Randomized Optimization Model Comparison

Zhilan Deng

Abstract- This paper has two parts in general, the first part is to use randomized optimization algorithms on three optimization problem domains and the second part is to use the same randomized optimization algorithms to find the optimal weights for the neural network algorithm that has been applied in homework1.

PART 1

I. INTRODUCTION

OPTIMIZATION PROBLEMS

The three optimization problem domains this paper selected are: Travelling Salesperson Problems (TSP), Flip Flop Problems (FFP) and Four Peaks Problems (FPP). This paper selects these three kinds of problems because they have very different attributes and would be able to show the advantages of different randomized optimization algorithms. Also, those problems are relatively simple and would be easier to see if the algorithms stuck in local maxima or successfully find the global maxima. Besides, these problems cover a large range of applications in real projects, and it would be beneficial to find out which algorithm is the most suitable for each one of these problems.

TSP is a problem which is looking for the shortest path starting and ends in the same city/node. One way to find the shortest path is to brute forth and look at every path before selecting the shortest one, however, when the number of cities/nodes increasing, the number of paths will be increasing exponentially, so it would be harder or even impossible to find the shortest path by brute forcing. In this situation, a randomized optimization method would be considered a good solution to TSP. In this paper, we initialized 50 cities randomly and run TSP on these 50 cities.

FFP looks at the alternative pairs in the array and will count the number of elements where x_i not equal to x_{i+1}. Similar to TSP, when the number of bits is small, brute force would be an easy solution to this problem, but when the number of bits raises, it would be impossible to brute force every situation and find the maximum number of alternations. In this paper, we input the array of 100 bits.

FPP looks at maximum of the number of 1s at the beginning and the number of 0s at the end. This problem has some interesting attributes in testing randomized optimization algorithm that it contains two local maxima except the global maxima, thus running different algorithms on this problem would be able to tell the ability of an algorithm to escape from local. In this paper, we initialize 20 bits array for FPP and the local maxima will be 20, which will be a test of algorithms performance later.

II. RANDOMIZED OPTIMIZATION ALGORITHMS

OVERALL PERFORMANCES IN DIFFERENT ALGORITHMS

The overall performances in the four algorithms is shown in figure 1. All algorithms are running with the best hyper parameters.

For TSP, the algorithm which returns the best result is genetic algorithm and it is much better than other algorithms. For FFP, the algorithms which has the best fitness value is MIMIC and randomized hill climbing (RHC), genetic algorithm (GA) performs the worst in FFP. In FPP where there are local maxima, RHC performs the worst as expected since this algorithm tend to be stuck in local maxima, and SA returns the best fitness value among other algorithms, and SA, GA and MIMIC perform similarly well in FPP.

The running time of each algorithm is shown in figure 2. MIMIC has the substantial longer running time compared to any other algorithms and GA also has longer running time compared to SA and RHC.

To conclude, for the problems chosen in this paper, considering the performance and the running time of each algorithm, GA would be a good selection for TSP and SA for FFP and FPP.

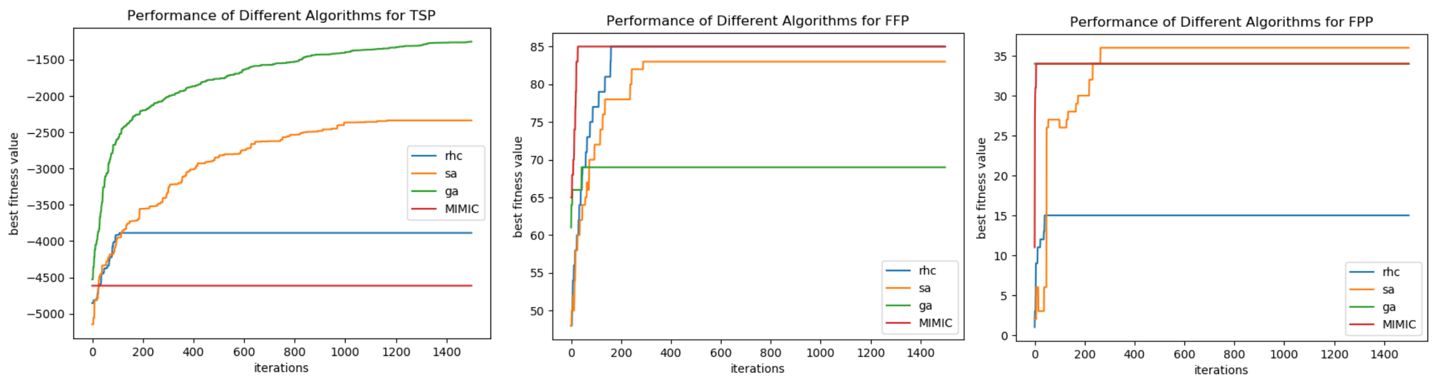


Figure 1. the overall performance of four algorithms in three different optimization problems.

