HW2 Report
Randomized Learning

Subject :
CS 7641 Spring 2017 - OMS

Author :
Melisande Zonta Roudes
GT account name : mzr3

March 13, 2017

1 Introduction

The aim of the assignment is the exploration of randomized optimization. It was separated in 2 parts, the first one was to apply three different optimization problems and compare the strength of each algorithm over the problems. The second part was about building a Neural Network for the diabetes dataset (one which I chose in the first assignment), the goal was to find optimal weights for the ANN.

2 Four Optimization Algorithms

Among the Randomized Optimization Algorithms which are designed to obtain global maximum for non differentiable problems, we will study Random Hill Climbing, Simulated Annealing, Genetic Algorithm and finally the MIMIC algorithm on which Charles Isbell worked.

2.1 Randomized Hill-Climbing (RHC)

As the name sounds, the aim of this method is to reach the edge of a mountain. Here, the goal is to look for the global optimum (maximum or minimum) by moving step by step in the direction of an higher neighbor. It stop when it reaches the peak. The random part of this algorithm stands in the choice of the initialization (the starting position). Indeed, we can figure out that we can often find ourselves stuck in a local optimum so the algorithm randomize the initial position and increases the probability of reaching the global optimum.

2.2 Simulated Annealing

Simulated technique is based on a probabilistic approach in order to approximate a global optimum of a given function. This method is adequated to large search space and is often used when the space is discrete. It is interesting for problems where the goal is to find an approximate global optimum rather than a precise local optimum in a fixed time. The method is based on a metallurgic analogy. Indeed, the algorithm is combination of exploration of the search space and exploitation of neighbors values. So it's a trade off between the local and global approaches. Initially, the temperatures are set to high values and a random search of new points and it proceed to the evaluation of neighbors.

2.3 Genetic Algorithm (GA)

As we heard in the lectures, Genetic Algorithms are inspired from biology such the mutation and crossover which are happening in population evolution. The starting point of this algorithm is the determination of the population of candidates solutions. The goal is to evolve step by step to a better solution over iterations by eliminations "bad" candidates and retaining the good ones. This selection is made by mutating or mating the different parts of the population. In order to evaluate the solution domain, fitness functions are used. The disadvantage of this algorithm occurs when the number of elements in mutation are large because it causes an exponential increase in the attributes area.

2.4 Mutual-Information-Maximizing Input Clustering (MIMIC)

The main difference between this algorithm and the other ones stands in the presence of a memory of previous iterations which help to build a cartography of the search space. It is again a probabilistic method since it uses probabilities densities to find optima and the structure mentioned before. The process is to keep information about the cost function from the previous iterations to the next ones of the search.

