CS 7641 Assignment 2

Student: Yinglin Li (yli973)   Email: allamli@gatech.edu

1. Neural Network Training Test

1.1. Dataset Description

Here I used the Magic Gamma Telescope dataset in assignment 1. The data are generated to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging processing technique. As the result, it was somehow to describe the features of the image. The dataset has totally 19020 instances with 10 attributes of real number as input. The labels were 'g' = gamma particle and 'h' = hadron particle. However, from the previous assignment, we could know that it extracts the real features, such as height, width of the images from the original pixels so there are only 10 features for each instance, and it had a good performance in the accuracy in all the 5 models in assignment 1.

1.2. Overview of Neural Network Test

In assignment 1, we have used Back Propagation to train Neural Network for Magic Gamma Telescope dataset. Here we used other three optimization algorithms, Randomized Hill Climbing (RHC), Simulated Annealing (SA) and Genetic Algorithm(GA) to solve the Neural Network without all the effort of feedforward. The size of feedforward network was: input layer = 10, hidden layer = 7, output layer = 1. The parameters of each algorithms were shown below.

Algorithm	Parameters	Training Accuracy	Testing Accuracy	Time(s)
Back Propagation	-	95%	93%	-
RHC	Iterations = 6000	78.378%	78.155%	65.57

SA	Iterations = 6000
	T = 100
	Cooling = 0.80
		77.822%	77.656%	70.948

GA	Iterations = 6000
	Population = 200
	Mute = 100
	Mutation = 10
		71.753%	72.976%	6448.96

Table 1.1. Overview of Neural Network Test

In table1.1, we could see that compared to the BP method, the three optimization algorithms could get a good performance but could not reach the accuracy as the BP Neural Network. Among the three algorithms, RHC could get the best accuracy and the best running time. SA's performance was close RHC's but its testing accuracy was higher than the training accuracy. GA got the worst accuracy and had a very slow running time. Detailed experiments would be shown below.

1.3. Detailed Analysis

In figure 1.1 and 1.2 below, we could see that RHC and SA started to converge within the 2000 iterations. But for the GA, it stated to converge much earlier within the iterations about 500. In both 2 graphs, the error of SA algorithm would increase first and then decreased to converge. That was because of the temperature was too high to make it easy to accept the neighbor candidate.

However, in the learning curves figure, we could see that the curves of testing accuracy were very close to the training curves. That was the reason of there were too limited data for the algorithm to get an accurate structure of neural network. As the result, three of the optimization algorithms got a worse performance then the BP Neural Network.