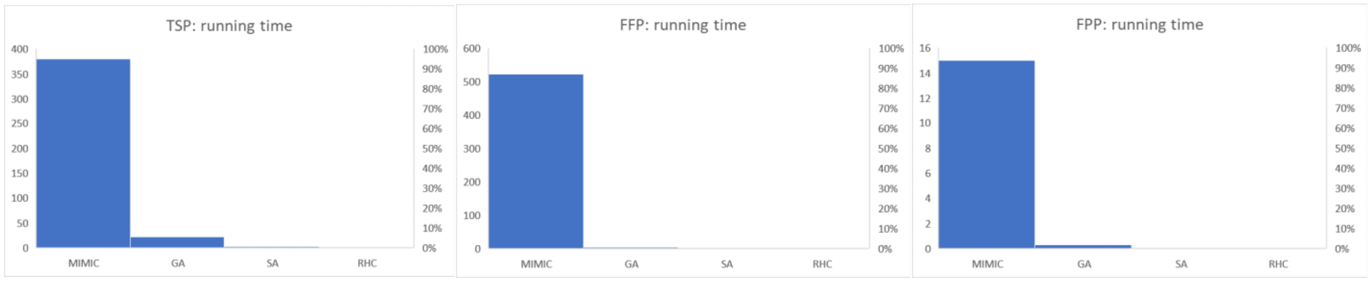


Figure 2. the running time of different algorithms in different problems.

RESULT ANALYSIS & HYPERPARAMETER TUNING

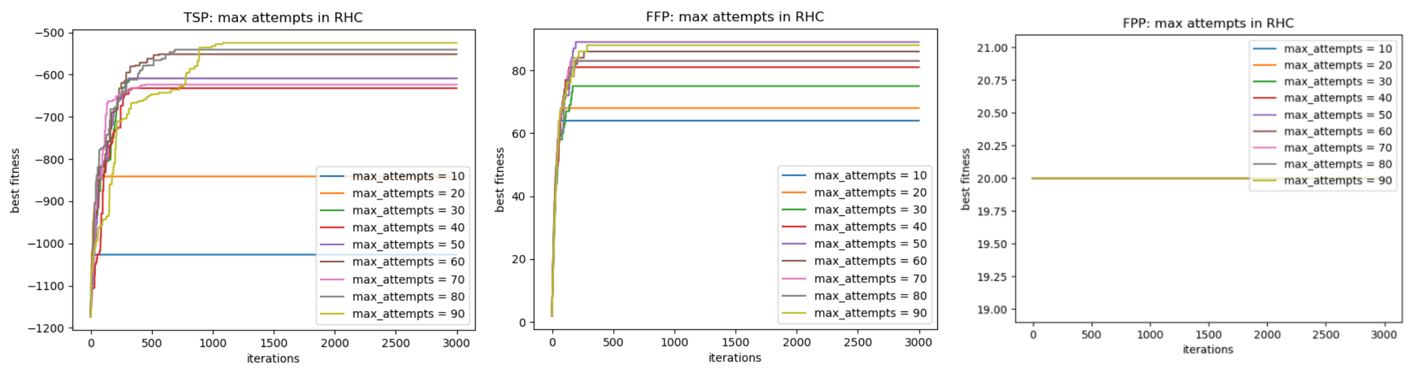
RANDOMIZED HILL CLIMBING (RHC)

Randomized hill climbing is a method of local search and RHC does not allow decreasing trend. It searches the neighbors and compare the value of neighbors with current value, if the value of neighbor larger than current value, then RHC moves to that neighbor, otherwise RHC will stop searching and return current value. The hyperparameter tuned in RHC is the number of maximum attempts and the number of random restarts. The results are shown in figure 3.

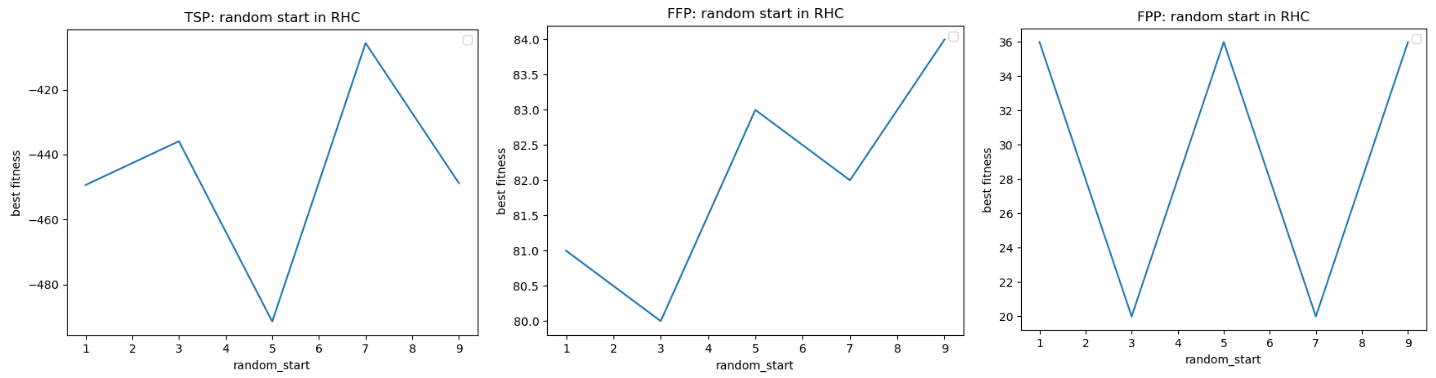
Maximum attempt is the maximum number of attempts to find a better neighbor at each step[1], if the max attempt set to 10, then the algorithm will just take 10 steps in the efforts of finding a local maxima, and return the maximum value in the 10 steps if it cannot find a local maxima. The influence of the number of maximum attempts in three different problems is shown in figure 3(a). The three problems show different reactions to increasing the number of maximum attempts. In TSP and FFP, with some shifts and jumps, the general trend shows that by increasing the maximum attempts, the best fitness value also increases. However, for FPP the best fitness value stays unchanged in 20 which is the local maxima. This tuning of the number of max attempts shows that generally when increasing the number of max attempts, RHC will find a better solution since it is allowed to take more steps to find the local maxima, however, it is also difficult for RHC to find the global optima if the problem has some local optima.