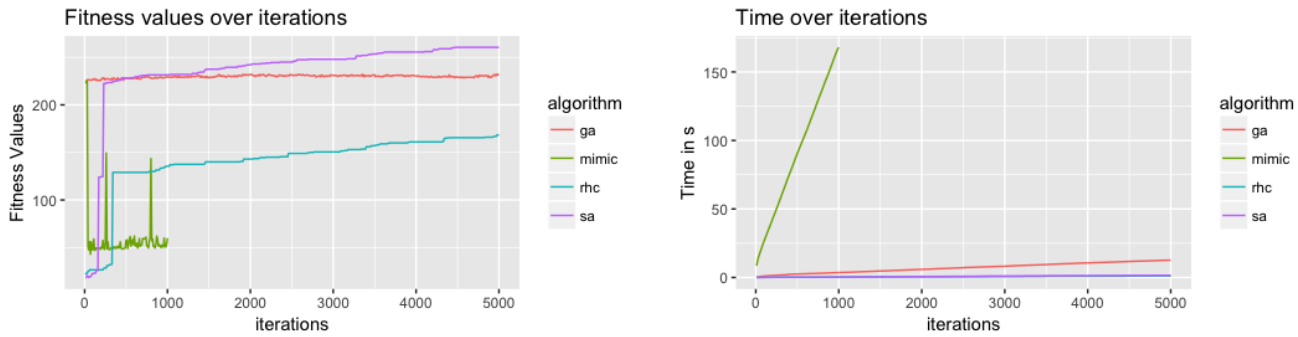
3 Three Optimization Problems

As mentionned in the introduction, the aim of these algorithms is their application over hard problems (the ones that we can't derive). In order to process the algorithms over these problems we used the library ABAGAIL in Java that we ran on Eclipse. The problems were picked in this library as well as the algorithms. The processing of the output data was done on R with the ggplot2 library adequated for data visualization. All problems were ran on 5000 iterations except MIMIC which was really slow so we restrained to 1000 iterations. In order to have a repeatable process, we ran 10 times each test and the median value was reported. The median is indeed more adequated to a statistical problem with random values.

3.1 Continuous Peaks Problems

Problematic As FourPeaks Problem, the Continuous Peaks Problem is a good problem to highlight the advantages of the different algorithms over the research of global optima instead of local peaks. Although, this problem is simple, it highlights different behaviours for the algorithms. Indeed, the main issue in problem such as hill climbing is that it is easy to find a local optimum but it is far more complicated to find the global one.

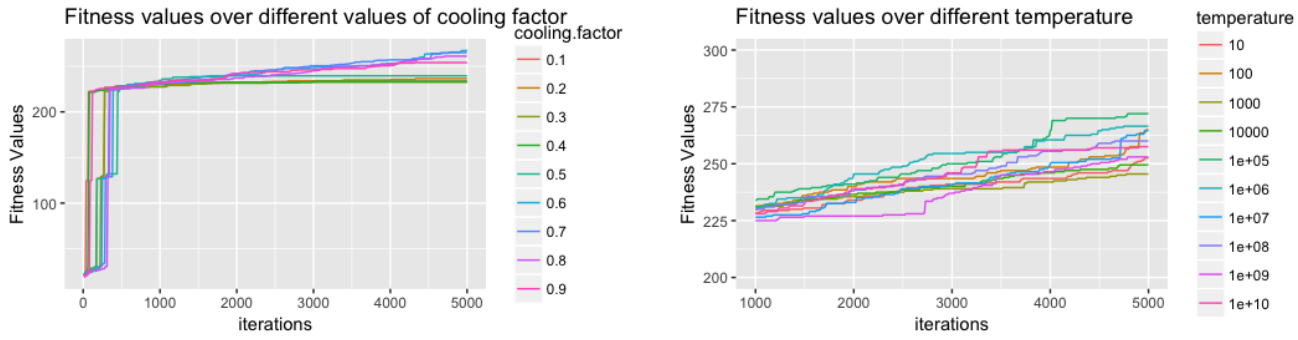
Results In order to see the behaviour of each algorithm on this problem, a computation of the values of the fitness function over iterations and the associated time was made.



a b

Figure 1: Continuous Peaks Problem
a. Fitness Values. *b.* Time

The curve 1a. shows that Simulated Annealing globally performs better than the other algorithms. However, if we look closer, we can see that for the 500 first iterations, Genetic Algorithms is above. MIMIC and RHC didn't act well. As it seems logical for RHC to perform badly since this is typically the type of problem were RHC can get stuck in a local optimum, it is surprising that MIMIC does not stand out from the crowd since it allows usually to map the search space which would be efficient in that problem. SA and GA performs better thanks to the introduction of probabilities in the resolution and the modification of elimination of elements that does not fit the function. The curve 1b. demonstrates the computation time of MIMIC which keeps increasing over iterations in comparison to really small computation times for the others algorithms even if GA is higher than the two others.



