CS 7641 Machine Learning

Chen Peng

Assignment 2: Randomized Optimization

1. Introduction

In this assignment, I studied 4 randomized optimization algorithms, the randomized hill climbing (RHC), the simulated annealing (SA), the genetic algorithm (GA), and the mutual-information-maximizing input clustering (MIMIC). The four algorithm are implemented with the python package mlrose.

1.1 Randomized hill climbing (RHC)

Randomized hill climbing (RHC) is a mathematical optimization technique that belongs to the category of local search. It starts with a random solution to the problem, and continuously finds better solutions through incremental changes of the nearest neighbors, until no improvement can be found. RHC climbs the hill iteratively, starting with a random initial guess each time, and moving to the direction of increasing fitness in each iteration. This algorithm occupies very little memory and is optimal in convex problems.

1.2 Simulated annealing (SA)

Simulated annealing (SA) is a non-deterministic global optimal hill climbing algorithm. SA simulates this process by occasionally accepting a function of decreasing fitness, so it always has a chance to escape the local optimum. One of its parameter "T" is temperature. The higher the temperature, the greater the probability of accepting a poor solution. Generally, T decreases with the extension of the algorithm running time. In theory, the algorithm can always find the global optimal solution, but for some problems, its running speed is very slow. In practical applications, how to determine the rate of reducing T will become a problem.

1.3 Genetic algorithm (GA)

Genetic algorithm is a meta-heuristic algorithm inspired by the process of natural selection, which belongs to the broad category of evolutionary algorithms. Genetic algorithms allow mutations and crossovers on the two parental chromosomes of a population, and put new offspring into the new population, thereby obtaining the best solution. It iterates until it finds the optimal solution for the given function. It can perform a random search on the population to find solutions that cannot be found by the local search algorithm, without knowing the derivative of the fitness function. However, this method may not be very good for complex problems.

1.4 Mutual-Information-Maximizing Input Clustering (MIMIC)

MIMIC is a kind of distribution estimation algorithm, which belongs to the general category of evolutionary algorithms. MIMIC first randomly selects samples from the region most likely to contain the optimal solution of the input space, and then uses an effective density estimate, which can be used to capture a variety of structures in the input space, and through simple data Second-order statistics are calculated.

2. Travelling Salesperson Problem

The TSM problem is the most basic route planning problem, and it is also a classic NP-Hard problem. The problem is to find the minimum path cost for a single traveler to start from the starting point, pass all the given demand points, and finally return to the origin. The earliest mathematical programming of the traveling salesman problem was proposed by Dantzig (1959). The most obvious algorithm is the exhaustive method, that is, to find all combinations and choose the shortest. The number of permutations of this algorithm is (n-1)! (where n is the number of nodes).

In this assignment, I try to solve a TSP of 20 cities, the location of the cities are integrals with x and y range of [1,20]. I tested 4 algorithms with different parameters. The results are shown in the Figure1:

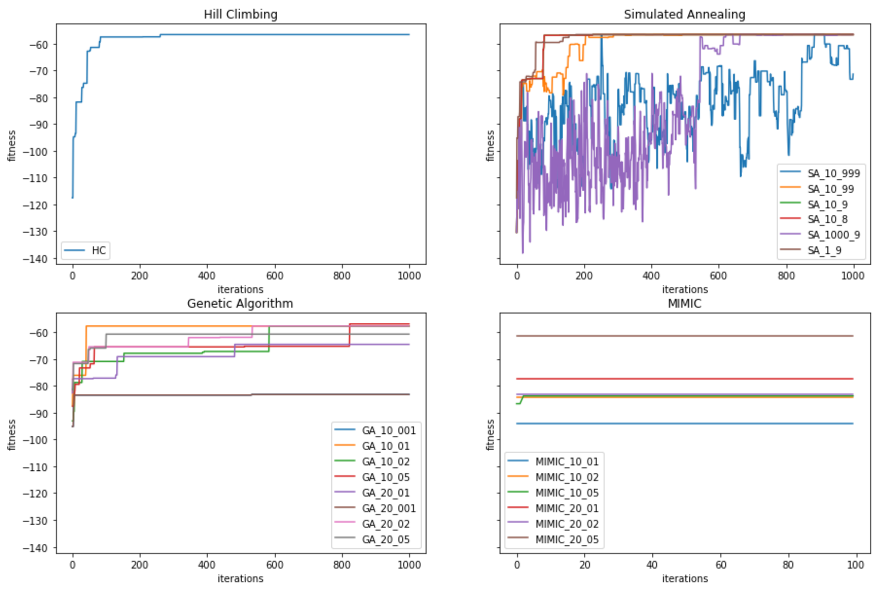

Figure 1, Fitness v.s. iterations of four algorithm (Hill Climbing, Simulated Annealing, Genetic Algorithm, MIMIC) for Travelling Salesperson Problem.

As the figure 1 shows, the best parameters for SA is temperature=10, the schedule is geometrically decaying the simulated annealing temperature parameter T. The decay factor is 0.8. For GA, the best parameters are population size is 10, and the mutation probability is 0.5. The best parameters for MIMIC is population size is 20, and proportion of samples to keep at each iteration of the algorithm is 0.5.

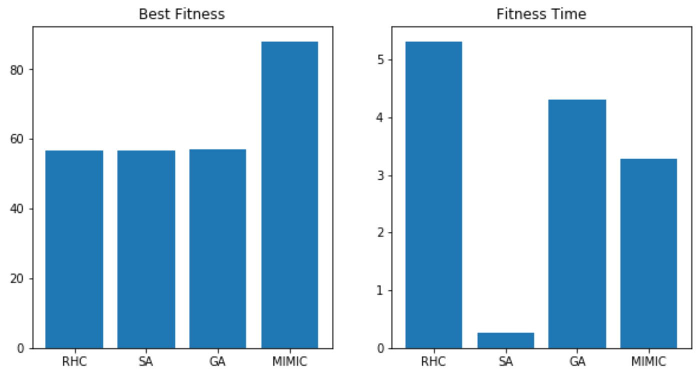

Figure 2, Comparison of the 4 algorithms for Travelling Salesperson Problem. Best fitness (left), and fitness time (right).

Figure 2 shows the fitness score (left) of the 4 algorithms with the best parameters of performance. The right figure shows the fitness time of each algorithms. For RHC, the restart time is 20. For MIMIC, due to the quick converge shows in Figure 1, the maximum iteration is set to be 100 here. In the Travelling Salesperson Problem, the best fitness score is the total route distance. The smaller value of fitness score represents the better performance. As we can see, the RHC, SA, GA obtained almost the same fitness score, while the MIMIC performs worst in this problem. For the fitness time, the SA obtained the shortest computational cost of time. The other 3 algorithms take almost same amount of time. However, as we can see in Figure 1, the SA are not as stable as other algorithms. If we change the initial temperature or the decay factor of the SA a little bit, the algorithms might easily obtain a much worse result. Therefore, the best algorithm for the Travelling Salesperson Problem is Genetic Algorithm.

3 Flip Flop Problem

Flip Flop Problem is a problem of calculating the number of bits transformation in a bit string. That is, any number from one number to the next is recorded as 1. The maximum fitness bit string should be a bit string composed entirely of alternating numbers.

In this assignment, I try to solve problem length of 100. The maximum iteration and attempts is 1000. I tested 4 algorithms with different parameters. The results are shown in the Figure3:

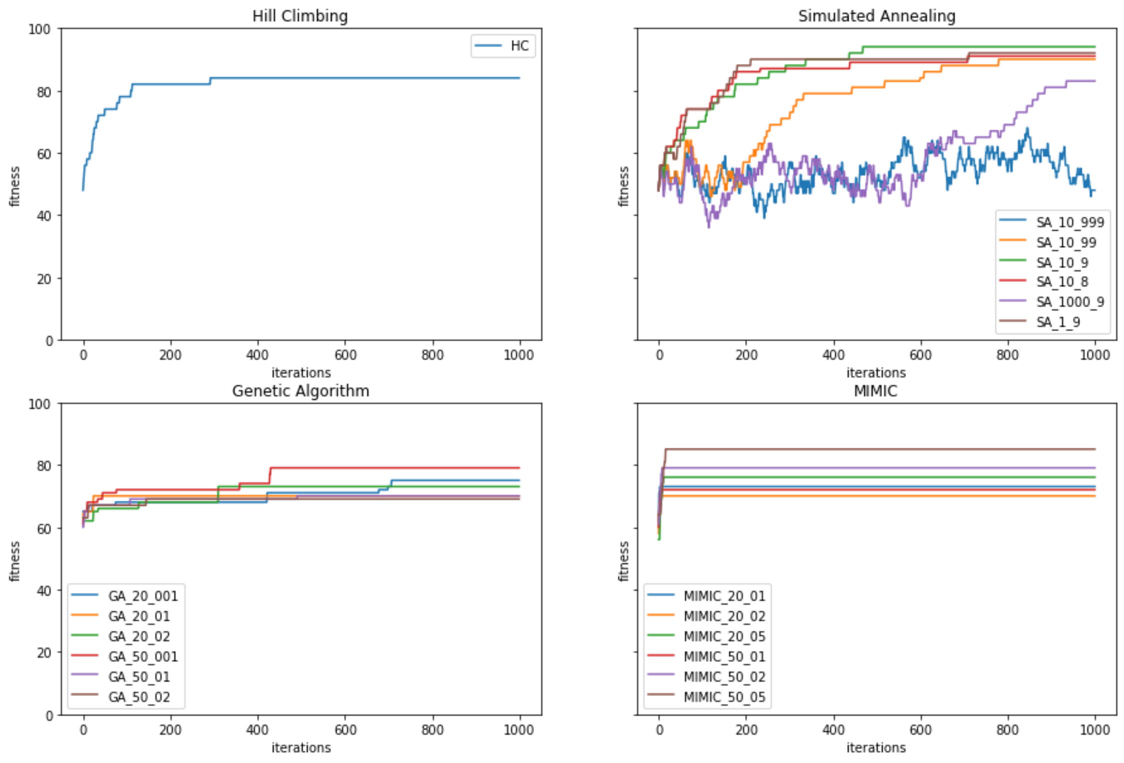

Figure 3, Fitness v.s. iterations of four algorithm (Hill Climbing, Simulated Annealing, Genetic Algorithm, MIMIC) for Flip Flop Problem.

As the figure 3 shows, the best parameters for SA is temperature=10, the schedule is geometrically decaying the simulated annealing temperature parameter T. The decay factor is 0.9. For GA, the best parameters are population size is 50, and the mutation probability is 0.01. The best parameters for MIMIC is population size is 50, and proportion of samples to keep at each iteration of the algorithm is 0.5.

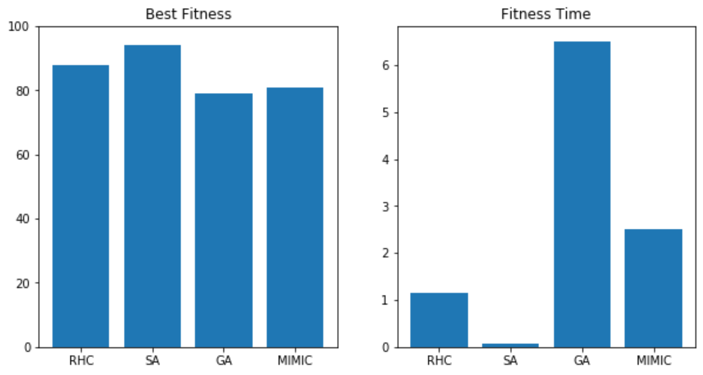

Figure 4, Comparison of the 4 algorithms for Flip Flop Problem. Best fitness (left), and fitness time (right).

