CS 7641 Randomized Optimization

Hyesung Ji (hji61@gatech.edu)

Abstract— The randomized optimization (RO) algorithm is widely used for discrete optimization problems. If the objective function does not have derivative(hard to be differentiated), or sample space is too big, a RO algorithm can be an efficient way to find optimal points. It searches the optimum point directly and does not require derivative of the objective function. In this project, each RO algorithm is analyzed and compared to each other. Additionally, these RO algorithms are compared to the back-propagation in terms of Neural Network. The detail of the result is discussed the below sections.

I. INTRODUCTION

Nowadays, there are four kinds of randomized optimization (RO) techniques widely used. These RO algorithms are Randomized Hill Climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA), and Mutual Information Maximizing Input Clustering (MIMIC). Each algorithm was applied to three discrete optimization problems. In order to compare and analyze the performance of each algorithm, the fitness score and learning time are computed. Additionally, RHC, SA, and GA algorithms are used to optimize weights of Neural network. The performance of Neural Network with weights from RO algorithms is investigated compared to the Neural Network using back-propagation method.

A. Discrete Optimization problems

1) 4 Peaks

The four peaks problem is to find global minima for the function $f(X,T) = \max[tail(0,X), head(1,X)] + R(X,T)$, where $tail(0,X)$ is the number of trailing 0's in X, $head(1,X)$ is the number of leading 1's in X, and $R(X,T) = N$ *or* 0 [1]. If the tail (0, X) >T and head (1, X)>T, $R(X,T)$ is N, otherwise it is 0 [1]. This problem evaluates the fitness of n-dimensional vector X and threshold parameter T. The threshold parameter can be calculated $T = t_{pct} \times N$.

2) Flip-Flop

Originally, the Flip-Flop problem was electric circuit problem. It evaluates the fitness of a state vector x and it counts the number of pairs of consecutive elements are not the same. If the vector is [0, 1, 0, 1, 1, 1], the number of Flip-flop is 3. If all of elements in the vector are different from each other, that is the maximum case. So, the maximum number of flip-flop problem is equal to $N - 1$.

3) N-Queens

The N-queens puzzle is to place chess queens and no two queens is able to attack each other. Each queen can attack other queens that are on the same row, column, or diagonal location. So, no two queens place on the same row, column, or diagonal location. The number of solutions was investigated by previous researchers. In this project, the number of non-attacking queen pairs is used to compute a fitness function [2]. The maximum value is $\binom{n}{2}$ [2].

B. Algorithms and hyperparameters

1) Randomized Hill Climbing (RHC)

Randomized Hill Climbing Algorithms (RHC) is to search optimal point of fitness function randomly. It picks a point/state(s) randomly and searches the neighborhood(x). If the fitness function of a neighborhood value ($f(x)$) is larger than $f(s)$, it picks the $f(x)$ as a maximum point. (In the case of minimum, it has the same logic.) It can easily stick at the local optima point and 'restart' can be a hyperparameter of this algorithm. So, the restart is selected as a hyperparameter to be investigated in this project. The range value of restart for each discrete optimization problem is from 0 to 25.

2) Simulated Annealing (SA)

Simulated Annealing (SA) is to search optimal point of fitness function probabilistically. It is similar as the RHC algorithm, but it has an opportunity not to fall in the local optima. The key is to searches the optimal point by using the probability $e^{-\frac{f(x_t)-f(x)}{T}}$. When $f(x_t)$ is larger than $f(x)$, it moves the point to $f(x_t)$. Otherwise, it moves the point to by using the probability. It can minimize the risk to fall in the local optima. There are three kinds of ways to decrease the value T. The Python library provides these three ways (Geometric decay, Arithmetic decay, and Exponential decay) and variable values. So. These ways are used for this project.

- Geometric Decay: $T(t) = \max(1 \times 0.99^t, 0.001)$
- Arithmetic Decay: $T(t) = \max(1 - 0.0001t, 0.001)$
- Exponential Decay: $T(t) = \max(T_0 \times e^{-0.005t}, 0.001)$

3) Genetic Algorithm (GA)

Genetic Algorithm (GA) is to find a best solution in an optimization problem. The selected 'parents' reproduces off-springs(solutions) by doing 'crossover' [3]. In the GA, there are two important hyperparameters. The set of current generation's solution is called 'population' and a small alteration of crossover is called 'mutation' [3]. A proper selection of population and mutation rate is very significant for GA algorithm to prevent local optima problem. So, population and mutation probability are selected as hyperparameters and investigated. It is restricted to search all values of hyperparameters. For convenience, the range of mutation probability is searched from 0.1 to 0.3. And the

range of population is searched from 200 to 800. In this paper, the mutation probability is calculated first and the population is calculated later. In the case of an initial computation of the mutation probability, the population is set to 100.

4) Mutual Information Maximizing Input Clustering (MIMIC)

The MIMIC algorithm is to generate a random 'population' uniformly came from the input space. The median fitness is extracted from the population, and it is called θ_0 [4]. And then, density estimator $(p^{\theta_0}(x))$ is made and samples are generated from distribution density estimator $(p^{\theta_i}(x))$ [4]. The value θ_i is updated according to the specific percentage, and the samples are kept less than θ_i. So, the 'population' and the 'specific percentage' can be hyperparameters for this algorithm. Similarly, for convenience, the range of the specific percentage (it is called "keep-pct") is searched from 0.1 to 0.3. And the range of population is searched from 200 to 800. Also, in the case of an initial computation for the keep-pct, the population is set to 100.

II. Application to Discrete Optimization Problems

As we discussed in the introduction, the three optimization problems are used for evaluating four algorithms. First, considering running time and memory, some trials and errors are implemented. After that, some common parameters are used the following.

- the maximum iteration of all algorithm: 1,000
- maximum attempts: 100
- the length of each problem: 100 and 50(only for N-Queens)

Second, hyperparameters of each algorithm are investigated and the hyperparameters which results in the best performance were selected. So, the best performance of each algorithm and learning time was compared each other.

A. 4-peaks

1) Randomized Hill Climbing (RHC)

As discussed in Introduction section, the restart is investigated hyperparameter in this project.

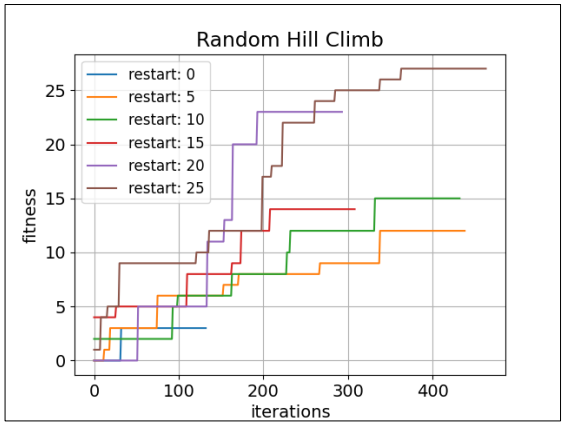

Figure 1. Fitness function of different restarts in RHC

As we can see in Figure 1, the performance of restart parameter has the best performance when the restart is 25. So, 25 is selected as a hyperparameter of RHC.

