# CS 7641 Randomized Optimization

Jun Wang

jwang3316@gatech.edu

***Abstract*—In this submission, I implemented four different search techniques, Random Hill Climbing, Simulated Annealing, Genetic Algorithm and MIMIC, on three different discrete optimization problems and highlighted the advantages of them. Then, I tried to find the good weights for a neural network, using them on one problem that I created in Assignment 1.**

## 1 PROBLEMS

### 1.1 Travelling Salesperson Problem

This is a problem of finding the shortest path to visit the next city and return to the original city, given a list of cities and the distance between each pair of cities. This is a typical optimization problem. It is an NP-hard problem in combinatorial optimization[1]. By changing the value of the maximize parameter in the *mlrose-hiive* package, we can convert the minimum problem to a maximum problem to meet the requirements of the assignment.

### 1.2 Flip Flop Problem

FFP is a problem that counts the number of bit-alternation times in a binary bit string. For example, change from 0 to 1 or from 1 to 0 in the next digit is counted as 1, which is a problem that has many local maxima. A maximum fitness bit string would be one that consists entirely of alternating digits[2], such as 010101. This is also a problem that can generated by *mlrose-hiive* package.

### 1.3 Continuous Peaks

This is a classic optimization problem, transformed from the 4-peak problem. In this problem, there are many local optima in a continuous array, consisting of

[1] TSP explained in Wikipedia: https://en.wikipedia.org/wiki/Travelling_salesman_problem

[2] FFP problem introduction

numbers from 0 to 1. It could highlight  advantages of different algorithms. This problem can be generated by the *mlrose- hiive* package.

## 2 TRAVELLING SALESPERSON PROBLEM

This experiment sets up 50 cities and initializes the distance of each pair of cities at random (the cost in both directions is equal). At the same time, this experiment is a maximization problem by setting *maxmize*=True.

### 2.1 Performance of the algorithms

***Random Hill Climbing*** - In this part, we focus on the influence that different numbers of restart  have on the fitness of the algorithm, because the restart count is very important to the random hill climbing search. The number of restarts is set from 0 to 250, to find the optimization. The fitness curve growing by iterations is different as shown in Figure 1.

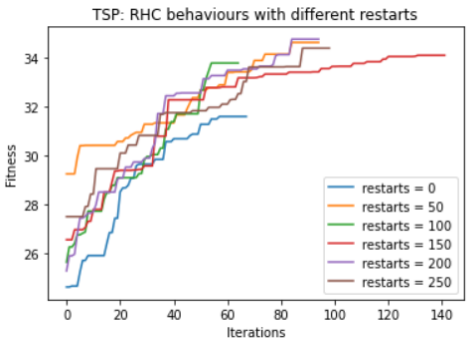


***Figure 1***—Performance of the RHC algorithm on TSP.

As we can see, the fitness grows with the iterations. The higher the restarts, the higher the fitness. But when restarts are bigger than around 50, the fitness reaches 35.0 and just keep the level, which results from the randomization times is enough and will not impact the fitness level too much.

***Simulated Annealing*** - The SA's formula is $T(t)=max(T0-rt,Tmin)$. The parameters that we focus on in SA is the temperature t and the temperature recession rt. We change their values to find the optimal search solution. The performance is shown in Figure 2.

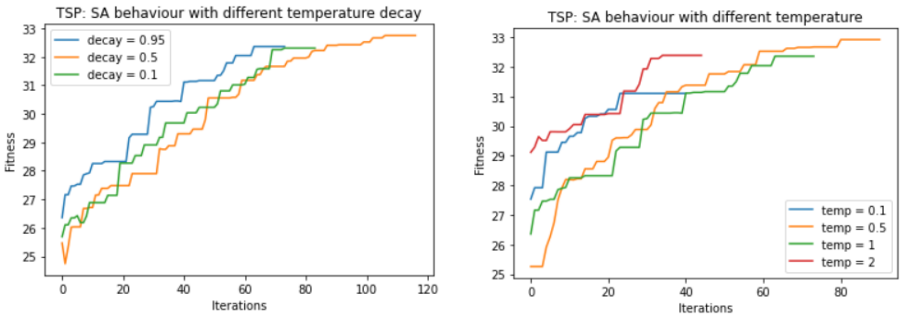


***Figure 2***—Performance of the SA algorithm on TSP. Left is temperature recession, on the right is temperature.

As we can see, when $rt$ = 0.5, the fitness reaches the optimization after around 80 iterations. When $t$ = 0.5, the fitness reaches the optimization after around 75 iterations.