The running time of increasing the number of max attempts is shown in figure 3(c). The running time generally increases when the number of max attempts increases.

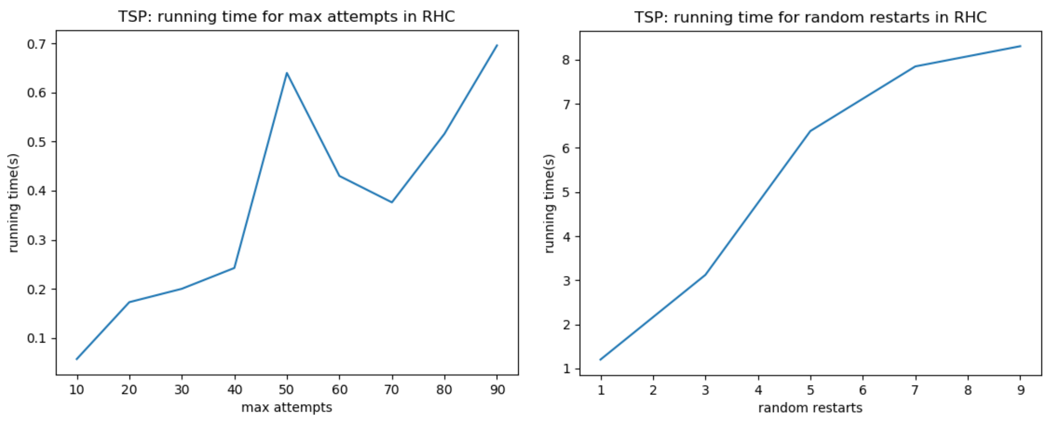
Number of random restarts is the number of allowed restarts that the algorithm can take from some random places. If the number of random restarts is 2, then the algorithm can pick two random places to restart with. Large number of random restarts can help in finding the global maxima, but the running time will also be longer. The best fitness values when increasing the number of random restarts is shown in figure 3(b) and the running time is shown in figure 3(c). Increasing the number of restarts increases the performance of all three problems, for TSP, it returns a better result compared to original method with no random restart, for FFP, it increases the best fitness value found by RHC when the number of random restarts increases, for FPP, it manages to escape from the local maxima and finds the global maxima. According to the performances, introducing random restarts would be helpful in finding best fitness values, however, increasing the number of random restarts would also increase the running time, so a tradeoff between the best fitness value and the running time need to be considered.



(a)



(b)



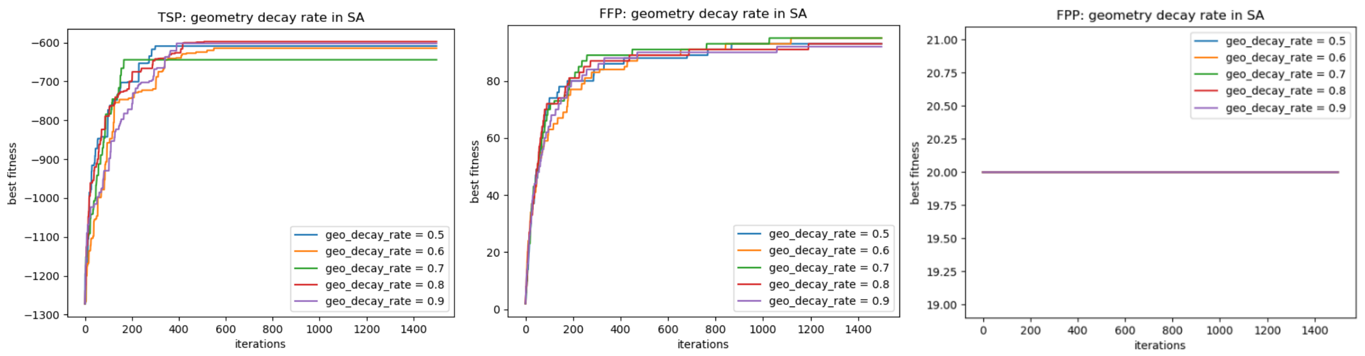
(c)

Figure 3. best fitness found by RHC in (a) different number of max attempts and (b)) different number of random restarts. (c) running time of the two hyperparameters.