Figure 4 shows the fitness score (left) of the 4 algorithms with the best parameters of performance. The right figure shows the fitness time of each algorithms. For RHC, the restart time is 20. For MIMIC, due to the quick converge shows in Figure 3, the maximum iteration is set to be 100 here. In the Flip Flop Problem, the best fitness score is the total flips of the string. The larger value of fitness score represents the better performance. As we can see, the SA get the best fitness score. The following algorithms are RHC, MIMIC, and GA. For the fitness time, the SA obtained the shortest computational cost of time. The GA takes the longest fitness time. As a result, the SA performs best for the Flip Flop Problem. This is because this problem can be solved very well with the greedy algorithm with good computational efficiency. The GA and MIMIC also can get good results, but they take much longer time.

4. Knapsack Problem

Knapsack problem is a NP-complete problem of combinatorial optimization. The problem can be described as: Given a set of items, each item has its own weight and price, within the limited total weight, how do we choose to make the total value of the item the highest. It was proposed by Merkle and Hellman in 1978.

In this assignment, I try to solve problem with 50 different packs. The weight and value of each pack is a random integer with range of [1, 10]. The maximum weight percentage is 50% of the total weight of the 50 packs. The maximum iteration and attempts is 1000. I tested 4 algorithms with different parameters. The results are shown in the Figure5:

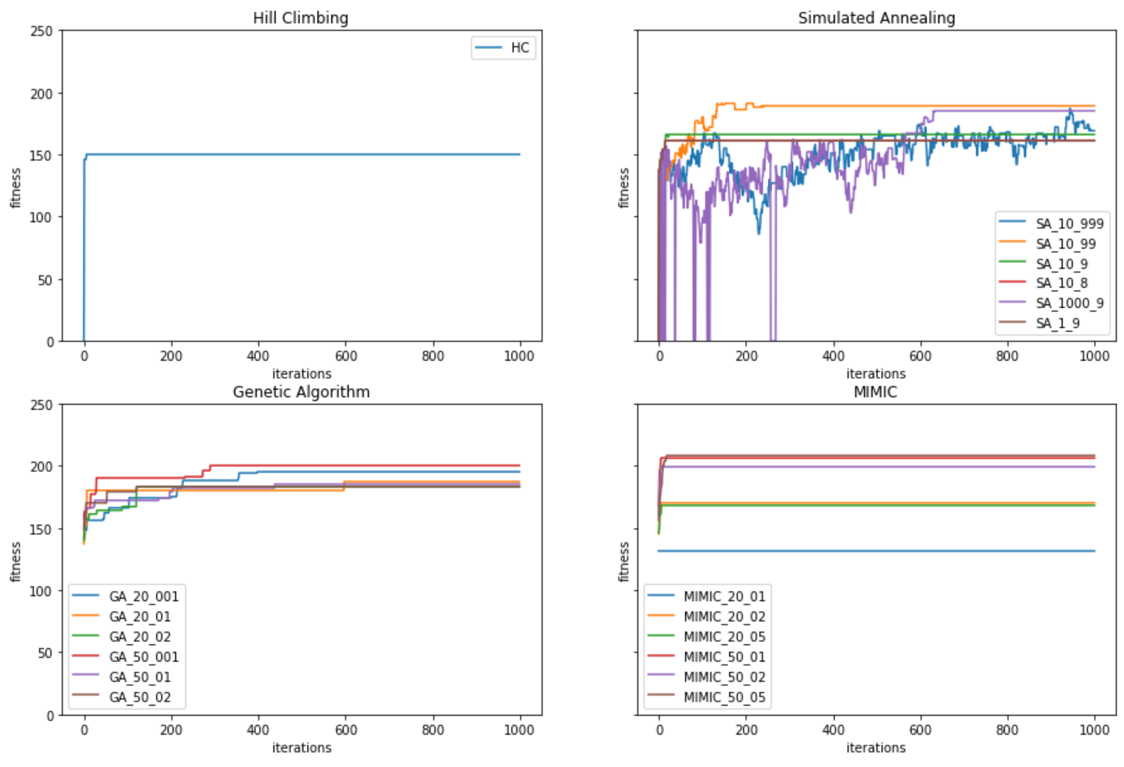

Figure 5, Fitness v.s. iterations of four algorithm (Hill Climbing, Simulated Annealing, Genetic Algorithm, MIMIC) for Knapsack Problem.