***Genetic Algorithm*** - For GA algorithm, we focused on the population size and mutation probability, which will impact on the fitness. The result is shown in Figure 3.

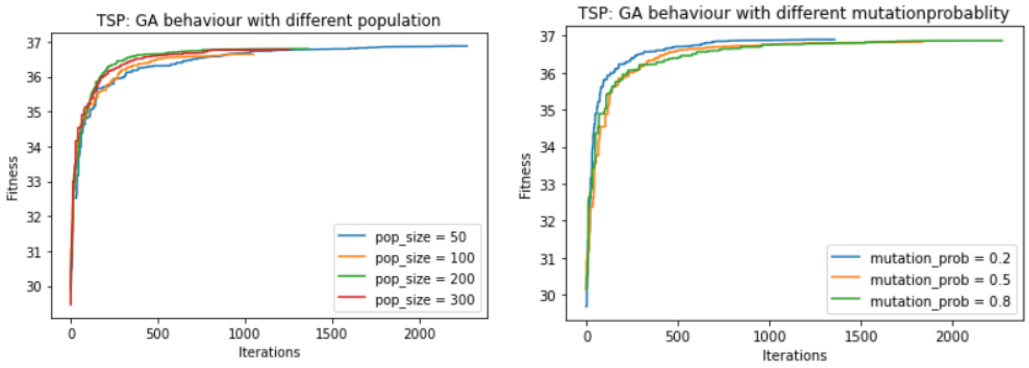


***Figure 3***—Performance of the GA algorithm on TSP. Left is population, on the right is variation rate.

As we can see, the larger population size and lower probability of mutation increase the fitness faster in the first 1000 iterations. But after 1000 iterations, the fitness with different pop_size or different mutation probabilities gradually reach the same value. Therefore, the optimal result will be generated when the values of population size and mutation probability are reasonable.

***MIMIC*** - For MIMIC, I tried to find the optimal results by changing the mimic. As shown in Figure 4.

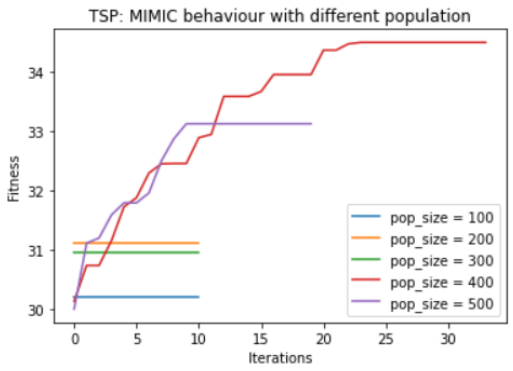


***Figure 4***—Performance of MIMIC on TSP, different population size.

As we can see, fitness performs a similar trend as the population size changes.

## 2.2 Conclusion

Comparing the performances of the four different algorithms on the Travelling salesperson problem, Genetic Algorithm performs better than other 3 search techniques. As shown in Figure 5.

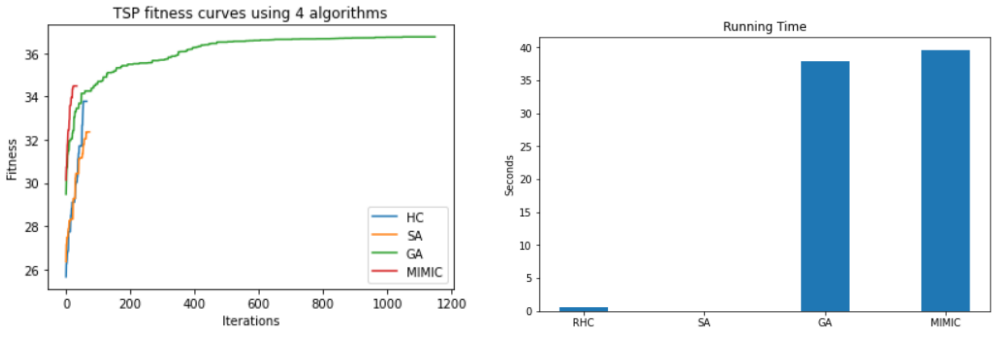


***Figure 5***—Performance of the 4 algorithms on the TSP.

As we can see, from the left graph, the Genetic Algorithm got the best fitness after around 200 iterations. I think the reason is that GA's chromosome could solve the TSP by generating more appropriate states, which help GA's variants to

search the hypothesis space in a more efficient way. In addition, these different states are able to be blended in an appropriate way by the GA's crossovers.