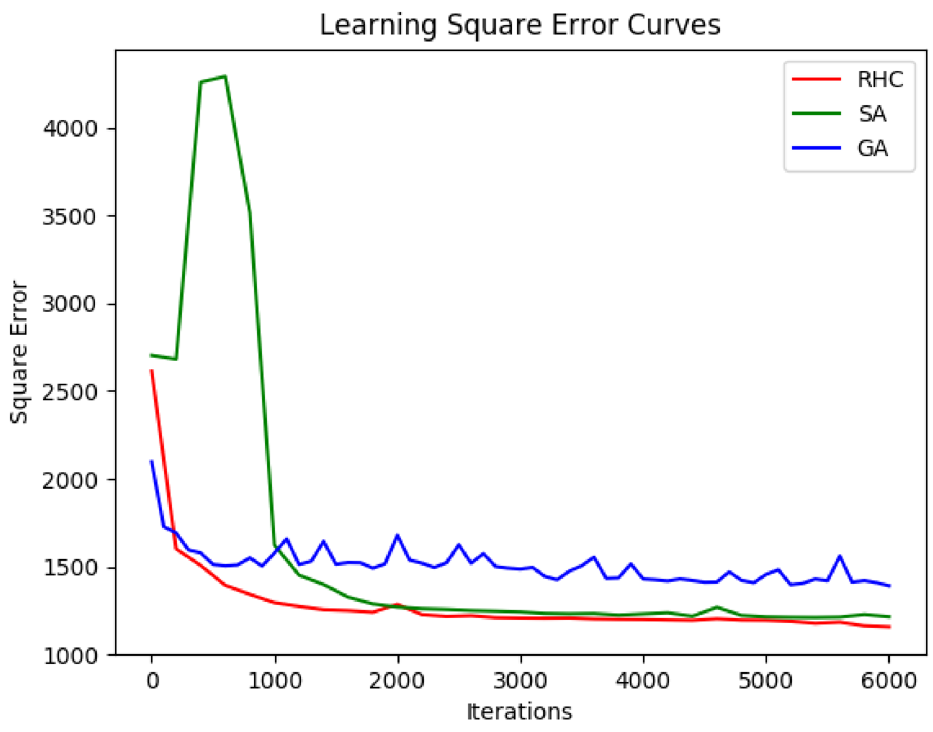

Fig 1.1. Square Error Curves

1.3.1. Randomized Hill Climbing (RHC)

In the learning curve of figure 1.1, we could see that at first the square error curve converged very fast and it was fairly stabilized after about 2000 iterations. After that, the trend of the curve became smooth which indicated that a local optimum had been found and it couldn't find another optimal peak.

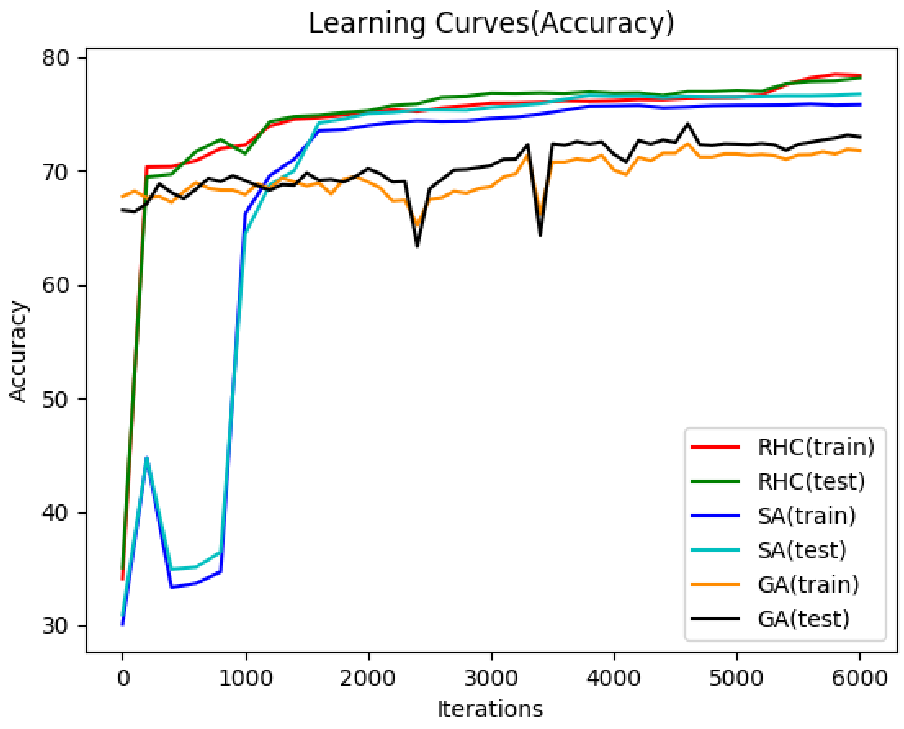

Fig 1.2. Learning Curves of Accuracy

Compared to the other two optimization algorithms, RHC got the best performance. And thanks to the randomized picking neighbor point, it could get to the local optimal peak quickly. However, because of the bad structure of the test data, it could not reach the global optimal peak as the BP network. As the result, it was not suitable to deal with this problem comparing to BP network.