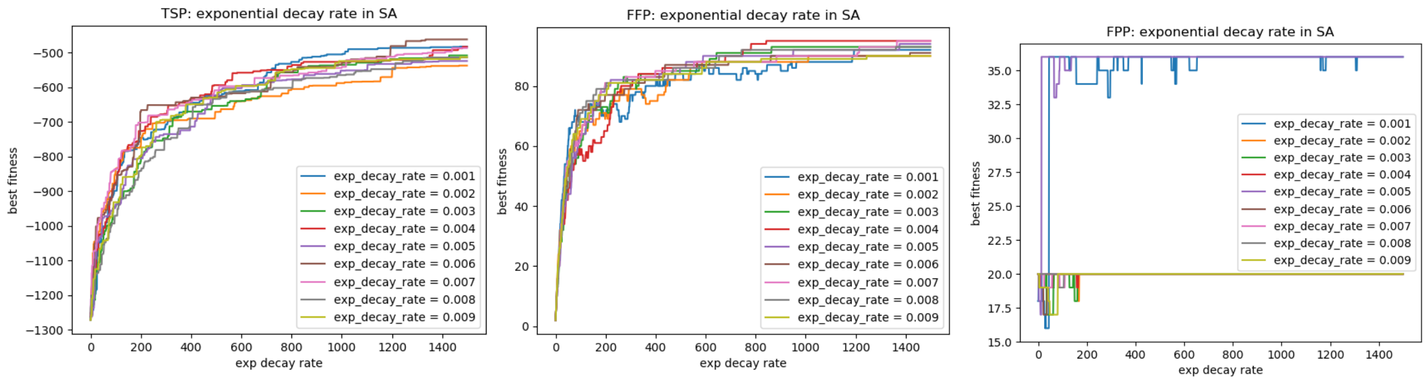
Simulated Annealing (SA)

Simulated annealing is similar to RHC but introduces exploring and allows decreasing trend. It introduces a parameter: temperature T. Similarly, it also looks at the value of neighbors and compare the value of neighbor with current value. The difference of SA compared to RHC is that if the value of neighbor is smaller than current value, it will have some possibility relate to the value of temperature T that SA will continue running and searching. Higher T values will have higher possibilities that SA accepts a decreased value and continues exploring. And T value will decrease when SA runs.

The hyperparameter tuned in SA are the maximum T value and the decay rate of T in geometry decay and exponential decay. The results are shown in figure 4 and figure 5.



(a)



(b)

figure 4. The results in (a) different geometry decay rate and (b) different exponential decay rate.

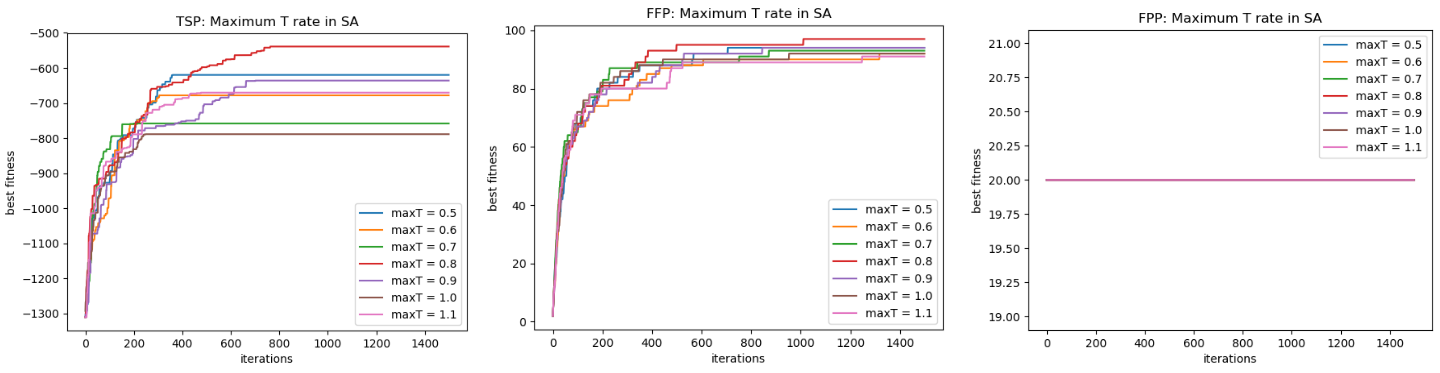


figure 5. the best fitness value of different T rates.

