COMP4641 Assignment #2
Liu Qinhan (qliu359@gatech.edu)

Randomized Optimization Report

1.Optimization Algorithms

1.1 Randomized Hill Climbing (RHC)

An arbitrary solution is initially set and then altered incrementally, one step each time, to check if it becomes a better solution. This process is repeated until the solution converges.

1.2 Simulated Annealing (SA)

The original idea comes from the annealing process in metallurgy, where a metal is brought to a high temperature then slowly cooled to decrease defects. Based on RHC, a probability function is added to determine whether or not to accept the new point. In this way, points that may not improve the performance are adopted sometimes so that the overall algorithm is able to jump out of local maxima (or minima). The probability function usually reflects the "temperature", when the temperature is high, conceptually, the opportunity of the algorithm to jump around large distance is higher, and when it is low, the algorithm turns to converge.

1.3 Genetic Algorithm (GA)

The methodology of genetic algorithm originates from the theory of natural selection. At the beginning of each iteration of the algorithm, a list of solutions (generation) is provided. Then several processes, namely mutation, crossover, and reduction, are performed on the list to yield a better list of solutions for next iteration. The mutation alters the parameter of part of the generation to add diversity. In crossover, 2 or more parent solution are taken to somehow produce child solutions. Lastly, a new list of better solutions are chosen from the last generation and the newly generated solutions. It is conventionally called "the 2nd best option" for it works out better on a bunch of problems.

1.4 Mutual-Information-Maximizing Input Clustering (MIMIC)

It is designed to explicitly transfer information about the cost function obtained from one iteration to later iterations to capture the intervals with optima from a well-sampled input space.

2. Optimization Problems

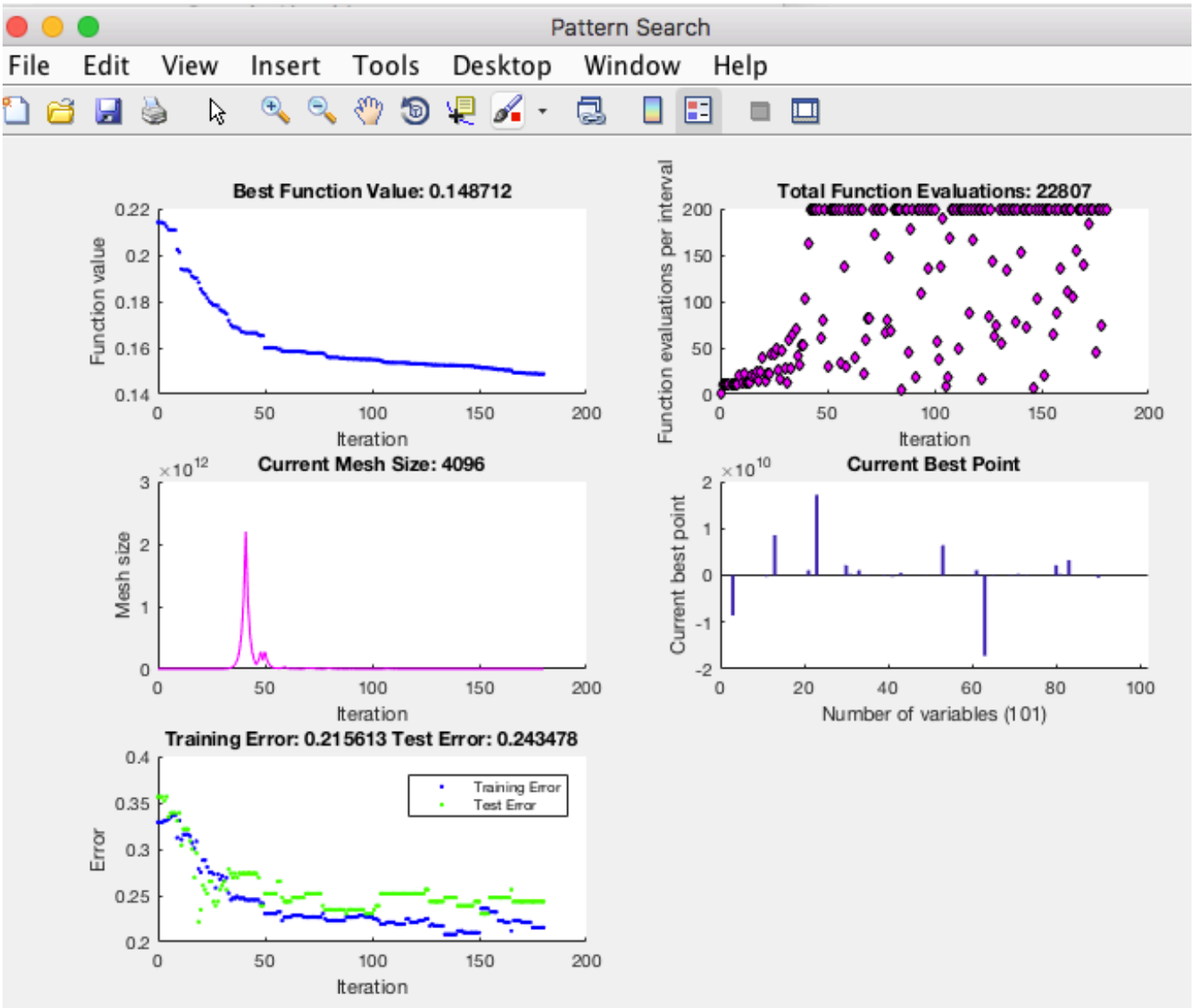
The purpose of this assignment is to understand and try out the utility of the random search algorithms above. In total of 4 optimization problems were chosen with the intention of demonstrating the strengths and weaknesses of each algorithm. The first one is finding optimal weights for a neural network, and only the first 3 algorithms are tested