1.3.2. Simulated Annealing (SA)

Since SA was a kind of Hill Climbing algorithm with improvement of probability to pick the neighbor point, its performance on this dataset was close to RHC.

In the figure 1.3, we made cooling = 0.95 as constant to test the best temperature T for this SA neural network. At last we decided to pick T = 100 because too large T might make SA become random picking points like RHC.

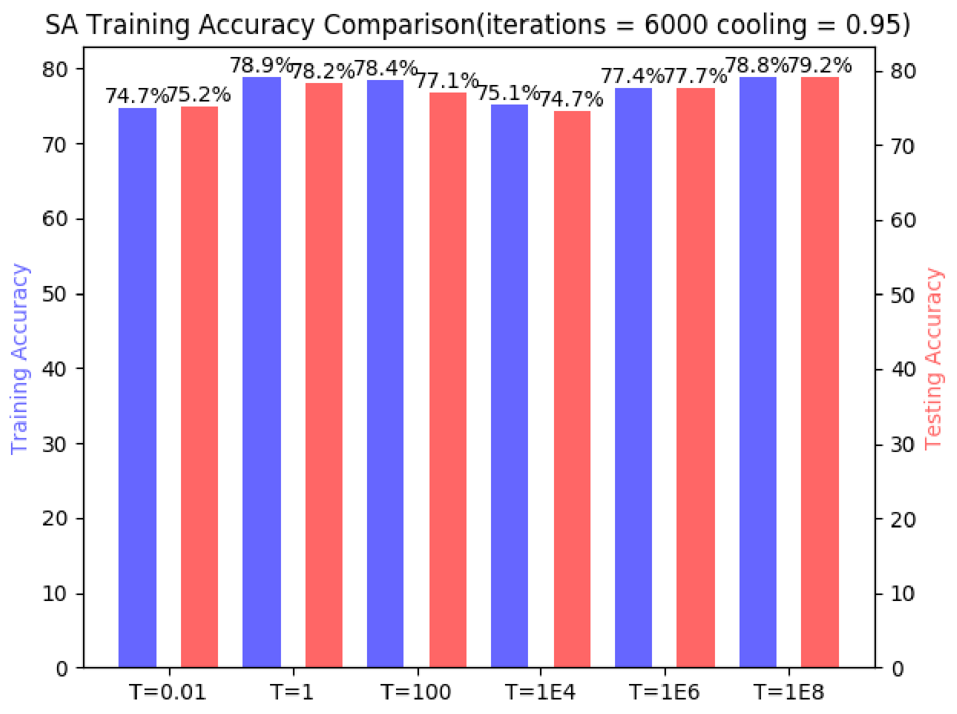

Fig 1.3. SA Accuracy Comparison

In the figure 1.4, three different values for cooling rate were tested. We could find that the algorithm tried to explore the space more and care less about performance in all of the three curves. And we could also find that lower cooling rate would lead to converge faster and higher one needed more iterations to converge but it would become more smoothing after converging.

As the result, for SA algorithm, we chose T = 100 and cooling = 0.8 as the best parameter for the input of this dataset.