2) Simulated Annealing (SA)

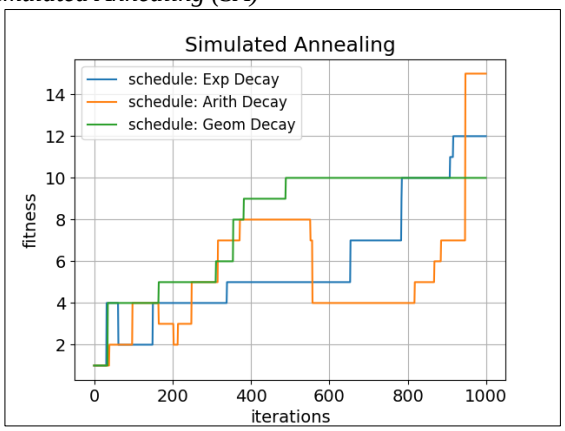

Figure 2. Fitness function of different Decay methods in SA

In Figure 2, the Arithmetic Decay method has the best performance. However, all of fitness score is too low.

3) Genetic Algorithm (GA)

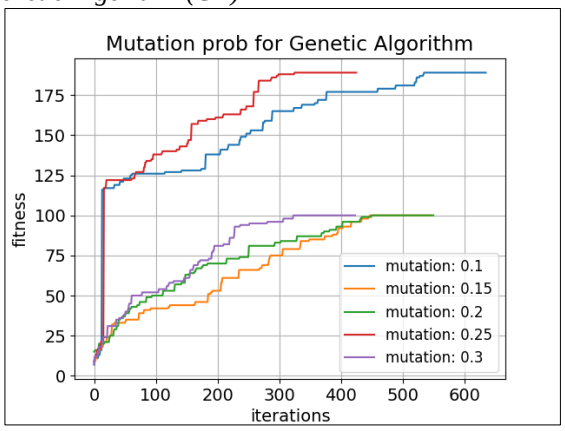

Figure 3. Fitness function of different mutation prob in GA

When the mutation probability is 0.25, the GA algorithm has the best fitness score in Figure 3. 0.25 is applied to the GA algorithm and the population is calculated.

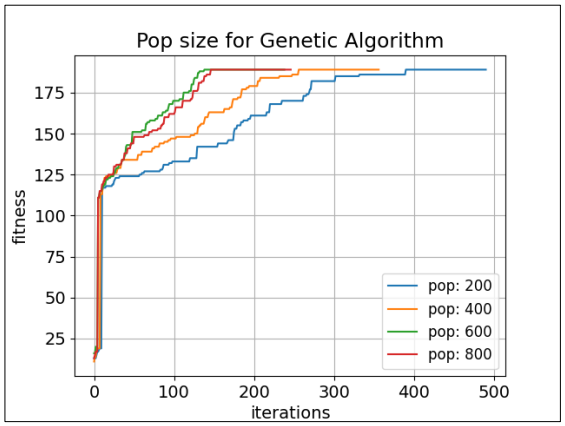

Figure 4. Fitness function of different populations in GA

All of populations converge to the same value in Figure 4. In this case, any number can be used as a hyperparameter. In terms of iteration, 600 can be the best hyperparameter.

4) MIMIC Algorithm (MIMIC)

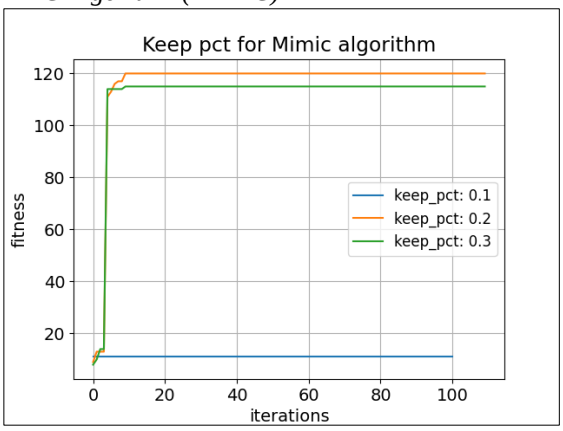

Figure 5. Fitness function of different keep-pct in MIMIC

In Figure 5, the best value of keep-pct is 0.2.

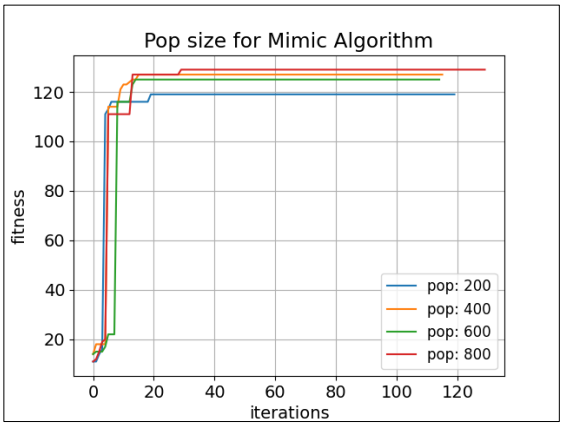

Figure 6. Fitness function of different populations in MIMIC

Figure 6 indicates that the population is 800 when the MIMIC algorithm has the best fitness score in this problem. Now, the performances and learning times are computed based on the results of these hyperparameters.

5) Comparison between each algorithm

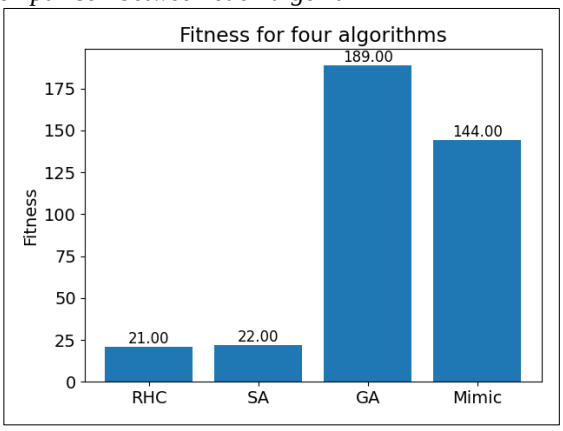

Figure 7. Best fitness of four algorithms

In terms of fitness score, the GA algorithm shows significantly excellent performance. Considering the characteristic of 4-peaks problem, this result can be reasonable. This problem is designed so that the classical methods such as RHC and SA can easily fall into the local optima [5]. Because it has sharp global optimal points and wide local points [5]. So, the elaborate algorithm can solve this problem properly. This is the reason that GA and MIMIC algorithm can have overwhelmingly higher performance than RHA and SA algorithm.