on it, namely RHC, SA, and GA. The other three problems are the Four-Peaks Problem (FSP), the Traveling-Salesman-Problem (TSP), and the Flip-Flop Problem (FFP) , respectively. They are used for trying out all 4 optimization algorithms.

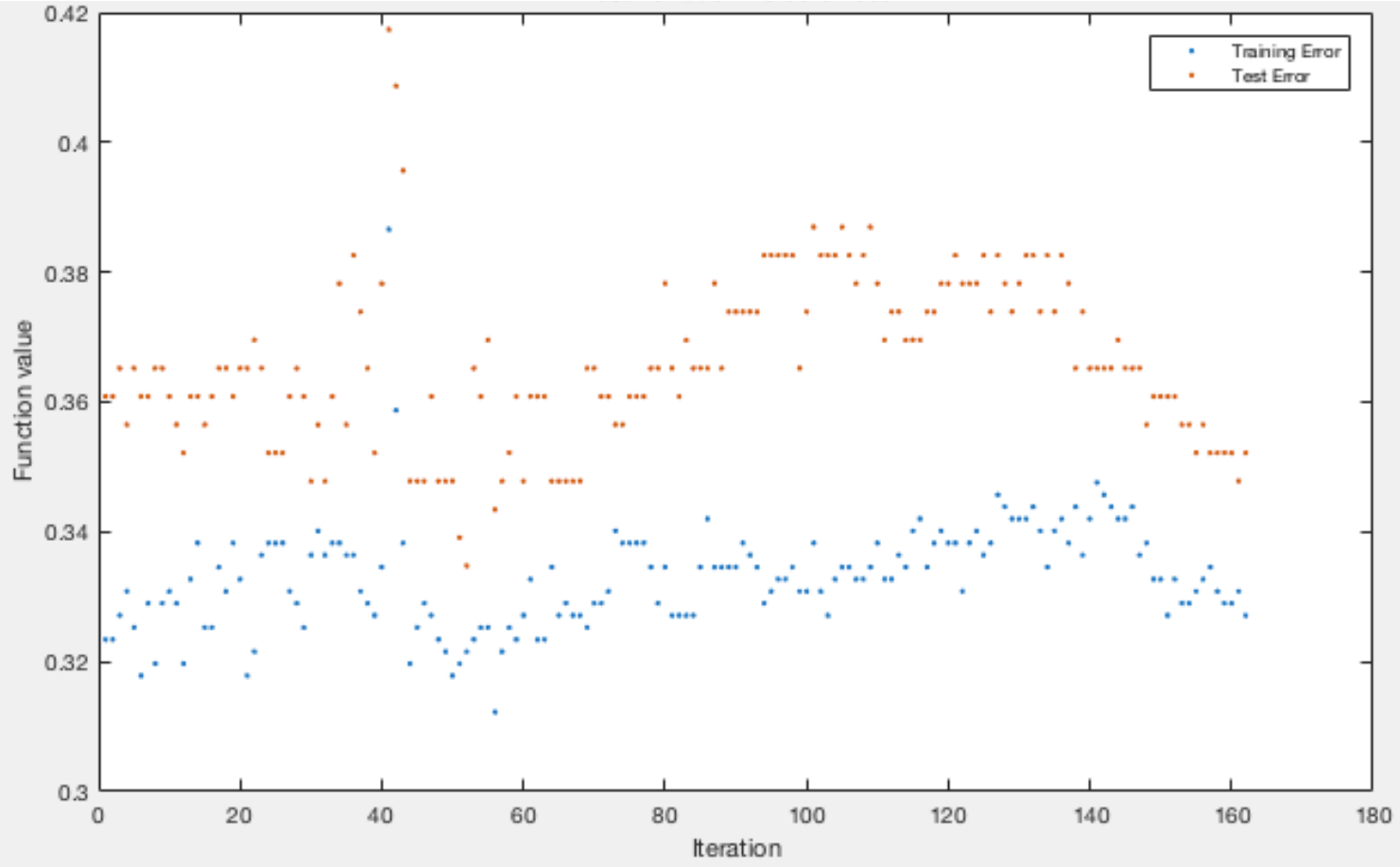
2.1 Finding optimal weights for neural network