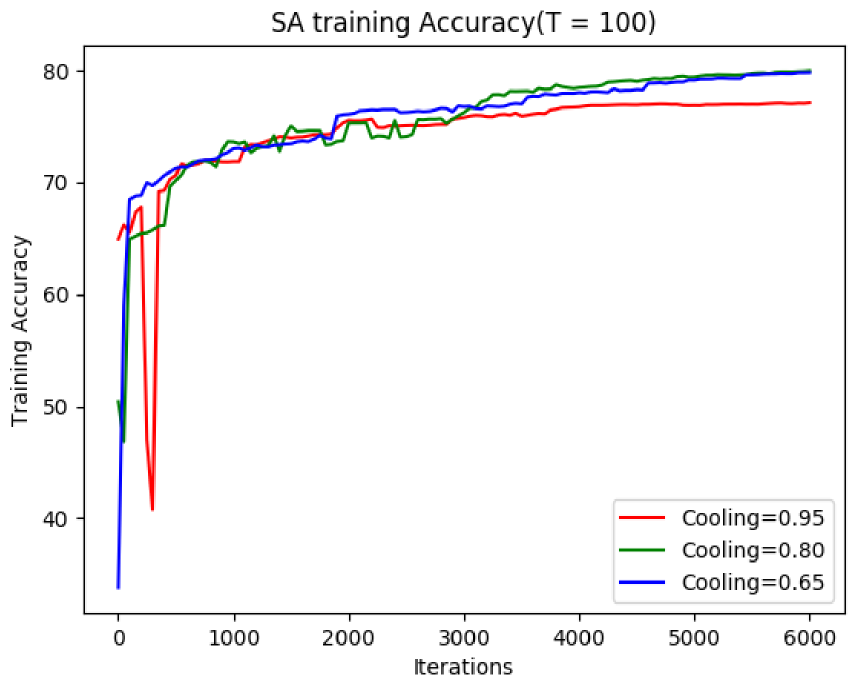

Fig 1.4. SA Training Curves Comparison

1.3.3. Genetic Algorithm (GA)

For GA algorithm, it got the worst score both in accuracy and running time. In the learning curve of figure 1.2, we could also find that the test accuracy was always close to the training accuracy. And when we check the training dataset, we could find that there were 12332 instances were 0 ('g') and 6688 instances were 1 ('h'), which means their ratio was 65:35. Therefore, we could know that the Neural Network could not label the instance correctly in many cases.

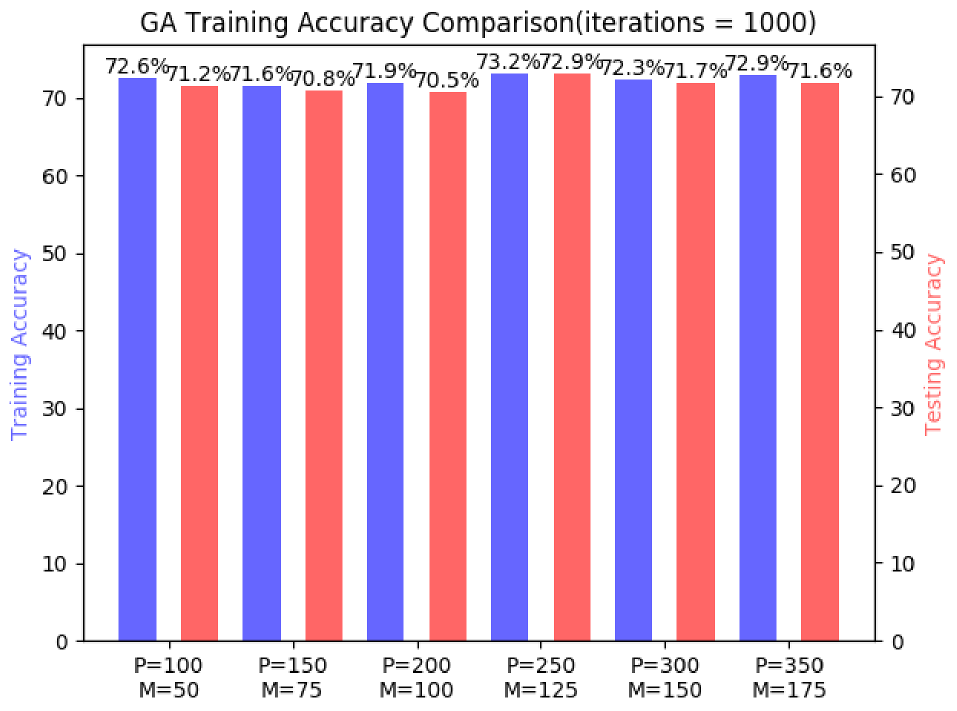

Fig 1.5. GA Accuracy Comparison

To get more proof, we could see the figure . From this figure, we could see that changing the population size and mating size could not bring an improvement of the accuracy to the algorithm. As the result, we could conclude that GA feedforward network was not suitable for this dataset.