a b

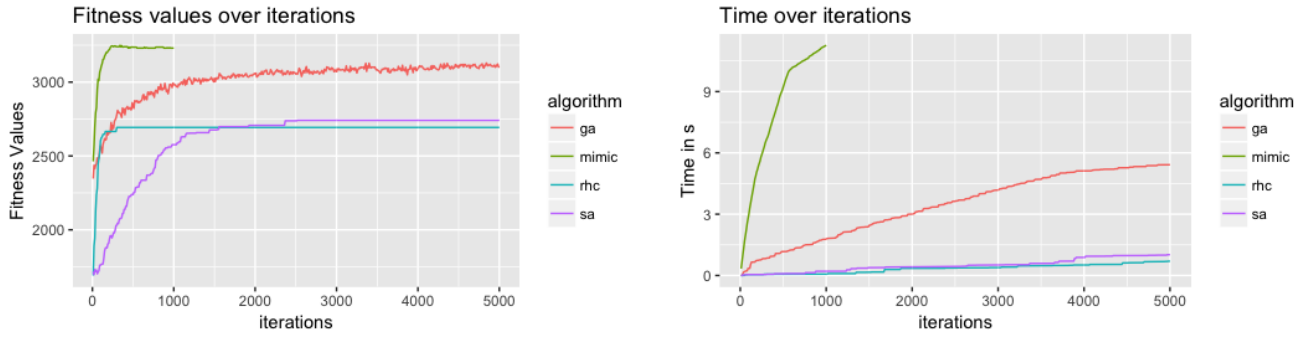
Figure 2: Continuous Peaks Problem
a. Variation over cooling factor. *b.* Variation over temperature

As we spoke about in the introduction to the SA algorithm, the parameters are cooling factor and the initial temperature. In order to tune this algorithm, the determination of the best combination of values was done. The first variation was made on temperature, the graph 2b. shows that the ideal temperatures are around 10^5 and 10^6 and the graph 2a. creates two groups, the first one gathering the values of cooling exponent between 0.6 and 0.9. Indeed, the more the cooling exponent is high the better performs the fitness function.

3.2 Knapsack Problem

Problematic The Knapsack problem has a problematic of combinatorial optimization. The background is globally to consider a number of element, each possesing a mass and a value. The aim of the method is to determine the number of each type to put in the collection so that the total weight is less or equal to a threshold but that at the same time the total value must be as large as possible. In our case the aim is to find the balance necessary to answer this problem.

Results The number of items here is fixed to 40, the copies of each at 4 and the maximum possible volume of the sack was set at 3200. As before, we want to compare the performances of the algorithms on this problem

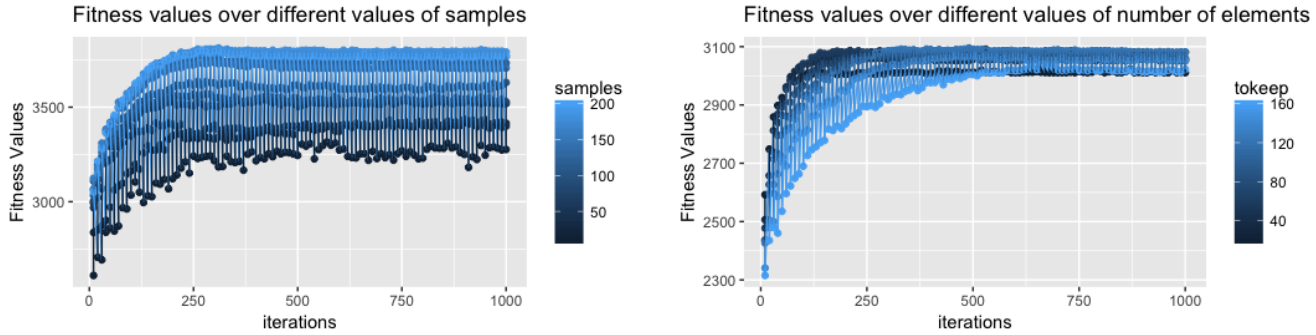


a b

Figure 3: Knapsack Problem
a. Fitness Values. *b.* Time

As the graph 3a. shows that the MIMIC algorithm performs well better on this problem than

RHC and SA which has low fitness values even for 5000 iterations. It demonstrates the efficiency of this algorithm over others which seems natural since it is adequated to combinatorial problem with the determination of the memorizing of previous iterations. The time of computation is as before well more important for MIMIC than others.