Cancer Data This is a set of data about cancer diagnosis, that is, whether the tested cell is of malignant (bad) or benign (mild). This problem attempts to determine whether the tumor is malignant or benign from a sample of cells removed from a tumor through a minimally invasive surgical process. Such a problem is a typical one of pattern recognition.

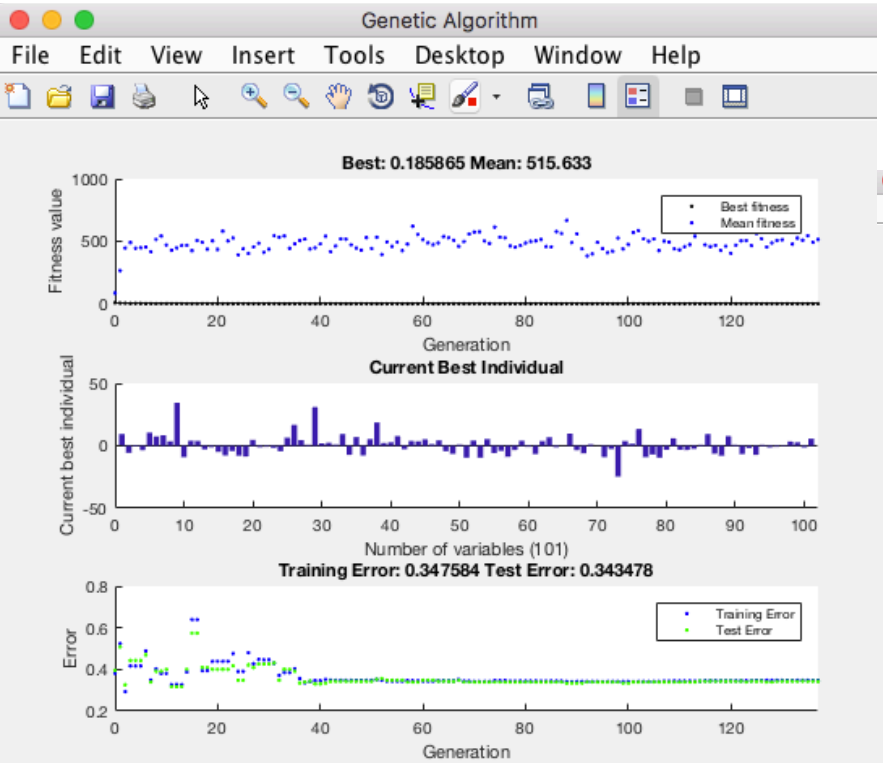
The architecture of the net is set to contain 1 hidden layer of 10 nodes. Since there are 20 attributes from the data, 10 nodes in 1 hidden layer, there are roughly 20*10+10*1 = 210 dimensions for the whole net. To initialize, all weights are randomly assigned in the beginning. As a reference, the popular method of solving the network, back propagation (BP), achieves accuracy of 95% or higher. The results of each of the algorithms in RHC, SA, and GA are shown in the following table.