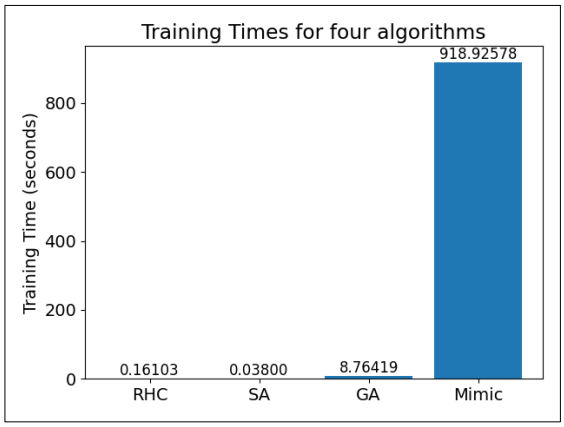

Figure 8. Time to train of four algorithms

In terms of train time, the MIMIC algorithm takes a long time to train. Even though the RHC and SA take significantly short time to train this problem than GA, the performances from the RHC and SA is very low and meaningless. Additionally, the fitness curve can show the difference between four algorithms.

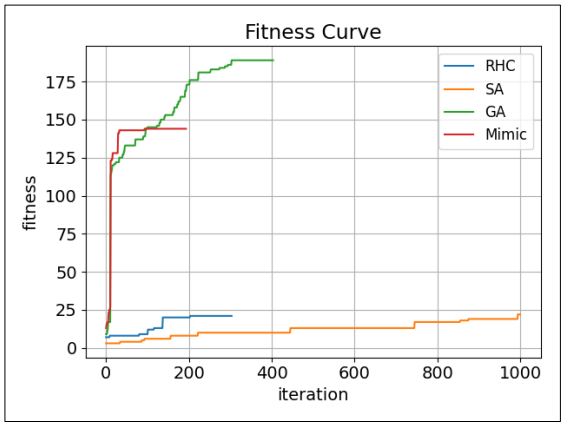

Figure 9. Fitness curves of four algorithms

Figure 9 also shows that the GA algorithm has the highest value among four algorithms.

<u>In conclusion, the GA is the best algorithm in the 4-peaks problem considering performance and time.</u>

Div.	RHC	SA	GA	MIMIC
Fitness	21.0	22.0	189.0	144.0
Time(sec)	0.1610	0.0380	8.7642	918.9258

B. N-Queens

1) Randomized Hill Climbing (RHC)

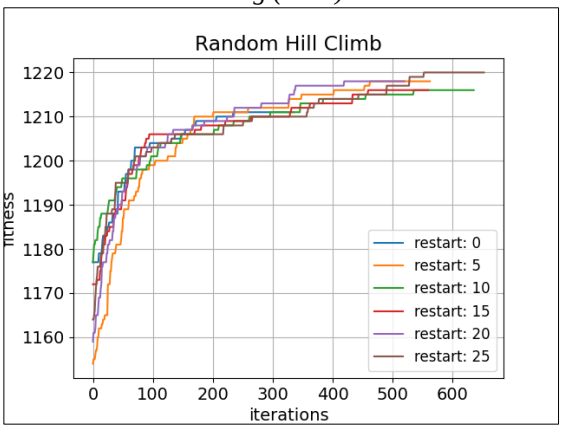

Figure 10. Fitness function for different restarts in RHC

In Figure 10, the restart values are similar to each other. Among these values, 25 is the best performance in this problem. So, the restart is selected 25.

2) Simulated Anealing (SA)

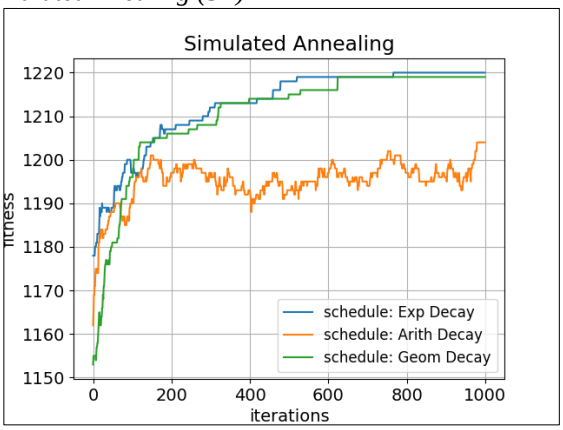

Figure 11. Fitness function for different Decay methods in SA

The exponential decay method has slight higher result than the geometric decay method in Figure 11.

3) Genetic Algorithm (GA)

Like the problem #1, population is set to 100 initially.

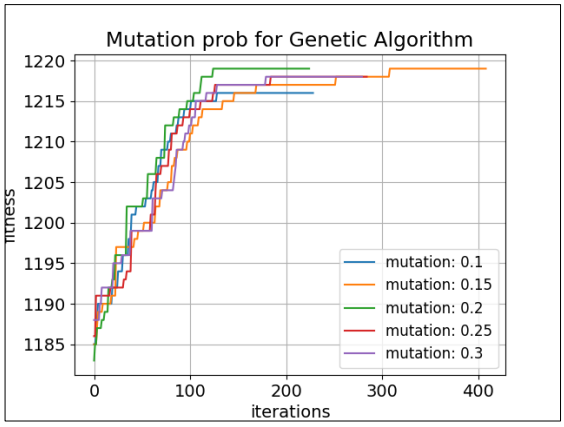

Figure 12. Fitness function for different mutation prob in GA

Figure 12 show when the mutation probability is 0.2, the GA algorithm has the best performance. So, 0.2 is applied to GA and the number of population is searched.

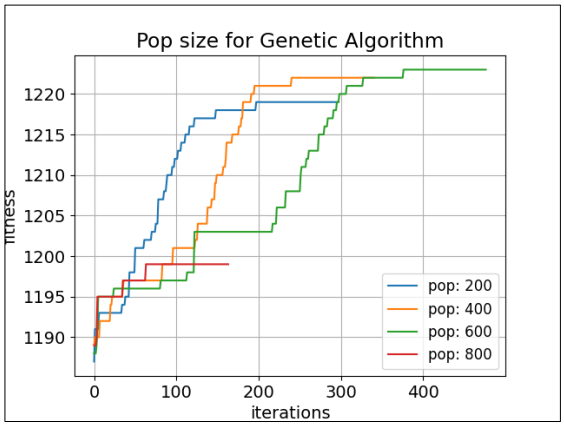

Figure 13. Fitness function for different populations in GA

In Figure 13, the number of population is 600. So, 0.2 and 600 are selected as the parameters in this problem.