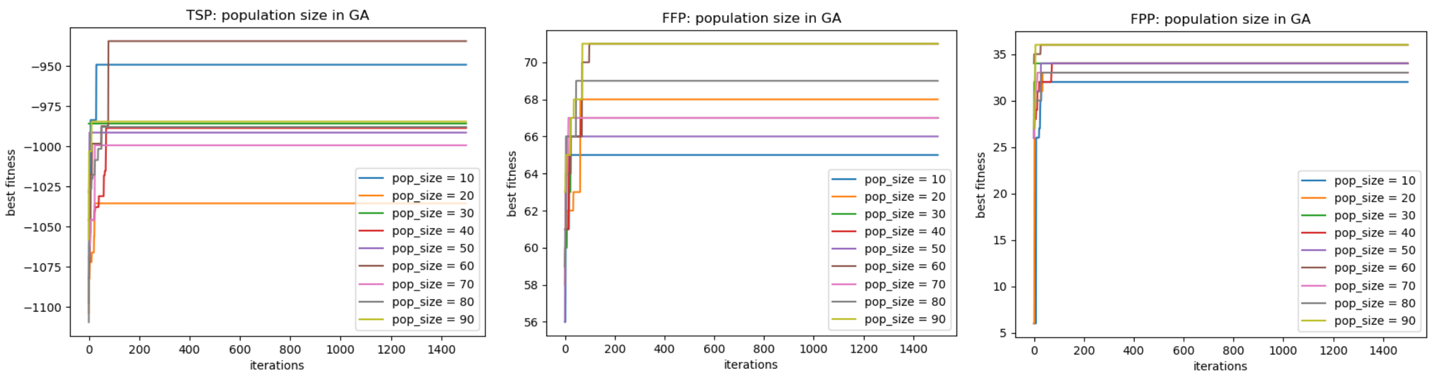
Figure 4 shows that in the range of the decay rates this paper sets, the performance is similar when changing the decay rates, one thing to be noted that although using geometry decay cannot escape from the local maximal for FPP, using exponential decay can find the global maxima, so in the final result, this paper uses exponential decay in SA. This paper also uses exponential decay to experiment the influence of the max T rate in SA and the result is shown in figure 5.

The results in figure 5 shows that, large or small T performs not as good as an appropriate T value. In both TSP and FFP, the best T rate is 0.8. The value of T influences the possibility that SA accepts decreasing trends and continue exploring, a higher T might introduce some less optimal situations, but a smaller T may also lead to overfitting and be stuck in local maxima, so choosing an appropriate T value would be important for SA.

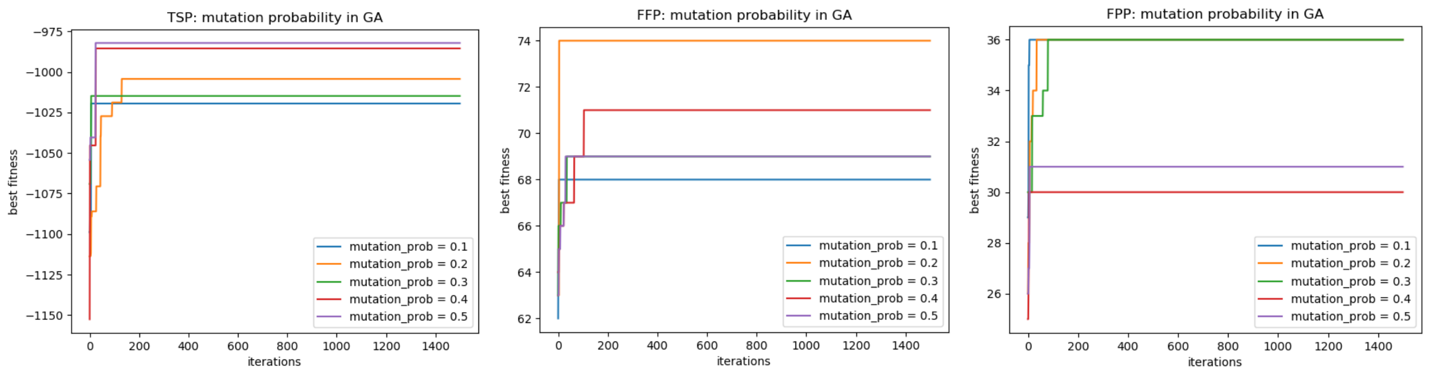
Genetic Algorithm (GA)

Genetic algorithm is inspired by the process of natural selection and it combines two solutions by mutation and crossover and put the new solution in the new population. The hyperparameter tuned in GA is the population size and mutation probabilities, the results are shown in figure 6.