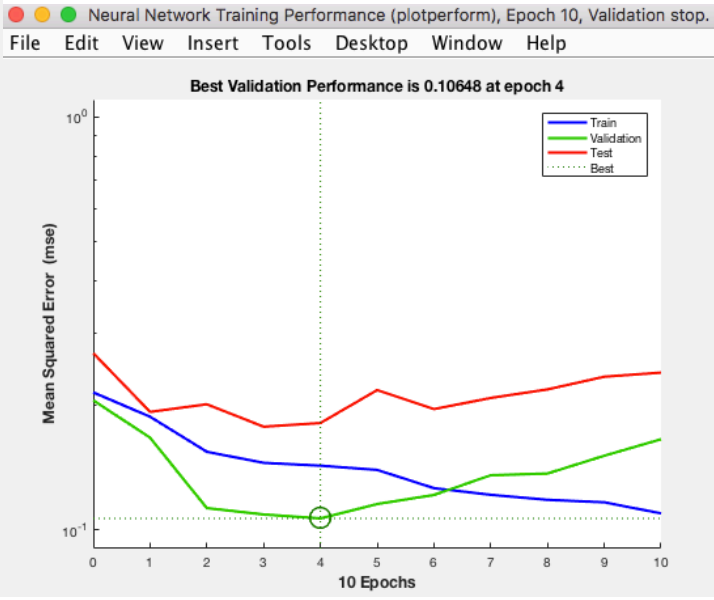
RHC


Training Error
Test Error
Function value
Iteration
0.42
0.4
0.38
0.36
0.34
0.32
0.3
0
20
40
60
80
100
120
140
160
180

SA


Genetic Algorithm
File Edit View Insert Tools Desktop Window Help
Best: 0.185865 Mean: 515.633
Best fitness
Mean fitness
Fitness value
Generation
Current Best Individual
Current best individual
Number of variables (101)
Training Error: 0.347584 Test Error: 0.343478
Training Error
Test Error
Error
Generation

GA


Neural Network Training Performance (plotperform), Epoch 10, Validation stop.
File Edit View Insert Tools Desktop Window Help
Best Validation Performance is 0.10648 at epoch 4
Train
Validation
Test
Best
Mean Squared Error (mse)
10 Epochs

BP

Algorithm	Clock Time (s)	# of Functions/ Iterations	Accuracy (%)
RHC	851.340231	22807	85.13
SA	1199.517	32421	77.77
GA	1205.7644	27600	81.43
BP	2.585	10	96

The first thing that worth noting is that none of these 3 algorithms ended up performing as well as the back propagation. This is actually expected since back propagation prevails in solving pattern recognition problems.

One reason why the optimization algorithms may not work as well is overfitting. The set of weights of a whole neural network is like a high dimension function. Known as the curse of dimensionality, to find the maxima of a function, the number of samples demanded grows exponentially as the dimension of the function increases. For such a function in this case, the training and testing data may not be enough to well generalize the problem.

Of all the optimization algorithms, RHC seems to be the best. It gets the highest accuracy amongst them and evaluates intermediate number of functions in least time. No clear shortcomings for it against others. GA comes second regarding accuracy and checks the largest number of functions in longest time. But the truth is, the accuracy still keeps improving when the programs ends in time limit. Theoretically, if the running time is enough, GA could evolve to a surprisingly perfect accuracy. The mutation and crossover enabled an unlimited potential for GA. While for SA, it is not as desired may due to the fact that the probability function may burden optimizing and force a worse step, then the efficiency of converging falls.