4) MIMIC Algorithm (MIMIC)

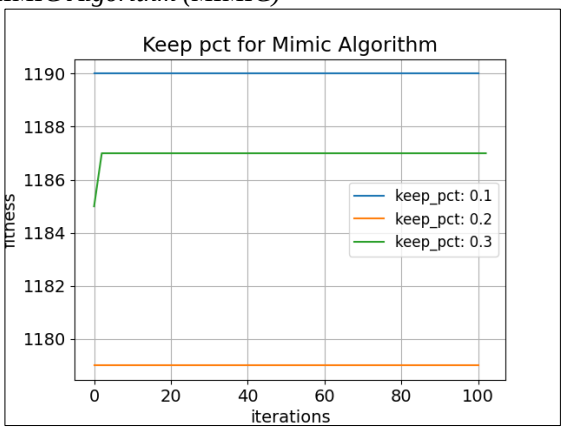

Figure 14. Fitness function for different keep-pct in MIMIC

In Figure 14, the best keep-pct value is 0.1 and it is applied to the MIMIC algorithm for this problem.

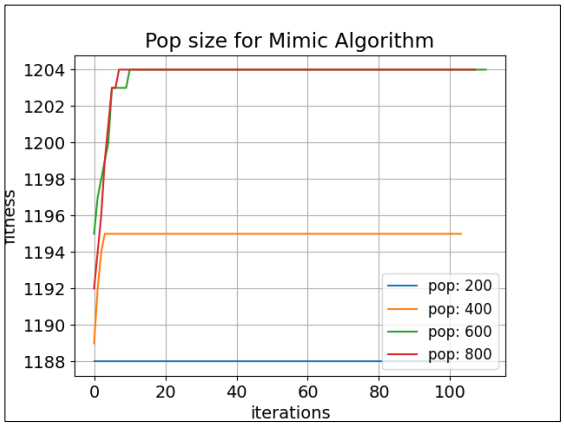

Figure 15. Fitness function for population in MIMIC

In Figure 15, the best performance of the MIMIC is 600 and 800. Even though 800 reached the highest value

slightly faster than 600, the performances are the same. Also, considering computation time, 600 populations can be faster than the 800 populations. So, the best number of the population is selected as 600. The hyperparameters of the MIMIC algorithm are 0.1 and 600 respectively for this problem. In terms of the number of populations, the smaller number is selected in the case of the same performance.

5) Comparison between each algorithm

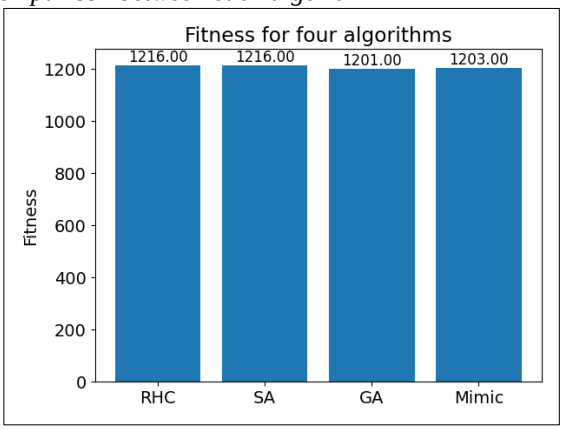

Figure 16. Best fitness of four algorithms

In terms of fitness score, the SA and RHC show the best performance among four algorithms. And the GA algorithm has the worst score in this problem. However, the differences between four algorithms are very small. There is just 1.2% difference maximally. The possible reason may be that N-queens problem is a classical problem and there are a few or no local optimal points. So, RHc and SA does not have risk to fall into local optima point. These algorithms can perform very well in a classic problem. On the other hand, GA and MIMIC algorithm always have to do a lot of computation and it cannot be an attractive algorithm in this case.

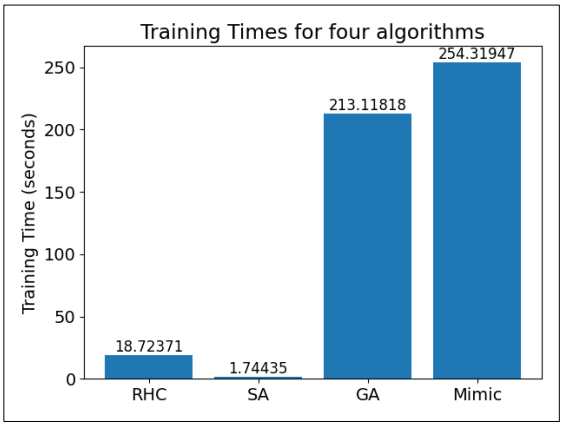

Figure 17. Time to train for four algorithms

In terms of time, the SA algorithm have the shortest time. And, the GA and MIMIC algorithms take a significantly longer time than RHC and SA. Compared to the RHC and SA, GA and MIMIC algorithm has to do a lot of computations to find optimal points. Even though, the fitness discrepancy is very slightly higher than other algorithms, there are a lot of difference in terms of time to train. Additionally, we can see the differences between four algorithms in terms of fitness curve.

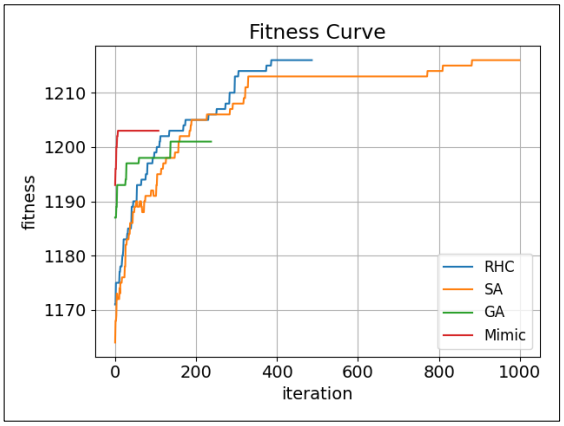

Figure 18. Fitness curves of four algorithms

In terms of the number of iterations, the RHC looks the best algorithm. However, it takes 18 times longer than the SA algorithm. In this case, the SA algorithm so fast that the number of iterations cannot be a significant factor. The reason SA algorithm takes more iterations than RHC is that the SA algorithm uses the probability with the temperature T. Even though the T is decreased gradually, the probability $e^{-\frac{f(x_t)-f(x)}{T}}$ can significantly be big in the beginning step. So, there are some random moves in the beginning step. As the iteration is increased and the temperature is decreased, there is a possibility for the SA algorithm to improve the performance.

In conclusion, the SA is the best algorithm in the N-Queens problem considering performance and time.

Div.	RHC	SA	GA	MIMIC
Fitness	1216.0	1216.0	1201.0	1203.0
Time(sec)	18.7237	1.7443	213.118	254.319

C. Flip-Flop

1) Randomized Hill Climbing (RHC)

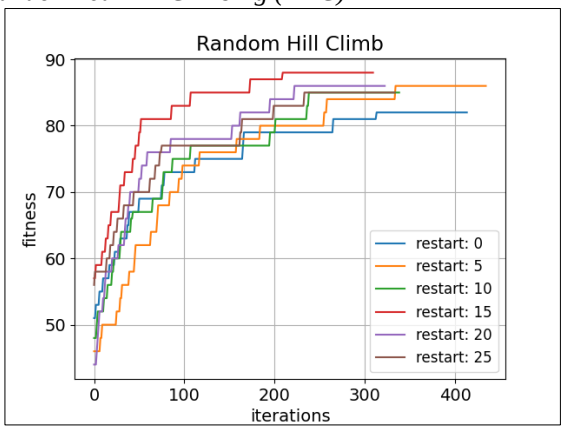

Figure 19. Fitness function of different restarts in RHC

As we can see in Figure 19, when the restart is 15, the RHC algorithm has the best performance.