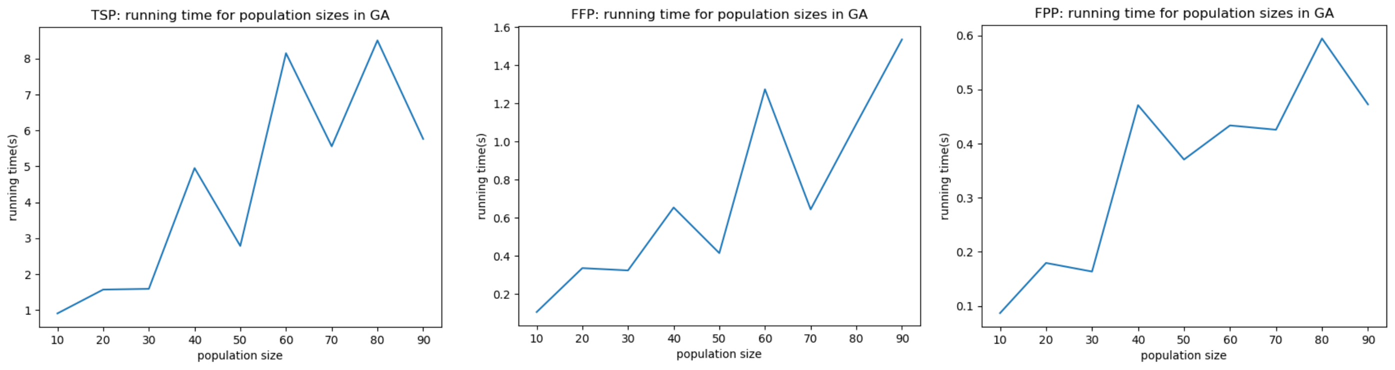
Population size is the population which contains possible solutions, if the population size too small, it may not include any good solutions and return a relatively low value, but if the population size too large, it will take more time to search for good solutions, and the increasing running time shows in figure 6(c).



(a)



(b)



(c)

Figure 6. (a) best fitness values in different population size; (b) best fitness values in different mutation probability; (c) running time of different population sizes.

Figure 6(a) shows that for FFP and FPP, the larger the population size is the better the result is. But there is no obvious trend in TSP when the population size increases.

Mutation probability describes the probability that the algorithm will break the current local optimal solution and find a potential better solution, it has the similar function to exploring as the temperature T in SA. The higher the Mutation probability is, the more likely GA will break the local optimal solution. The results of best fitness values found by different mutation probability is shown in figure 6(b). Interestingly, TSP, again, shows a different reaction to mutation probabilities compared to FFP and FPP. For TSP, the higher the mutation probability is, the better the result is. But for FFP and FPP, the best mutation probability is relatively small and large mutation probabilities fail to return good solutions.

MIMIC

MIMIC works by first uniformly sampling a group of candidates, evaluating the fitness and keep some percent of the candidates which have higher fitness values and sample more from this domain. The hyperparameter tuned in MIMIC is the percent of samples keep per round and the result is shown in figure 7.

The result shows that either too large or too small percent keep per round will return the best result, the best percent keep per round in TSP is 0.9， in FFP is 0.6 and in FPP is 0.4. If keep too much percent of candidates per round, the algorithm will learn very slow, but if keep a too small percent of candidates per round, it will be high probability that the optimal solution being removed in the candidates, so a propriate percent to keep per round need to be tested and selected carefully.

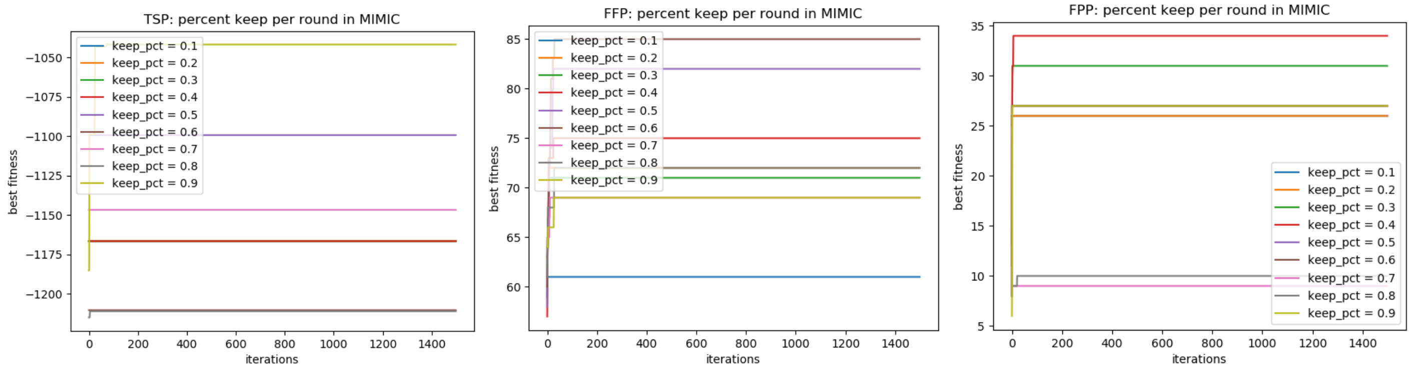


Figure 7. The best fitness values in different percent keep in each round.

PART 2

I. Introduction

We will use randomized optimization algorithms (RHC, SA, GA) to find the best weight in neutral network which we have implemented in homework1. Instead of the parameter tuned in back propagation, such as the hidden units, learning rates and models, we will frozen those parameter used in back propagation and tune the parameter of randomized optimization algorithm to find the best performance.