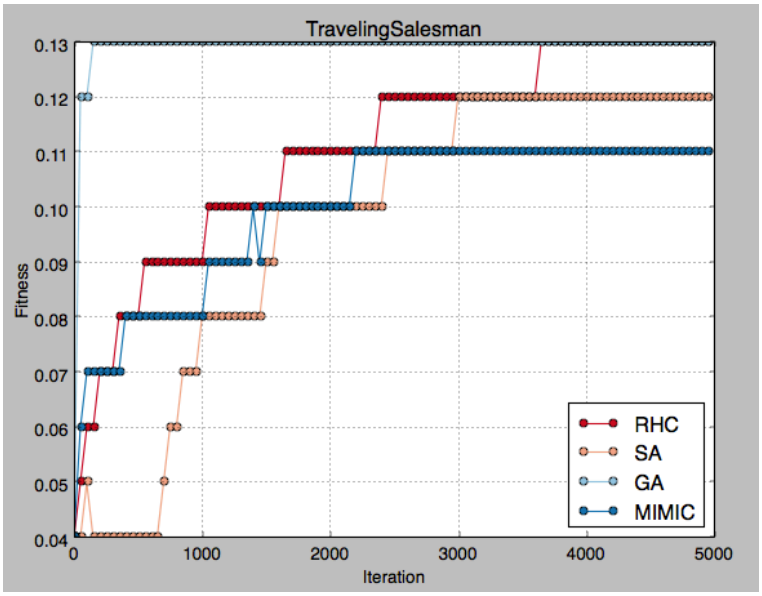
2.2 Traveling Salesman (GA)

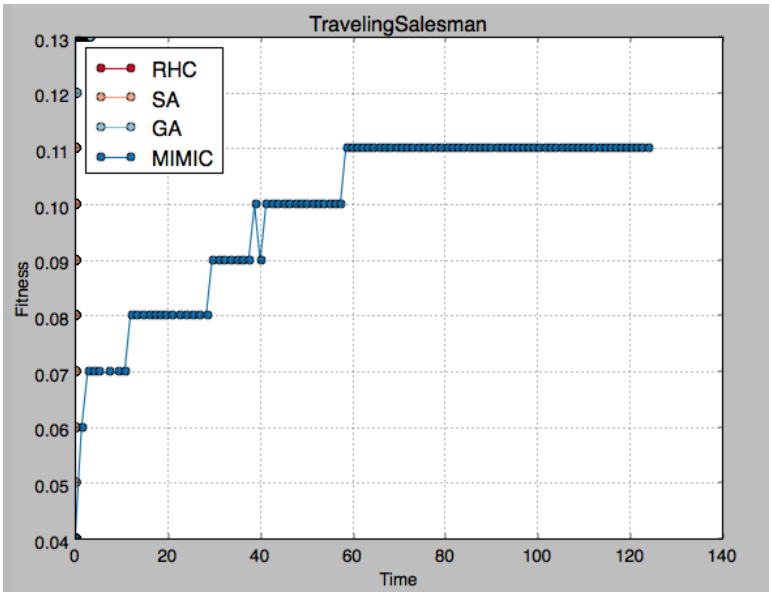
The traveling salesman problem (TSP) is a famous NP-complete combinatorial optimization problem which demands a traveling schedule to a list of cities and go back to the origin to minimize the total distances traveled. Although it is a minimization problem, it can be translated to a maximization problem easily by assigning a fitness function reversely proportional to the total distance traveled. The following result is generated by running each algorithm 5000 iterations.

We can easily make an statement that Genetic Algorithm (GA) solves the problem accurately and efficiently. This is expected since the Traveling Salesman Problem is a well-structured one for the mutation and crossover process of GA to demonstrate full utility. From the figure we know within a few iterations, it already gets the optima and it shows no shortcoming in the time consumption as well.

SA struggled a lot in the first several hundred iterations comparing to RHC! That may due to the exact issue of the forced acceptance of worse options of SA and SA ends achieving even lower fitness than RHC because of the temperature drops to a low level and forces SA to converge.