## 3 FLIP FLOP

In this experiment, the size of the problem (length of the bit string) is 100.

### 3.1 Performance of the algorithms

***Randomized Hill Climbing*** - As the same as the condition in TSP, we focused on the number of restarts. As shown in Figure 6.

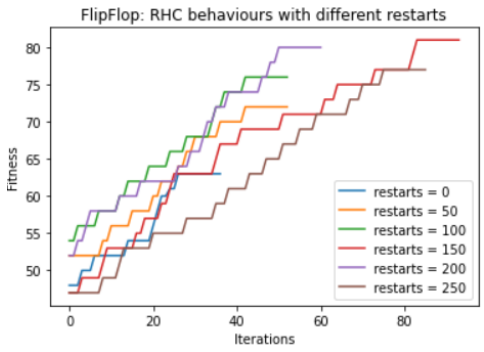


***Figure 6***—Performance of the RHC algorithm on the FFP with different restarts.

As we can see, the fitness achieves optimal value when the restarts is 150. After that, the fitness stays the level even after the restarts increase.

***Simulated Annealing*** - As shown in Figure 7, as the same as the condition of the TSP, the temperature and the temperature recession have a large influence on the performance of the SA after iterations.

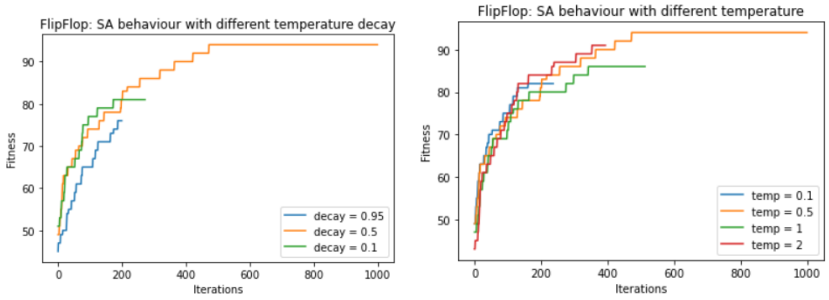


***Figure 7***—Performance of the SA on the FFP with different t and rt.

***Genetic Algorithm*** - As shown in Figure 8, the fitness increases as the population becomes bigger.

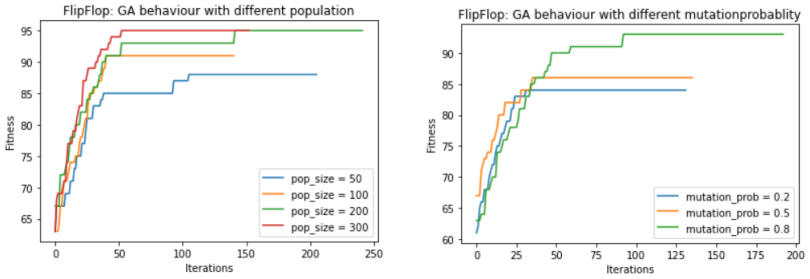


***Figure 8***—Performance of the GA algorithm on FFP.

***MIMIC*** - For MIMIC, as shown in Figure 9, the trend of the fitness is similar to that of GA generates. The bigger the population, the better the fitness.

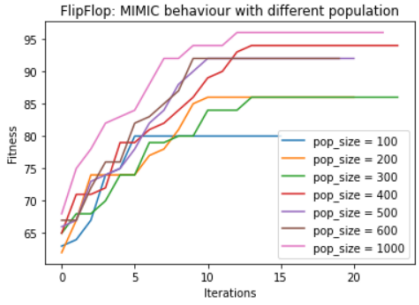


***Figure 9***—Performance of MIMIC on FFP, different population size.

### 3.2 Conclusion

Comparing the performances of the four different algorithms on the Flip Flop problem, Simulated Annealing performs better than other 3 search techniques. As shown in Figure 10.