The reason was that all the 10 attributes were continuous in this dataset, which would make a huge hypothesis space. As the result the crossover of muting process could not represent such a huge space and it had to converge at the local optimum with a not good accuracy.

2. Optimization Problems

2.1. Flip Flop Problem

2.1.1. Problem Description

Flip Flop is the problem to count how many bits that are different from its next neighbor bit in a binary bit string. The best solution is obvious the bit string with alternating bits with their next neighbor, which means it is 010101…. However, we could see each bit in the string as an individual attribute. As the result, there will be many local maximums for the problem.

2.1.2. Overview

In the test, the test iterations were 6000 for each algorithm. To make sure the stability, for RHC, SA and GA, it ran 20 times to get the average and for MIMIC, it ran 5 times. In the test, we chose the length N = 200

Algorithm	Parameters	Evaluation Value	Time(s)
RHC	Iterations = 5000	163.9	0.0022

SA	Iterations = 5000
	T = 100
	Cooling = 0.95
		184.45	0.0030

GA	Iterations = 4000
	Population = 200, Mate = 100
	Mutate = 10
		168.1	0.48

MIMIC	Iterations = 200	171.4	5.77

Table 2.1. Overview of Performances in Flip Flop

Among the four algorithms, SA had the best evaluation values and its running time was very close to the fastest one RHC but RHC had the worst score. The evaluation values of GA and MIMIC were similar but MIMIC converged within 200 iterations but GA converged at about 4000.

2.1.3. Detailed Analysis for Simulated Annealing

Here let me explain why SA can get a better performance to Flip Flop problem comparing to the other three. When the problem has many local optimums that are close to each other, SA works best to solve this situation. In a Flip Flop problem, each neighbor for SA was a flip of one bit in the string. It seems that RHC should work as well as SA, but however because of randomized process, RHC would be get to the local optimum then restarted in another point. Since it has a large search space for N = 200, and it needs to find an approximate global optimum, rather than a precise local optimum, GA and MIMIC got a close value evaluation but they were lower than SA.