2) Simulated Annealing (SA)

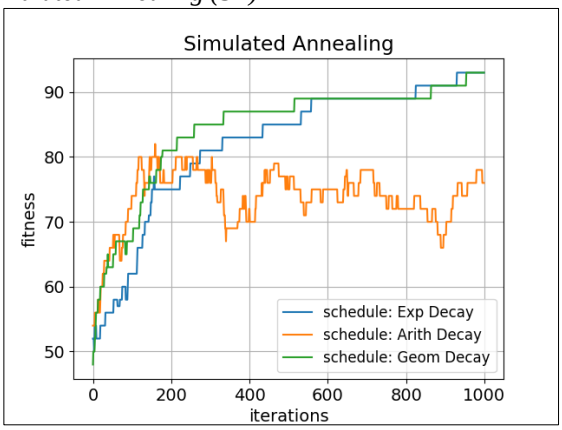

Figure 20. Fitness function of different Decay methods in SA

The exponential decay and geometric decay method have the almost same result. The exponential decay is slightly faster than the geometric method, so the exponential decay method can be used in this problem.

3) Genetic Algorithm (GA)

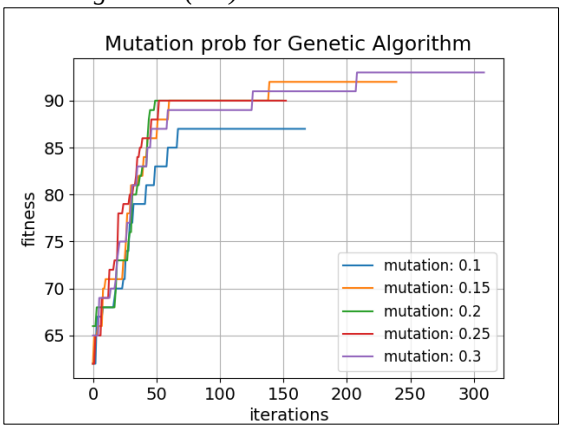

Figure 21. Fitness function of different mutation prob in GA

In Figure 21, when the mutation probability is 0.3, the GA algorithm has the best value. So, it is used to the best mutation probability.

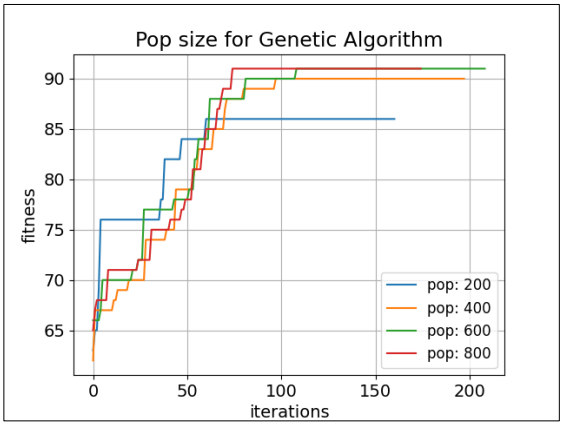

Figure 22. Fitness function of different populations in GA

The fitness function has the maximum value when the population is 600 and 800. As described in Figure 15, the smaller population is preferred and 600 is selected.

4) MIMIC Algorithm (MIMIC)

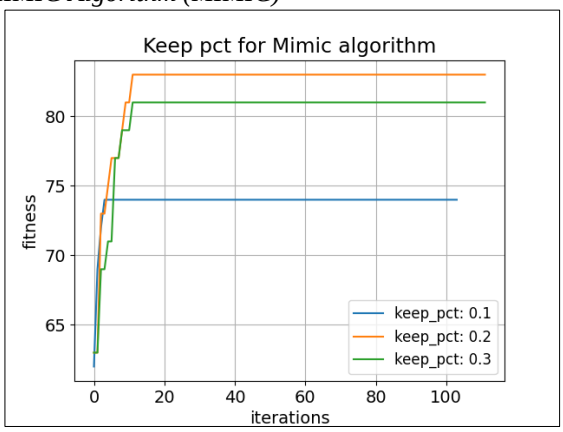

Figure 23. Fitness function of different keep-pct in MIMIC

In Figure 21, 0.2 has the best performance for this problem. So, 0.2 is selected as the best hyperparameter.

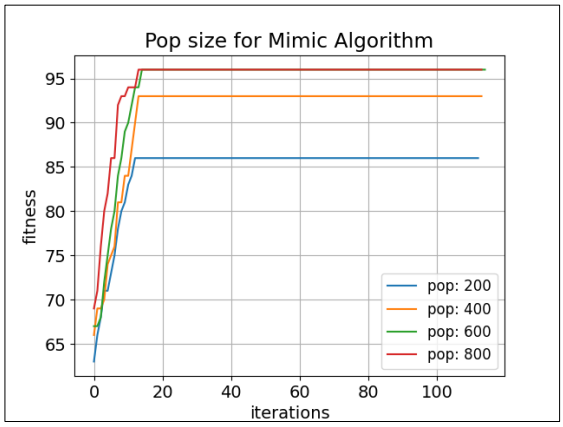

Figure 24. Fitness function of different population in MIMIC

In Figure 22, The population 600 and 800 are the same values. As stated above, 600 can be the hyperparameter as a number of population.

5) Comparison between each algorithm

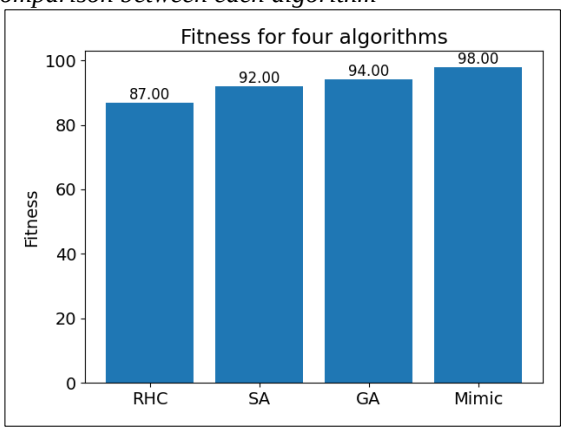

Figure 25. Best fitness for four algorithms

We can see the best fitness in Figure 23, the MIMIC algorithm has the highest value (98). And next is GA, SA, and the worst algorithm is RHC. In terms of fitness score, the MIMIC is the best algorithm in the flip-flop problem. Next, we need to consider the training time.