To brief the problem in homework1, this paper tried to classify if the income of a person higher than 50K per year or lower than 50K per year using the demographic attributes of that person including age, educational levels, occupation, race and sex. The dataset stores the demographic attribution of a person and a label of 0 if the income of this person is less than 50K and 1 if the income of this person is higher than 50K.

II. Results of Different Randomized Optimization Algorithms

Backpropagation Results

The results of original neural network using back propagation is shown in figure 8. This will used as a comparison to the results using randomized optimization algorithm.

Randomized Hill Climbing (RHC)

In RHC, the best restart number is 2, the best max iteration is 500 and the best max attempts is 50. The result of RHC is shown in figure 9. The result shows that although the score of training set and testing set of RHC are slightly lower than that of backpropagation, the difference between training set and testing set are smaller than the back propagation method, thus the overfitting is less in RHC.

Simulated Annealing (SA)

In Simulated Annealing, the best max iterations is 1000, the best max attempts is 10 and the best decay rate is Geometry decay, the result of SA is shown in figure 10. Similar to RHC, the score of SA is slightly lower than the score of back propagation, but the overfitting is less than the back propagation.

Genetic Algorithm (GA)

GA falls in some infinity loop in both mlrose package and mlrose_hiive package and didn't get a result when this paper submitted. But according to the performance of GA in Part I, I would assume that it has a better result compared to RHC and SA, but has longer running time compared to them.

Overall Performance & Running Time

The running time and overall F1 score of different algorithms are shown in figure 11. The F1 scores of the four algorithms only have slight differences and the running time of randomized optimization methods are much longer than that of back propagation, but the overfitting is lower in randomized optimization methods compared to back propagation.

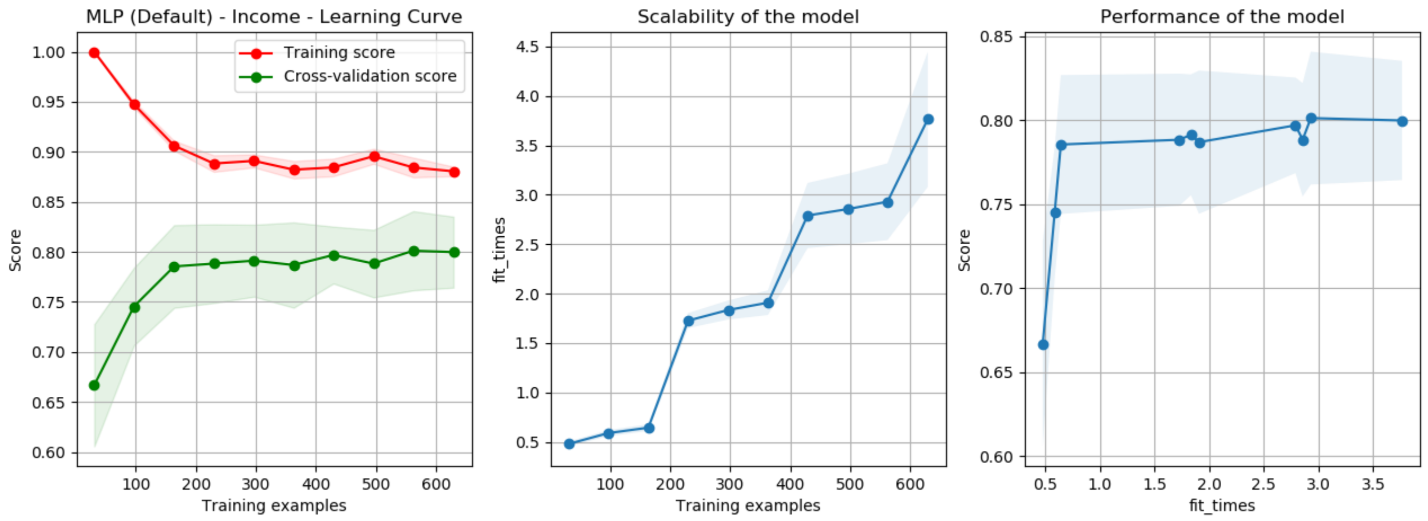


Figure 8. Results of NN using back propagation.