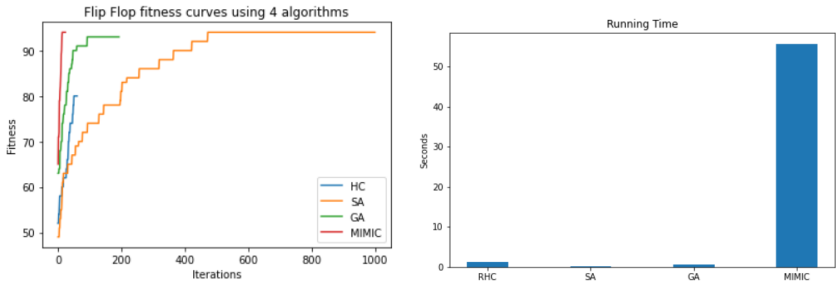


***Figure 10***—Performance of the 4 algorithms on the Flip Flop.

As we can see, from the left graph, the Simulated Annealing got the best fitness after around 500 iterations. I think the reason is that SA could reach the global optimal solution if it is in the ideal case. For SA, the fitness curve rises fast and fluctuates more frequently at the beginning, which it explores more random points, as the temperature is high at the same time. Then, as the temperature drops, it tends to refuse to accept the bad points, which leads the optimal solution to get closer to the global optimal solution, while the fluctuation decreases and the curve gradually stays. This process might need more iterations to get the optimal solution, so does the experiment, in which the SA finished the optimization after 1000 iterations. In addition, the runtime of the SA is far less than other 3 algorithms, which is because the SA is not a global random search, which doesn't need to generate populations, saving lots of time.

## 4 CONTINUOUS PEAKS

Compared to the 4-peak problem, the continuous peaks problem only aims to find the longest sequence of 0 and the longest sequence of 1, no matter where their positions are. In this experiment, the problem size (local optima) is 150.

First of all, because RHC and SA perform badly when the *max_attempts* = 10 and the *max_iters* = 1000 (*min_temp* = 0.01 for SA ). In order to get better results and comparisons, I change the *max_attempts* to 100 and the *max_iters* = 5000, generating more states and going through more iterations.

### 4.1 Performance of the algorithms

***Randomized Hill Climbing*** - For RHC, as shown in Figure 11, it indicates that the best fitness reaches 181.0 when restarts is 250.

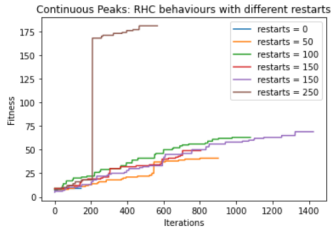


***Figure 11***—Performance of the RHC on the CPP.

***Simulated Annealing*** - For SA, the best fitness is 269.0 when temperature is 4.

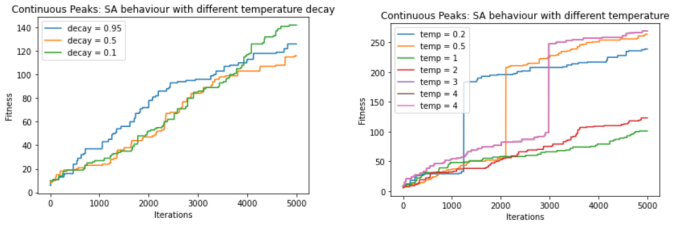


***Figure 12***—Performance of the SA on the CPP.

***Genetic Algorithm*** - As shown in Figure 13, it indicates that the best fitness is 244.0 when mutation probability is 0.2. GA performs well, due to the Continuous Peaks problem having too many local optimas, which might highlight the advantage of GA that searches the optimal solution randomly.

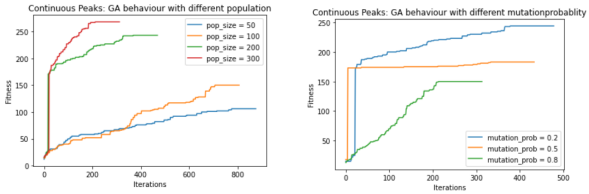


***Figure 13***—Performance of the GA on the CPP.

***MIMIC*** - For MIMIC, because it runs a very long time on my local machine for computing and plotting when the *max_attempts* to 100 and the *max_iters* = 5000. So I run MIMIC in the original condition. And the best fitness is already 248.0 in

only 40 iterations, which is close to that of others in over 1000 iterations. Therefore, it is obvious that MIMIC performs better than others.