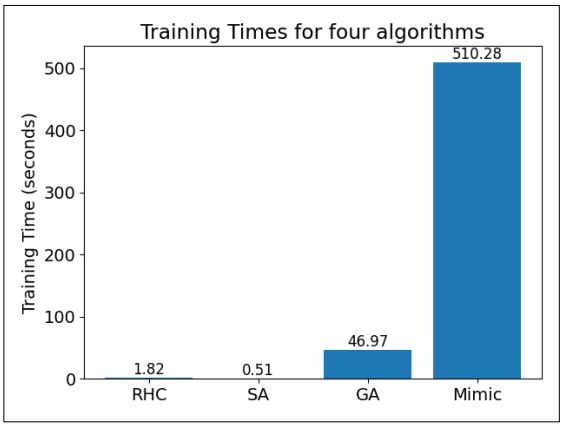

Figure 26. Time to train for four algorithms

In terms of time the MIMIC algorithm has the longest time to train. Contrast to the MIMIC algorithm, the RHC and SA algorithm take very short time to train. If we evaluate a problem quickly, the RHC or SA algorithm can be considered.

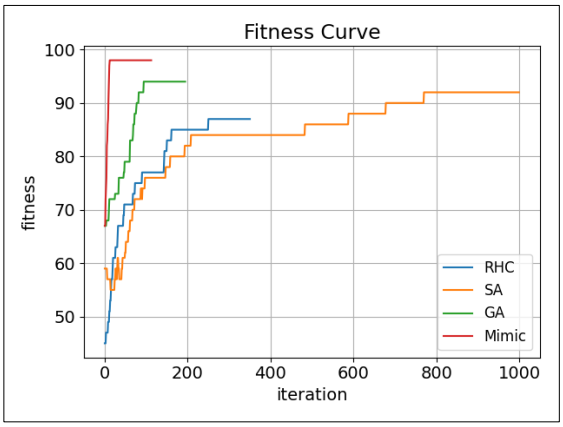

Figure 27. Fitness curves of four algorithms

In terms of number of iterations, the MIMIC is also the best algorithm. Even though the MIMIC algorithm takes so long time to find optimal solution, it can find with a smaller number of iterations.

In conclusion, the MIMIC is the best algorithm in the Flip-flop problem considering performance and number of iterations.

Div.	RHC	SA	GA	MIMIC
Fitness	87.0	92.0	94.0	98.0
Time(sec)	1.82	0.51	46.97	510.28

III. Application to Neural Network

In the Project 1, I implemented and compared five classifiers including neural network. In this section, one of two datasets that are used in the project 1 is selected. The dataset is 'WiFi-localization' dataset [6]. The reason to select 'WiFi-localization' is the dataset has four classes. Rather than binary classification, the multi-class classification may result in more interesting performance. After some trials errors to avoid spending too much time to find best solution, the some hyperparameters for neural network are set to like the below.

- the maximum iteration of all algorithm: 1,000
- maximum attempts: 100
- Learning rate range: 0.001 ~ 10 (divided by 8 points)

A. *Randomized Hill Climbing (RHC)*

The learning rate is very important hyperparameter in the Neural Network. So, it is very important to determine the learning rate in advance. The number of restart is set to 1 and then the learning rate is calculated. As investigated above section, the range of restart is from 0 to 20.

1) *Learning rate*

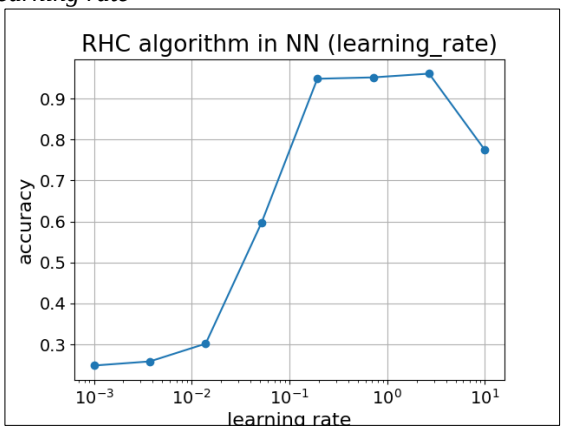

Figure 28. Learning curve for different learning rates in RHC

Figure 28 show that the RHC is very sensitive to the learning rate. When the learning rate is less than 0.1 or greater than 10, the accuracy is significantly decreased. The learning rate is 2.6827 when the accuracy is the best.

2) *The number of restart*

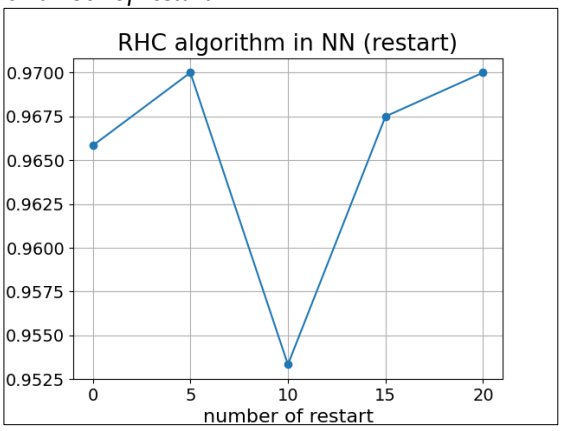

Figure 29. Learning curve for different restart number

As we can see in Figure 29, the variation of learning curve is very slight. This means that the objective function of this dataset is not complex and there are just a few local optima points. It means that the probability to fall into the local optima is slight. When the number of restart is 5 and 20, they have the same performance. So, the number of restart is selected as 5.

B. *Simulated Annealing (SA)*

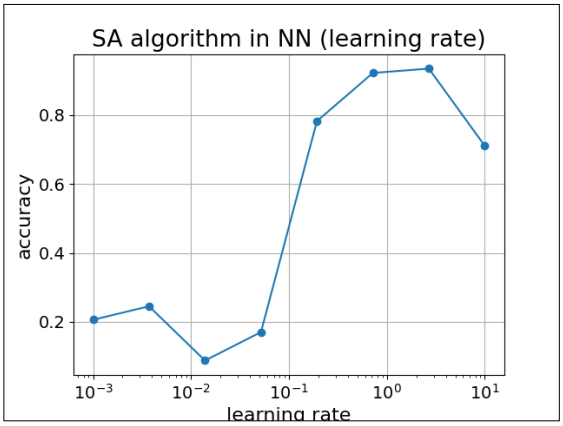

Figure 30. Learning curve for different learning rates in SA

The affect by learning rate is very similar to RHC algorithm. In Figure 30, the best learning rate is 2.6827 and it is applied to the SA algorithm. In terms of the decay method, the geometric and exponential methods are very close each other as stated above section. In this problem, the geometric decay method is applied.