a b

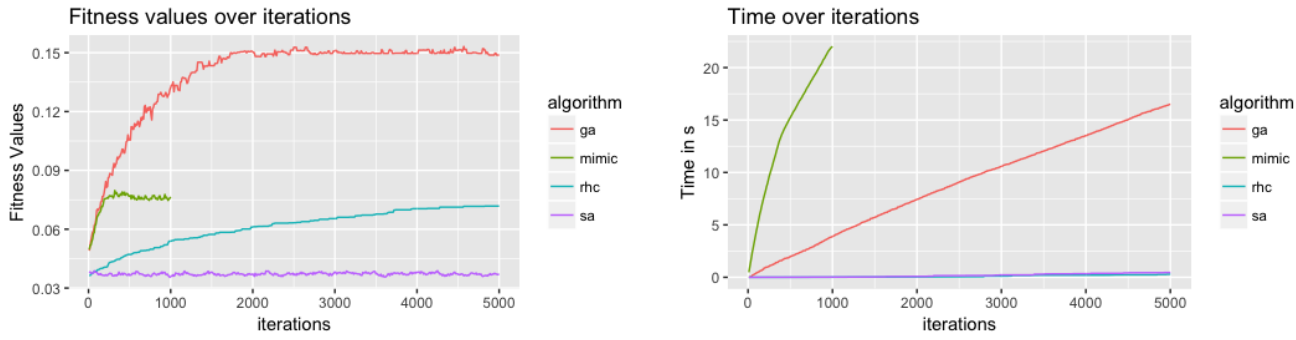
Figure 4: Knapsack Problem
a. Fitness Values. *b.* Time

In order to obtain better performances on the MIMIC algorithm, a tuning of the two parameters was done. So the number of samples was increased as well as the number of elements to keep. We can observe that the optimal value for the size of samples is the maximum one which in this case 200 but we could have plotted more values and the bigger the number of samples is the better it is. After having fixed the number of samples to that value, we tried to determined the number of elements to keep. as we can see the best value is around 120 elements for 200 samples which is logical since keeping to much elements is bad in the algorithm and doesn't solve the problem.

3.3 Traveling Salesman Problem

Problematic Traveling Salesman Problem belongs to the class of NP-hard problem which means that they can't be resolved in a polynomial computation time and as the number of points increases as it becomes impossible to solve the problem. Here the goal of this problem is to find the shortest round trip between N cities by passing through the city just once. In this problem the main constraint is not the direction but just the unique cross over points.

Results The implementation of the problem produces the effect that large reciprocal of distance on average has smaller distance calculation and leads to a better solution. Again we want to compare the reaction of our 4 algorithms on this problem.



a b

Figure 5: Traveling Salesman Problem
a. Fitness Values. *b.* Time

The graph 5a. demonstrates the advantages of genetic algorithm to find the optimal solution with mutations over generations. MIMIC is also not bad. The difference between the previous problem and this one is the dimension (here in 2D).Moreover here the order of visits matters whereas in the knapsack problem it was not a constraint. MIMIC and GA use probabilities, MIMIC try to develop a structure of the data, how they relate. In order to figure out the effect of the variation of the mutation, and crossover, we mapped the results, but no matter the values, we found no differences on the fitness values so this won't be presented in this report.