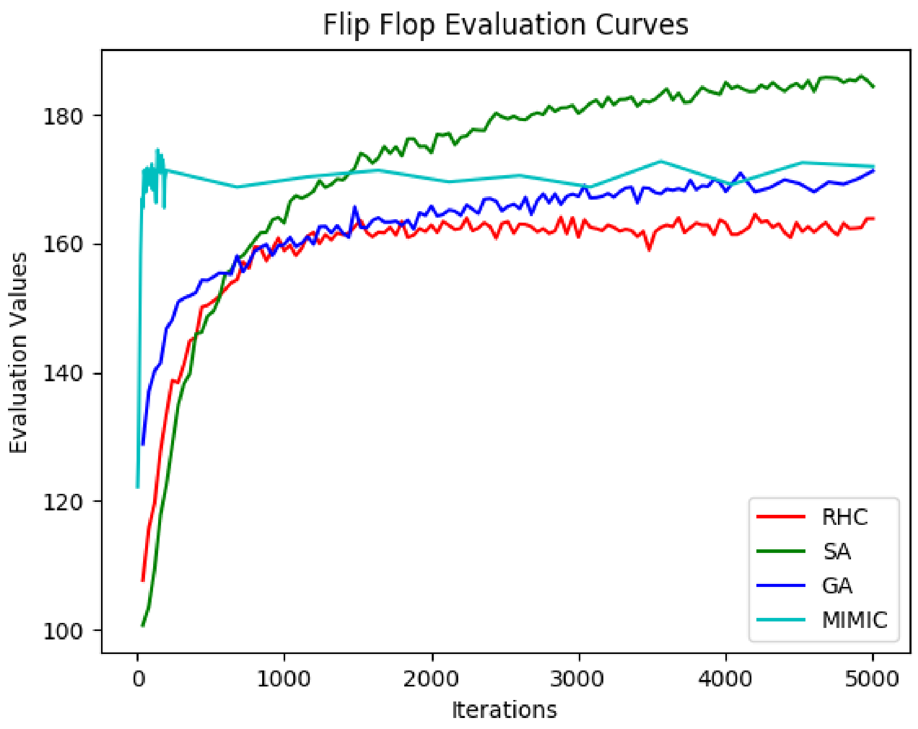

Fig 2.1. Flip Flop Evaluation Curves

For SA, with proper temperature and proper cooling rate, it can easily skipped local optimums and get to a better result. And in figure 2.1, RHC, GA and MIMIC would stop at that local optimum so they converged much earlier and their performances were much worse than SA.

2.2. Traveling Salesman Problem

2.2.1. Problem Description

The Travelling Salesman problem is a NP-Hard problem which has no polynomial time solution. Given a set of cities represented by points in 2-dimension, we need to find the possible route with the shortest distance that visits each city exactly once and returns to the start point. This classic problem has many applications in many areas, such as city planning, logistics, microchips manufacturing and DNA sequencing.

2.2.2. Overview

In the test, the test iterations were 6000 for each algorithm. To make sure the stability, for RHC, SA and GA, it ran 20 times to get the average and for MIMIC, it ran 5 times. Here we picked N = 50

Algorithm	Parameters	Evaluation Value	Time(s)
RHC	Iterations = 5000	0.112591	0.002471

SA	Iterations = 5000
	T = 1E14
	Cooling = 0.95
		0.114487	0.003904

GA	Iterations = 1000
	Population = 200, Mate = 150
	Mutate = 20
		0.152971	0.382164

MIMIC	Iterations = 1000	0.094772	16.08303

Table 2.2. Overview of Performances in TSP

Notice here that the evaluation value equaled to 1 over sum of distance so the higher evaluation values means the higher optimization solution for the problem.

In table 2.2, we could see that GA got the best evaluation value with a not bad running time. Since the similarity of the SA and RHC to this problem, both their evaluation value and times got a similar result. However, since it was a problem in 2D, so MIMIC had to use a sort to make the 2D points into vector to solve the problem.

2.2.3. Detailed Analysis for Genetic Algorithm

A route could be represented by the permutation the one route obtains when sorting the points. As the result, it was similar to the process of cross-over mating, while calculating the neighbor or the cross-over of the permutations were not trivial. The substituting neighbor process by swapping two cities might lead to a better optimization. Therefore, it was one of the reasons why GA performed better than other three.