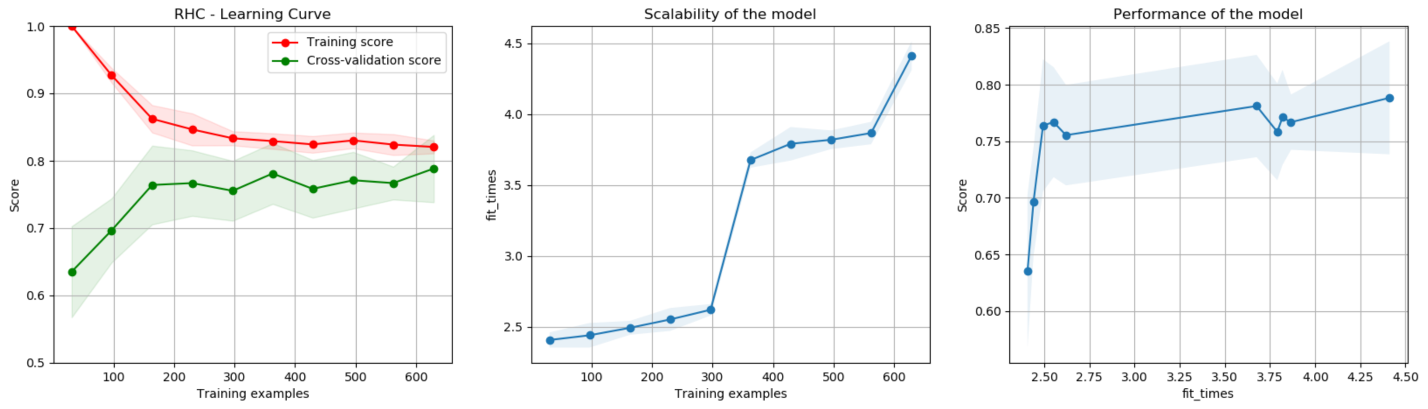


Figure 9. Results of RHC.

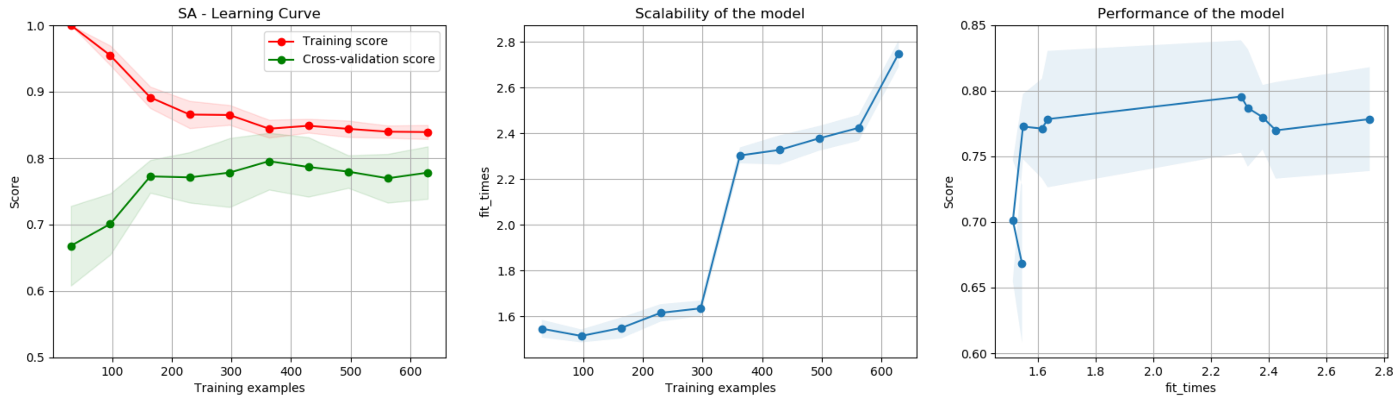


Figure 10. the result of SA.

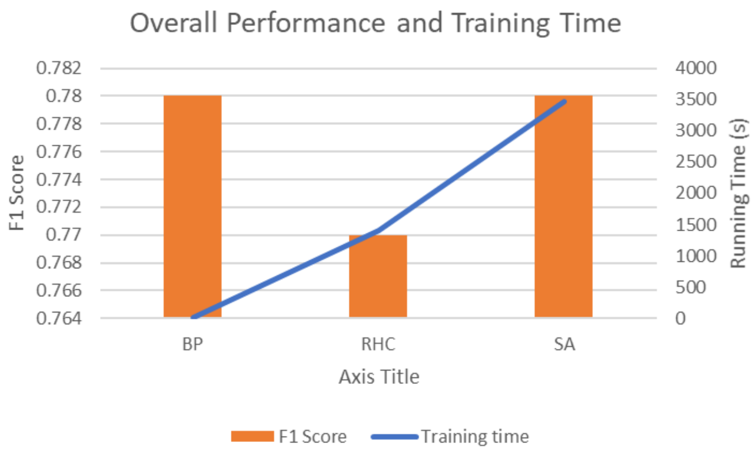


Figure 12. the overall F1 scores and running time for BP, RHC and SA.

DISCUSSION

To conclude the randomized optimization algorithms, they are suitable for complicated optimization problems while brute forcing is impossible and different question domains have different algorithms that performance the best. RHC is relatively fast but is tend to stuck in local maxima. SA and GA performs good on all problems discussed in this paper, but the parameters in these two algorithms need to be selected carefully. MIMIC performs good on the problems too and be able to escape from the local maxima, but the running time of MIMIC is substantially longer than any other algorithms and it cannot work in problems in continuous domain. So to select algorithms for a specific problem, we need to be careful and consider the object and complexity of the project.

References

[1] De Bonet, Jeremy S., Charles Lee Isbell Jr, and Paul A. Viola. "MIMIC: Finding optima by estimating probability densities." Advances in neural information processing systems. 1997.
[2] https://www.researchgate.net/post/What_is_the_role_of_mutation_and_crossover_probability_in_Genetic_algorithms