, we can obtain that MIMIC has the longest searching time. For the problem length and fitness evaluation, MIMIC both performed worse than the other three algorithms. Therefore, if we want to save searching time. MIMIC is not a good method to choose for optimization.

### 5.2 Optimization on a Neural Network analysis

By comparing the training and testing time of neural network between the normal approach and three optimization approach, we can conclude that for the Relu activation function of the neural network, the optimization methods need more training and testing time. Randomized Hill Climbing needs the most time for both training time and testing time. However, for the sigmoid Neural Network, the results are different. Randomized

Hill Climbing is better than the others. Simulated Annealing is the worst algorithms.

When comparing with three optimization algorithms on my datasets of NBA games, Randomized Hill Climbing has the best accuracy score and F1 score on Relu activation function of Neural Networks. But on sigmoid Neural Networks, these three optimizations have close accuracy score and F1 score.

Therefore, in summary, for my choose, I will use Randomized Hill Climbing as the optimization between this provides the better accuracy score and F1 score, which means that the predicting model is better, and it can provide more accurate results and predictions. The reason might be that the dataset I used is small and with strong correlation to the match result. The output is binary which is neither 1 or 0. The datasets is not complex so that Randomized Hill Climbing can get better optimum weights to this problem. From the figure of the comparison of four algorithms, we can conclude that the more the length of problem we set, the more cost we will need for optimization.

## 5.3 Summary

From this assignment, I learned that we need to choose the best optimization algorithms for each of the problem. By evaluating the performance of each optimization algorithm, we can find the best approach to get the optimal neural network weights. From the course materials, I learn the theoretical knowledge about these four algorithms and by experiencing the algorithms on my own project, I have a deep understanding on machine learning. Besides, by using the optimization algorithms on my previous neural networks, the classification accuracy and F1 scores are significantly improved. With the comparison of three algorithms, I learned that how I can find the optimal neural networks weight to improve the performance of training and prediction.