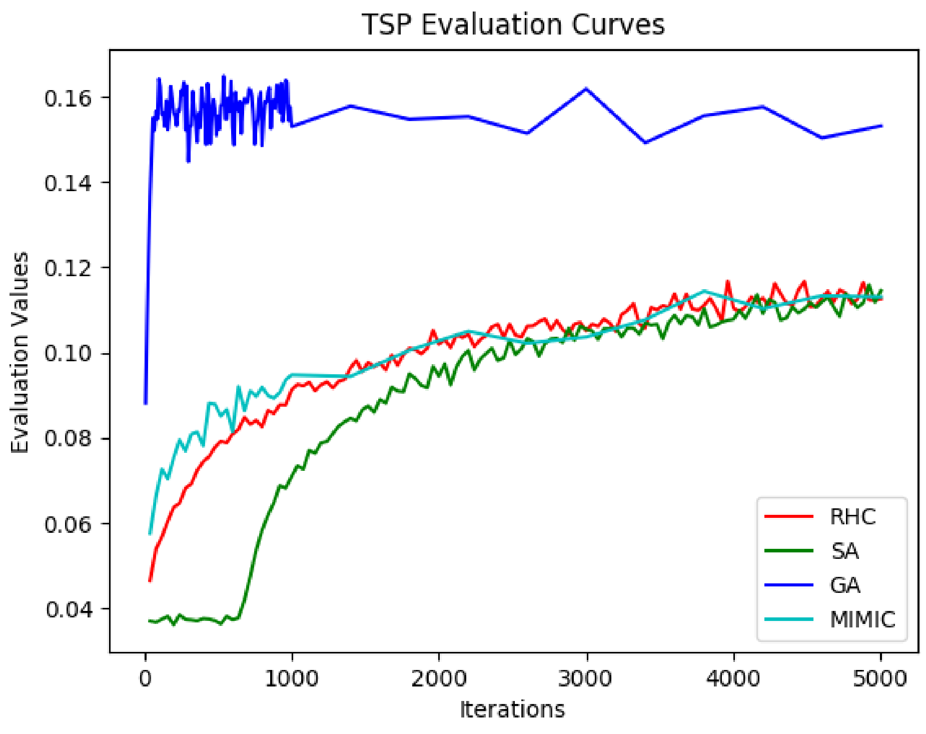

Fig 2.2. TSP Evaluation Curves

Moreover, even though TSP problem has a large space of hypothesis but each of them still have a local optimal value that is close to the best global optimum. As the result, GA could find this good value within just a few iterations just as shown in figure 2.2. GA converged much earlier than the other three algorithms, which were still looking for local optimum after 30000 iterations.

2.3. Knapsack Problem

2.3.1. Problem Description

Knapsack problem is also a classic NP-Hard optimization problem. The problem is to get the best values combination of items with a value and a weight, which makes their sum of weights is less or equal to the given backpack capacity. In the test, we used binary bit to represent whether the item was chosen or not and it was convenient for us to use the algorithms to get the max values.

2.3.2. Overview

In the test, the test iterations were 6000 for each algorithm. To make sure the stability, for RHC, SA and GA, it ran 20 times to get the average and for MIMIC, it ran 5 times. We picked items number N = 40 and the capacity of backpack with C = 3200.

Algorithm	Parameters	Evaluation Value	Time(s)
RHC	Iterations = 5000	3641.078	0.000619

SA	Iterations = 5000
	T = 100
	Cooling = 0.95
		3732.016	0.001887

GA	Iterations = 1000
	Population = 200, Mate = 150
	Mutate = 25
		3969.393	0.110255

MIMIC	Iterations = 200	4049.137	0.331412

Table 2.3. Overview of Performances in Knapsack Problem

It was a problem similar to TSP, but MIMIC could get the best performance with a fast running time thanks to the small hypothesis space. As the result, GA also could get a good evaluation performance. For RHC and SA, they could get a value close to MIMC in a very short running time.

2.3.3. Detailed Analysis for MIMIC

As the figure showed, MIMIC started to converged around 200 iterations and MIMIC obviously could get a better value given the capacity C comparing to the other three algorithms. However, due to the trade-off, even MIMIC could converge within much less iterations, its running was still the worst because we had to find the MST during each iteration to select the best option according to the information gain by previous search, which had to take a great deal of time.