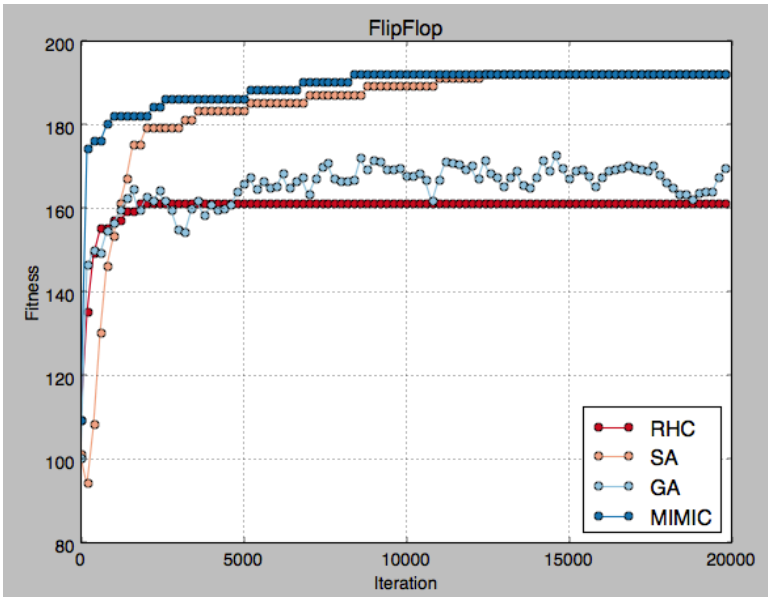
MIMIC is the worst option in this situation. It yields the least fitness solution in the longest time.

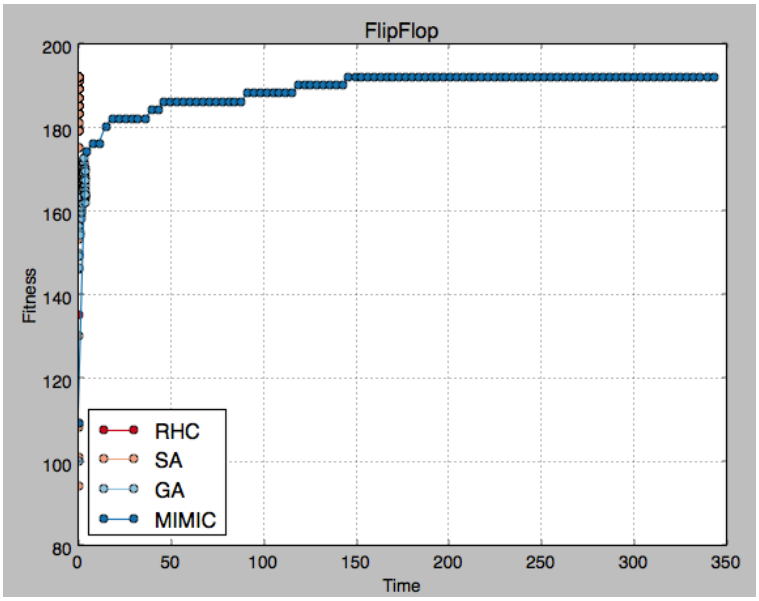




2.3 Flip Flop (SA)

Unlike the TSP, Flip Flop (FF) is relatively simpler problem. It is a fitness function measuring the number of bit alternations occurred in a bit string. A bit alternation or a "flip flop" is that the bit flips when it goes to the next bit like "01" or "10". For example, "01111" has bit alternations of 1, "01110" has 2, and "010010" has 4. Intuitively, a bis string has maximum bit alternations equal to its length and in that case, it is a alternating bit string. Again, 4 algorithms are tested on this problem for a bit string of length 200.