C. *Genetic Algorithm (GA)*

For the GA algorithm, the mutation probability and the number of population should be investigated. The mutation probability is searched from 0.0 to 0.2 and the number of population is searched from 200 to 800. To compute the learning rate, the mutation probability is set to 0.1 and the number of population is set to 500.

1) *Learning rate*

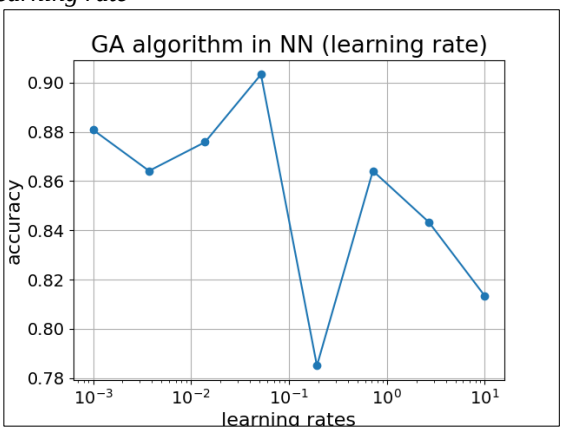

Figure 31. Learning curve for different learning rates in GA

In Figure 31, the variation of accuracy by the learning rate is not too significant compared to the RHC and the SA. In this case, the learning rate is selected as 0.0518. And the population is set to 200 to compute the mutation probability.

2) *Mutation probability*

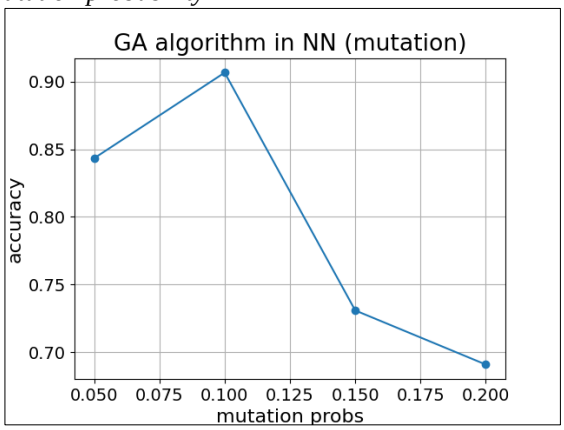

Figure 32. Learning curve for different mutation prob in GA

When the mutation probability is 0.1, the GA algorithm has the best performance. So, 0.1 is selected as the mutation probability.

3) *The number of population*

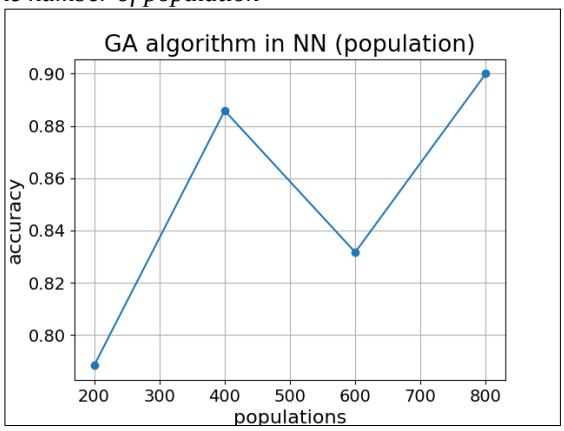

Figure 33. Learning curve for different populations in GA

As we can see Figure 33, when the population is 800, the GA has the best performance. So, 800 is applied to the GA algorithm to find the weights for the Neural Network.

D. *The hyperparameters of each algorithm*

This Back-propagation method was studied in the project 1. So, the result of the experiment is used this project again and the below table is the finalized parameters to be used.

Div.	RHC	SA	GA	BackProp
Hidden layer (No of nodes)	30	30	30	30
Learning rate	2.6827	2.6827	0.0518	0.00278
Other values	Restart: 5	GeomDecay	Mut. Prob: 0.1 Pop: 800	Regul.param: 0.001 Optimizer: Adam

E. *Comparison between each algorithm*

As stated above section, the performance of each algorithm is compared in terms of accuracy and time to train.

1) *Accuracy score*

The accuracy score is the most effective factor to evaluate the performance of an algorithm.

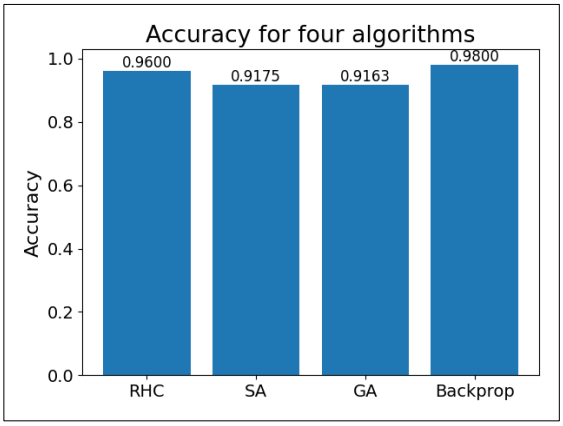

Figure 34. Accuracy score of each algorithm

Figure 34 shows that the Back-propagation algorithm is the best algorithm for this dataset. Compared to RHC and Back-propagation, the SA and GA have relatively poor accuracy. It is very hard to discern why SA and GA algorithm has poor results. The possible reason is that the dataset has a few or no local optima. So, the computation of moving probability or cross-over method are not necessary to find the global optima.

2) *Time to train*

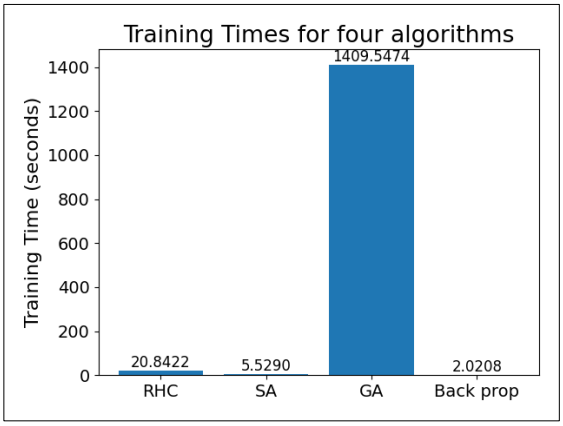

Figure 35. Training time of each algorithm

In Figure 5, the Back-propagation is the shortest algorithm and next is the SA algorithm. Considering the accuracy score and time both, the Back-propagation is the best algorithm for this problem. On the other hand, the GA algorithm is time-consuming algorithm. It takes a long time and the accuracy is not good as the RHC/Back-propagation. The GA algorithm can be a proper algorithm for significantly complex problem which has a lot of local optima points.