As the figure 5 shows, the best parameters for SA is temperature=10, the schedule is geometrically decaying the simulated annealing temperature parameter T. The decay factor is 0.99. For GA, the best parameters are population size is 50, and the mutation probability is 0.01. The best parameters for MIMIC is population size is 50, and proportion of samples to keep at each iteration of the algorithm is 0.5.

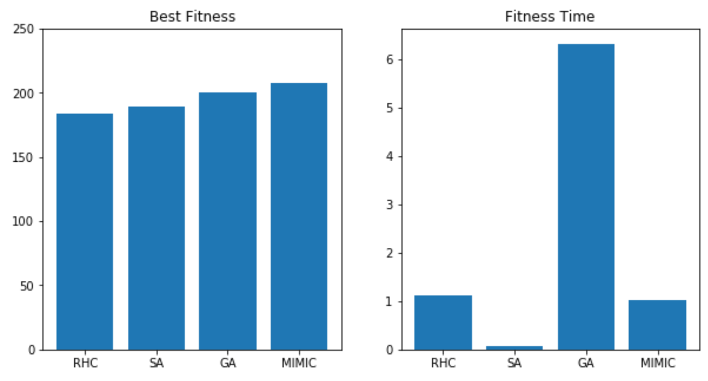

Figure 6, Comparison of the 4 algorithms for Knapsack Problem. Best fitness (left), and fitness time (right).

Figure 6 shows the fitness score (left) of the 4 algorithms with the best parameters of performance. The right figure shows the fitness time of each algorithms. For RHC, the restart time is 20. For MIMIC, due to the quick converge shows in Figure 5, the maximum iteration is set to be 100 here. As we can see, the MIMIC obtained the highest fitness score, and the RHC get the lowest. For the fitness time, the SA runs quickly, the RHC and MIMIC are also acceptable. The GA take much longer than the others. Considering the fitness score, the best algorithm for Knapsack Problem MIMIC. It gives the highest score with acceptable time cost. The greedy algorithms perform not as well as the GA and MIMIC, this is due to the complexity of the estimation space, and they are easy to fall into a local optimum in this problem. GA and MIMIC can avoid the local optimum problem better.

5. Neural Network Weight Optimization

In this part, we choose one of the datasets from assignment 1. The Dataset is from the UCI Machine Learning Repository, which is an early stage diabetes risk prediction dataset. The dataset contains 16 attributes (Age, Gender, Polyuria, Polydipsia, sudden weight loss, weakness, Polyphagia, Genital thrush, visual blurring, Itching, Irritability, delayed healing, partial paresis, muscle stiffness, Alopecia, Obesity) for 520 diabetic patients. The problem is to find the target labels, which are detection of the disease containing of 320 positives and 200 negatives.

To make predictions, one way I used in the assignment 1 is to use the Multi-layer Perceptron classifier in scikit-learn package to implement the neural network algorithm. The neural network use ReLU activation function with 1 hidden layer. The hidden layer has 10 neurons. The ReLU activation function is a linear function. In this assignment, I try 4 different optimization algorithms to optimize the neural network and compare the results of the 4 algorithms. The first algorithm is the back propagation with adam solver, which is implemented with the scikit-learn package. The other 3 algorithms, RHC, SA, and GA are implemented with the mlrose package.