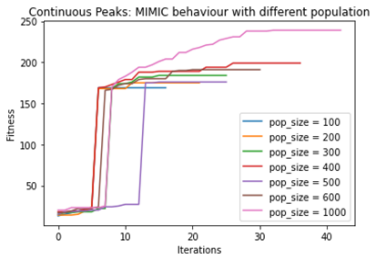


***Figure 14***—Performance of the MIMIC on the CPP.

### 4.2 Conclusion

Because the RHC and SA had bad performance at first, I thought it was caused by the parameter or the lack of iterations. But MIMIC still performed well with the original parameters. Therefore, I think that MIMIC is better than others in this optimization problem. I redo the test with the original parameters and compare their performance, as shown in Figure 15. MIMIC is able to converge quickly and reach best fitness in less iterations. I think the reason is that this problem highlights the complete random optimization, while the SA and RHC easily stay in local optimas during the climb-up process. In addition, though MIMIC has far more runtime than others, it has a better starting situation than others because of its distribution estimation.

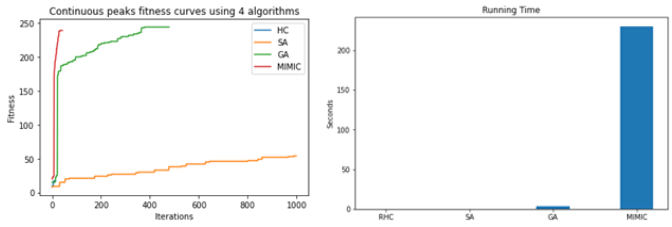


***Figure 15***—Comparison of the algorithms on the CPP.

## 5 REPLACE NEURAL NETWORK IN ASSIGNMENT 1

In this section, I used random optimization method instead of backprop for the neural network on the problem of breast cancer in Assignment 1[3]. The purpose of this data set is to judge whether a patient has breast cancer by learning the characteristics of the sample. Then the parameters are updated by those three algorithms, Randomized Hill Climbing, Simulated Annealing and Genetic Algorithm and gradient descent algorithms and finally compare them.

***RHC*** - From Appendices, we can see that RHC is not effective to optimize the neural network.

***SA*** - From Appendices, the loss curve is smooth with less fluctuation, and show better generalization on the validation set, which indicates Simulated Annealing is effective in optimizing the weights of the neural network.

***GA*** - From Appendices, we can see that Genetic Algorithm is not suitable for neural network optimization. The loss curve is not smooth. This results from the principles of the Genetic Algorithm.

***Gradient Descent*** - From Appendices, we can see why GD is often used for parameter optimization. Its loss curve is very smooth and has high accuracy.

***Comparison -*** As we can see from the following figures, Simulated Annealing performs best among the optimization algorithms on optimizing the neural network for Breast Cancer classification problem. However, its accuracy and loss are not as good as the gradient descent method. In short, Gradient Descent is better than optimization algorithms in parameters update of neural networks, due to its traceability. Additionally, as for the runtime of the optimization algorithms, the Genetic Algorithm has far more running time than the other two, running 6648.73s, while SA only runs 68.34s and RHC runs 60.48s.

[3] Breast Cancer dataset from UCI machine learning repository.