3) *Loss function of each algorithm*

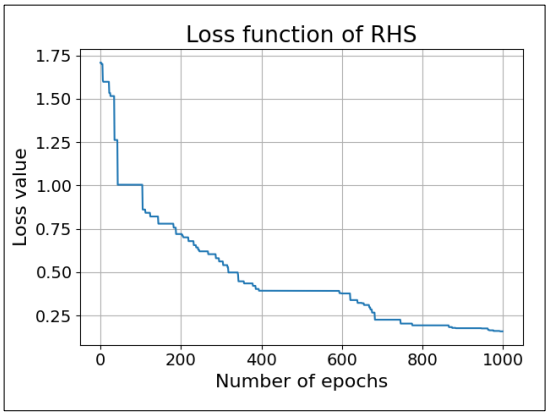

Figure 36. Loss function of RHC algorithm

The loss function of RHC shows that the RHC algorithm performs very well for this problem. However, it restarts several times to avoid local optima, it takes more training time than the SA or Back-propagation algorithm.

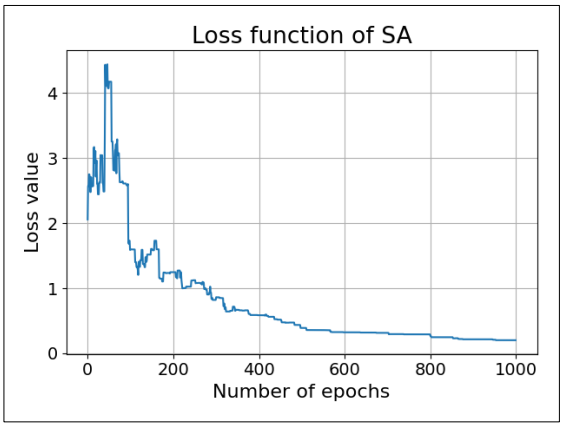

Figure 37. Loss function of SA algorithm

Figure 37 shows that the fluctuation of the SA algorithm when the number of epochs (iterations) is small. As discussed in Figure 18, the beginning step of the SA algorithm, the SA algorithm is computing a proper temperature, so, the loss function can be fluctuated. It can affect negatively to the final accuracy score. However, there is a possibility to improve the accuracy if the iteration number is significantly greater than 1,000.

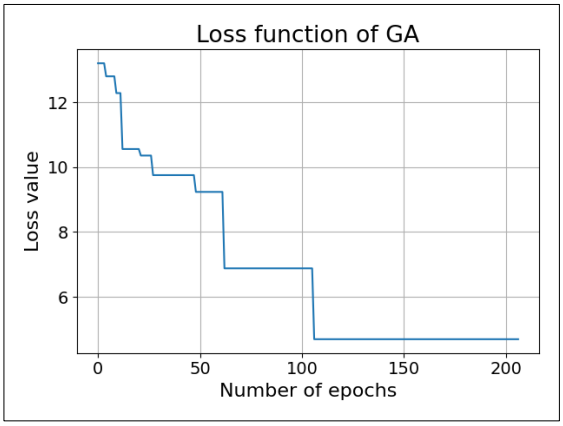

Figure 38. Loss function of GA algorithm

In Figure 38, the length of loss function for the GA algorithm is shorter than RHC or SA algorithm. The GA algorithm seems to determine to stop the training stage the loss function is not improved significantly.

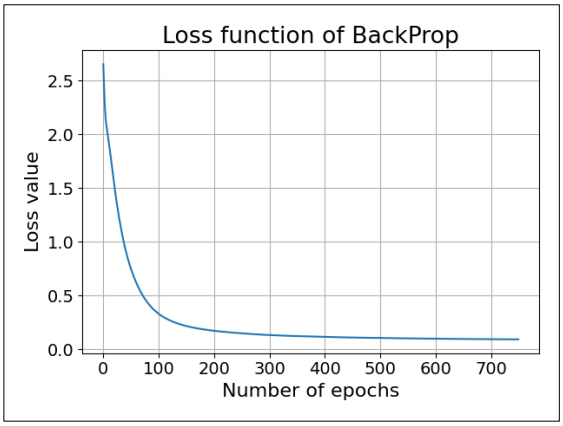

Figure 39. Loss function of Back-propagation

The Back-propagation algorithm performs very well for this problem. It seems to find the global optima points very fast and precisely in this problem.

IV. CONCLUSION

In this project, four algorithms are investigated based on three discrete problems and three RO algorithms and Back-propagation method are compared to each other. In summary, GA and MIMIC algorithms have normally relatively higher fitness than RHC/SA algorithms for discrete problems. GA/MIMIC algorithms are searching a lot of probability, and these processes can improve the accuracy in many cases. However, GA and MIMIC algorithms have a common important drawback. These two algorithms take a lot of time to train and the RHC/SA algorithms take a significantly short time to train. Therefore, before we decide to apply which algorithms to the problem, we need to investigate the problem. If the problem is not very complex, the RHC/SA algorithms can be more effective algorithm than GA/MIMIC. Otherwise, GA/MIMIC algorithms can be applied to the problem despite its drawback.

For the Neural network problem, the dataset does not have a very huge space, and the objective function may not have local optimal points. So, the RO algorithms were not very efficient in this case. The back-propagation algorithm performs very well and it has a significantly short training time than the RO algorithms. A possible reason why the back-propagation algorithm performs better than the RO algorithm may be this problem is a continuous problem, not discrete. Normally, the discrete problem can be more complex than the continuous problem, because a lot of combinations should be computed to solve discreate problem [7]. On the other hand, the gradient value for the domain can be necessary to solve the continuous problem. Therefore, a gradient-based method can be fast and precise than the randomized algorithm [7]. However, if the dataset is very huge or the objective function is not differentiated, the randomized algorithm can be considered. Therefore, we need to determine which algorithm is applied to a specific problem depending on the dataset and a characteristic of the dataset's objective function.

In this project, some topics should be improved in depth. First, the exact fitness functions of three discrete problems and the dataset were not investigated or calculated. To be specific, the clear reason of which algorithm can perform well is not described. Also, the reason the back-propagation performs better than the RO algorithms is not stated very reasonable. For the next time, the characteristic of the fitness function (objective function) should be analyzed more detail.

REFERENCES

[1] Randomized Local Search as Successive Estimation of Probability Densities (2006)

[2] Mukherjee, Soham, Santanu Datta, Pramit Brata Chanda and Pratik Pathak. "Comparative Study of Different Algorithms to Solve N Queens Problem." FOCS 2015 (2015).

[3] https://www.tutorialspoint.com/genetic_algorithms/

[4] Bonet, Jeremy S. De, Charles Lee Isbell and Paul A. Viola. "MIMIC: Finding Optima by Estimating Probability Densities." NIPS (1996).

[5] A.J. Keane, Genetic algorithm optimization of multi-peak problems: studies in convergence and robustness, Artificial Intelligence in Engineering, Volume 9, Issue 2, 1995, Pages 75-83

[6] UCI Machine Learning Repository: Wireless Indoor Localization Data Set

[7] A. Parkinson, Richard J. Balling, and John D. Hedengren. Optimization Methods for Engineering Design, 2013, Bringham Young University, Chapter 4