For the back-propagation algorithm, the learning rate is 0.0001. The total iteration number is 2000. For randomized hill climbing, the maximum iteration is 2000, the maximum attempts is 100, the learning rate is 0.1, and the weights range is limited to [-5, 5]. For simulated annealing, the maximum iteration is 2000, the maximum attempts is 100, the learning rate is 1, and the weights range is limited to [-5, 5]. For genetic algorithm, the maximum iteration is 2000, the maximum attempts is 100, the learning rate is 1, and the weights range is limited to [-5, 5], the mutation probability is 0.1.

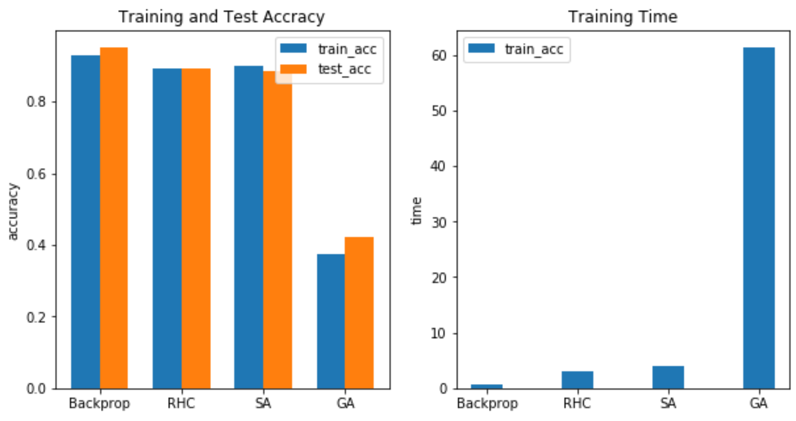

Figure 7, Comparison of the 4 algorithms for neural network weight optimization problem. Best fitness (left), and fitness time (right).

Figure 7 shows the results of the 4 algorithms to the problem. The left figure shows the training and testing accuracy of the 4 algorithms. As we can see, the back-propagation obtained the highest training and testing accuracy. The RHC and SA also work well results to the problem. The GA gets the lowest accuracy on both training and testing set. Considering the problem is a binary classification problem, and the two classes are almost well balanced, the result for GA shows that the model learns almost nothing from the data. This may because the GA needs more iterations and the mutation probability is too small for this problem. The right figure shows the training time of each algorithms. As we can see, the back-propagation algorithm implemented with scikit learn package takes the shortest time. The possible reasons are: first, the back-propagation algorithm are great for optimization the neural network weight; second, the back-propagation algorithm in the scikit-learn package is optimized and more efficient for execution. The RHC and SA has the second and third shortest time for training process. The GA take the longest time. This result shows the GA are not good for the neural network weight optimization. The best algorithm among RHC, SA, and GA for weight optimization is RHC, which has the highest accuracy and shortest training time.

6. Conclusions

In this assignment, I try 4 different optimization algorithms for 4 different problems. Different algorithms can work well on some scenarios than other.

In general, the greedy algorithm like RHC and SA run fast. They can work well in some problem with the function space not too complicated. However they are easy to fall into local optima in some problems.

For SA, it needs to tune the parameters very carefully. The results is hard to converge if the temperature is too high. The result is very easy to fall into local optima and performs like the hill climbing if the temperature decay is too quickly.

The GA and MIMIC can work well for complicated problems. However, they usually take much longer computational time to get good results than the greedy algorithms.

For GA, the results sometimes fluctuates and hard to converge.

For MIMIC, it can converge to a good result within a few iterations. Therefore, if we cut the iteration number, we can fit the problem in a short time with good results in some problem.

Reference

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.

Islam, MM Faniqul, et al. 'Likelihood prediction of diabetes at early stage using data mining techniques.' Computer Vision and Machine Intelligence in Medical Image Analysis. Springer, Singapore, 2020. 113-125.

Code source:

All of the code that use in the assignment is available on my dropbox. Here is the link, and I provide a security code: cpeng78, in case of improper transmission.

https://www.dropbox.com/sh/emvi9jvwz8oymlb/AAB7271wh27kCQ6IyKNmFMxNa?dl=0