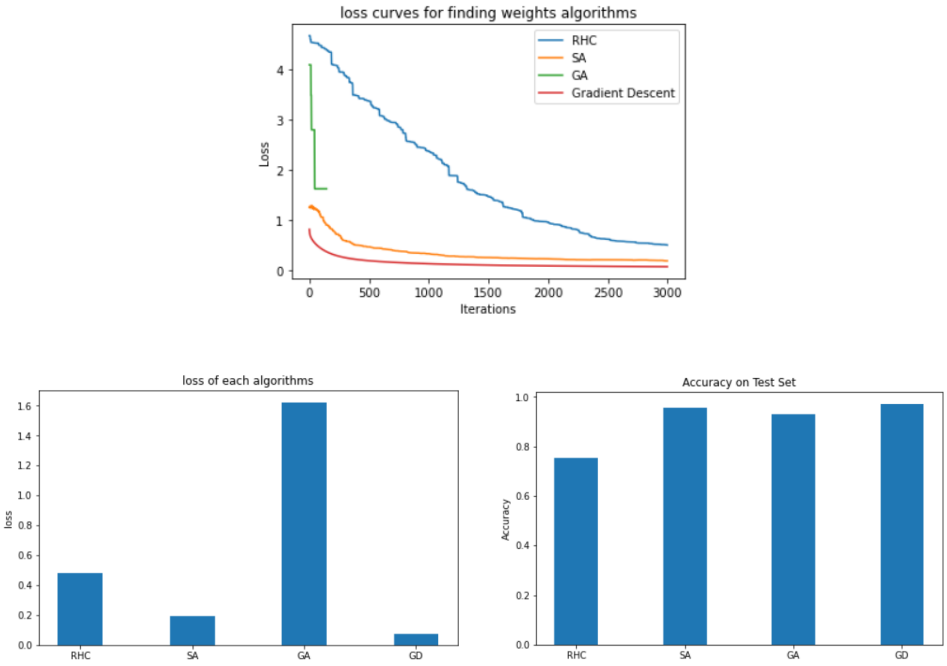


***Figure 20***—Comparison of different methods on optimizing parameters of the neural network.

## 6 REFERENCES


1. *Travelling salesperson problem*. Wikipedia, https://en.wikipedia.org/wiki/Travelling_salesman_problem
2. Nini (2019), Introduction, Implementation and Comparison of Four Randomized Optimization Algorithms.

   https://medium.com/@duoduoyunnini/introduction-implementation-and-comparison-of-four-randomized-optimization-algorithms-fc4d96f9feea

3. Hayes, G., Rollings, A. (2020). mlrose: Machine Learning, Randomized Optimization and SEarch package for Python
   https://github.com/gkhayes/mlrose
   https://github.com/hiive/mlrose.

4. De Bonet, J., C. Isbell, and P. Viola (1997). MIMIC: Finding Optima by Estimating Probability Densities. In Advances in Neural Information Processing Systems (NIPS) 9, pp. 424–430.
1. William H. Wolberg,W. Nick Street,Olvi L. Mangasarian, "Breast Cancer Wisconsin (Diagnostic) Data Set", UCI Machine Learning Repository.https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

## 7 APPENDICES

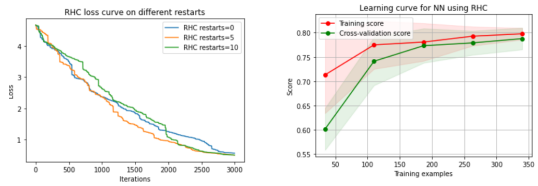


***Figure 16***—Performance of the RHC on the optimizing NN.

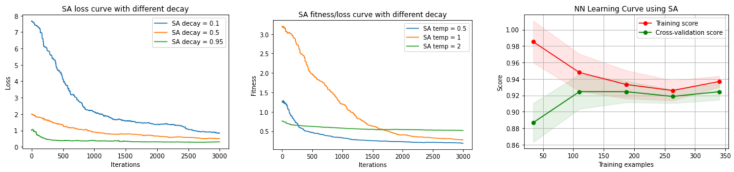


***Figure 17***—Performance of the SA on the optimizing NN.

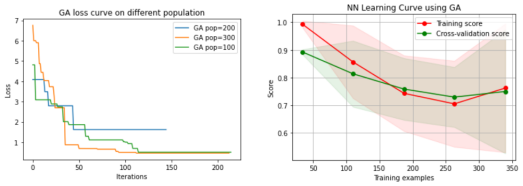


***Figure 18***—Performance of the GA on the optimizing NN.

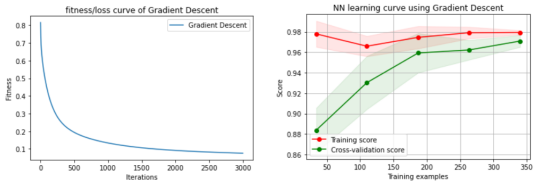


***Figure 19***—Performance of the GD on the optimizing NN