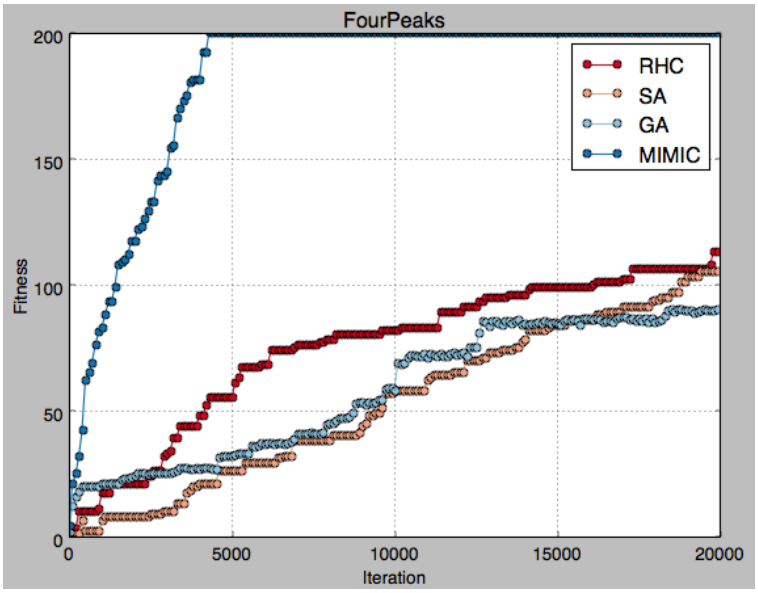
Although the MIMIC achieves the highest fitness, it uses too much more time and boosts only limited fitness comparing to SA. SA has its edge over GA and MIMIC when optimizing simple problem like this one because it runs incredibly faster. While in the other hand, RHC should not be slower than SA, but RHC has a trouble converging to a local maxima. That is exactly the case here, since after the first 1000 iterations or so, it never gets better.

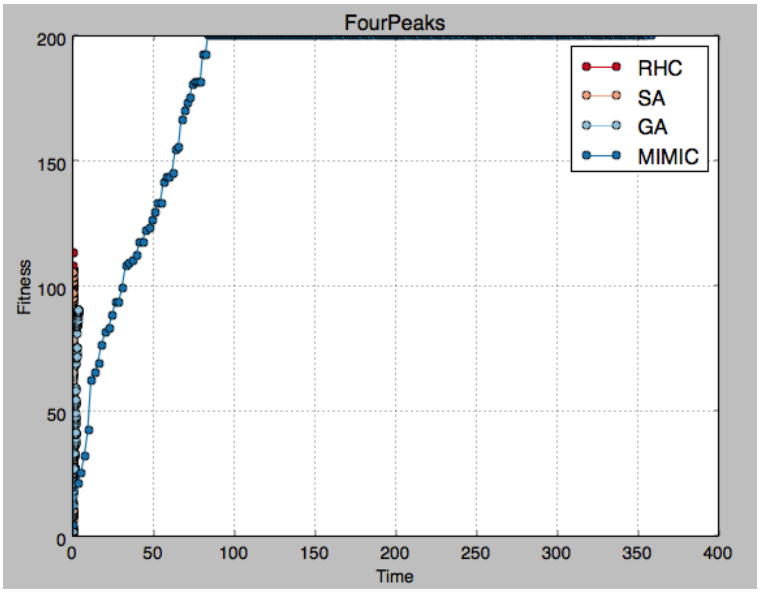
If the time is not a constraint to the problem, MIMIC can do a better job (or maybe just a fairly good one). Based on the methodology of MIMIC, it stands out when there are multiple optima so that it would not lose the interval containing a optima in the sampling. In this particular problem, it may or may not know the optimal solution, but SA does. Eventually, if the iterations goes large enough, SA would perform as least as well as MIMIC. But still, even as accurate, MIMIC costs way more time.

GA is not a preferable solution to this problem because there is no straight-forward gain in mutation and crossover and that is why the fitness of GA keeps oscillating even after several thousands of iterations. The efficiency of GA is rather low for this problem.

2.4 Four Peaks (MIMIC)

The full definition of the four peaks problem can be found in this *file*. Without loss of generality, it is a function with 2 local maxima and 2 global maxima. All 4 optimization algorithms are implemented to search for the global maxima. The following result is generated by running each algorithm 20000 iterations.





Unsurprisingly, MIMIC seems to best suit this problem. There is a pretty obvious reason why MIMIC is good. It focuses on the intervals where the potential optima lies in and keep shortening the intervals to filter out the optima. This strategy works out perfect because the problem is a continuous one and has more than one optima so the sampling would not easily miss the optima.

Other algorithms like RHC and SA suffered a lot getting to local maxima and do not reach a reasonably good score even after 20000 iterations.

GA is not preferable since there is no gain doing mutation and crossover generally. There is no guarantee that the fitness of GA keeps improving in every iteration.

Library

- Scientific computing package *numpy*
- 2D plotting library *matplotlib*
- Java machine learning library ABAGAIL
- Matlab