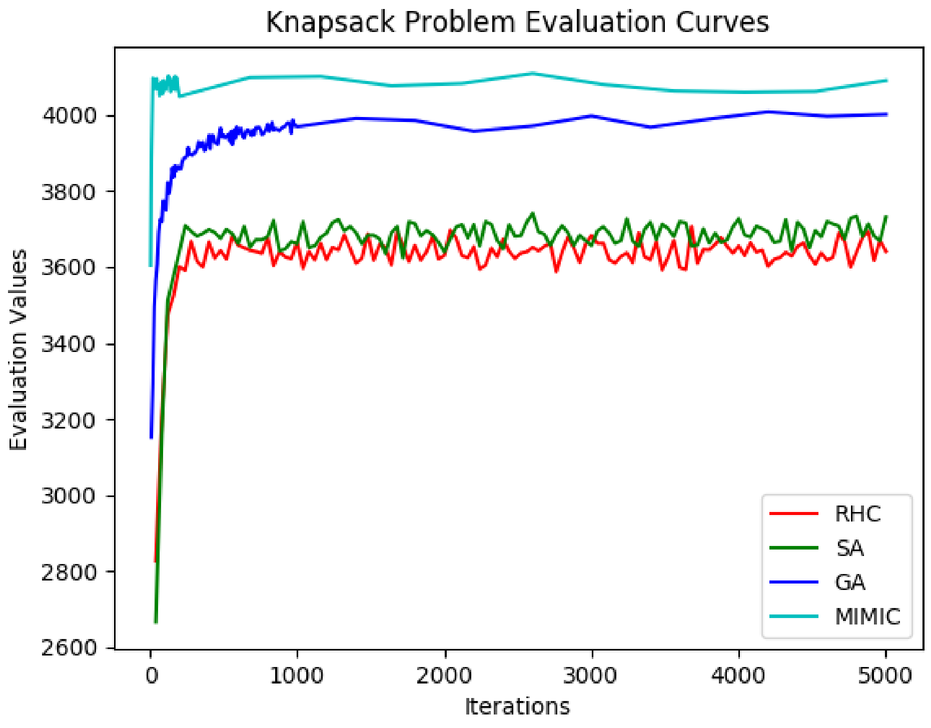

Fig 2.3. Knapsack Problem Evaluation Curves

GA also performed good in this problem (much better than RHC and SA, close to MIMIC). It was because MIMIC and GA could perform better with successive generation interpolates or extrapolates from the parents. However, MIMIC performed better than GA because MIMIC made good use of the information gain from the previous search and as the result it could select the best option.

3. Summary and Conclusions

3.1. Randomized Hill Climbing

RHC starts with a random point, then locates local optima by moving to the more optimal neighbors until it reaches local optimal peak. If it reaches the local optimum, it will start randomly in another point. At last it selects the best local optimum where it has been to as the approximate global optimum.

Due to its random process of starting point and picking points, global maximum cannot be guaranteed but it can find the local optimum very quickly. However, it can get a high score when dealing with the problem with many local optimum close to the global optimum especially it is with a good algorithm choosing a restart point.

3.2. Simulated Annealing

SA is an improved version of Hill Climbing which modifies the picking neighbor rule. It uses probability to decide whether it picks the neighbor point or not to get the global optimum and this probability is determined by temperature, which will be exhausted after several iterations. It is inspired by the metallurgy with heating metal to high temperature and cool down it slowly to increase ductility.

As the result, it ensures to reach the local optimum but because of the cooling process it will slow down (decrease the probability) when it is close to the top especially there are many noisy local optimum near the top. Moreover, high temperature can help accept to jump to the neighbor easily but low temperature reduces the probability to accept the neighbor.

3.3. Genetic Algorithm

Genetic Algorithm simulates the evolution process of natural selection in biology by mating using crossover and mutating to strengthen the related characteristic instead of instances that are not suitable to the environment, which means the less related ones are replaced by offspring of instances with better performance.

GA can find a good instance quickly by getting offspring of parents with a good performance and each iteration is very fast. But it cannot guarantee the best candidate because it has tendency to converge in local optimum. GA has difficulty to deal with the problem with a large hypothesis space which increases exponentially as attributes number increases.

3.4. MIMIC

MIMIC is based on the framework of GA. The difference is that MIMIC estimates the probability densities to try to find the optimum. MIMIC makes good use of the structure of the solution space and uses it as the key to find the optimum.

Because it uses the structure of the prior information gain in each iteration, it needn’t put too much limitation of the structure of the problem. As the result, it has a good performance for the problem that has a strong connection with the input parameters. Compared to GA, it does not need to many iterations to find the best value but since it has to find the information gain in each iteration, it costs a lot more in each iteration.