4 Neural Network with Optimization Algorithms

4.1 Dataset

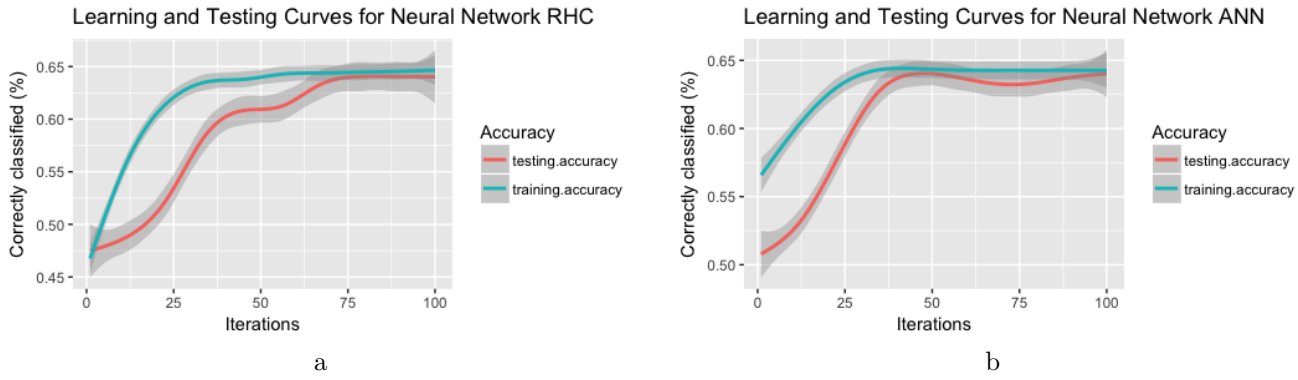
In this part, we will evaluate the optimal weights of the neural network for the Pima Diabetes data set we used int he first assignment. As a recall, the output of this dataset is the diagnosis of diabetes, it is a binary one. It has 8 attributes helping to give a label to the instances.

4.2 Algorithm

Again ABAGAIL was used and by assembling the classes, I came out with a code (inspiring muself from Abalon$_t$*est.java*)*allowingmetodeterminethetrainingandtestingcurvesaswellasthetrainingtimeandalsothesun*

4.2.1 Randomized Hill Climbing

RHC works on determining the direction allowing to find the global optimum by randomly making new start in order not to get stuck in a local peak. As Neural Networks are based on gradient descent methods, it should work well. We can visualize on the graph 6a. below that the testing curve is converging towards the learning curve which show the efficiency of this method. We reach at the 100th iteration 65% which is not bad for this dataset as demonstrated in the first assignment.

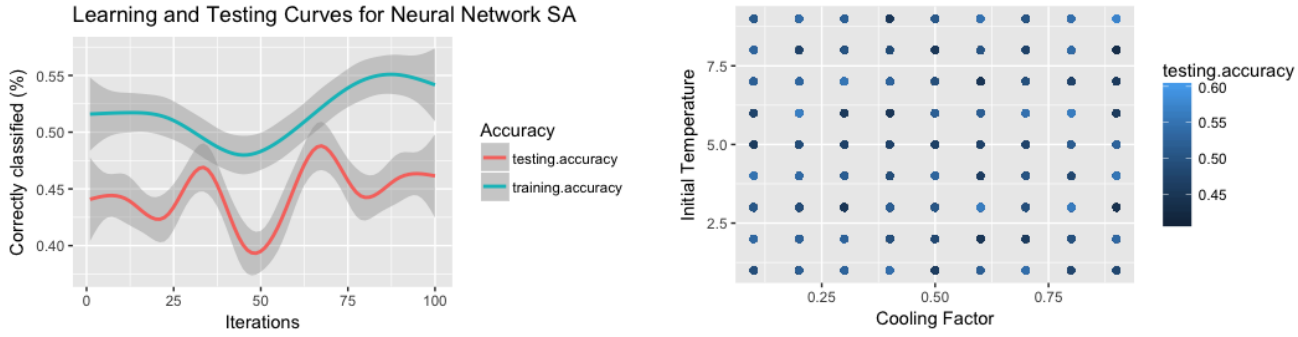


a b

Figure 6: Learning and Testing curves
a. RHC. *b.* ANN

4.2.2 Simulated Annealing

SA demonstrates a weird behaviour concerning the testing curve. The training curve has an increasing tendency, so we can see a converging behaviour whereas the testing curve has an oscillatory behaviour which can be explained by the randomized behaviour of our algorithms. Some overfitting could be identify by the decreasing trend but new runs could provide better results.



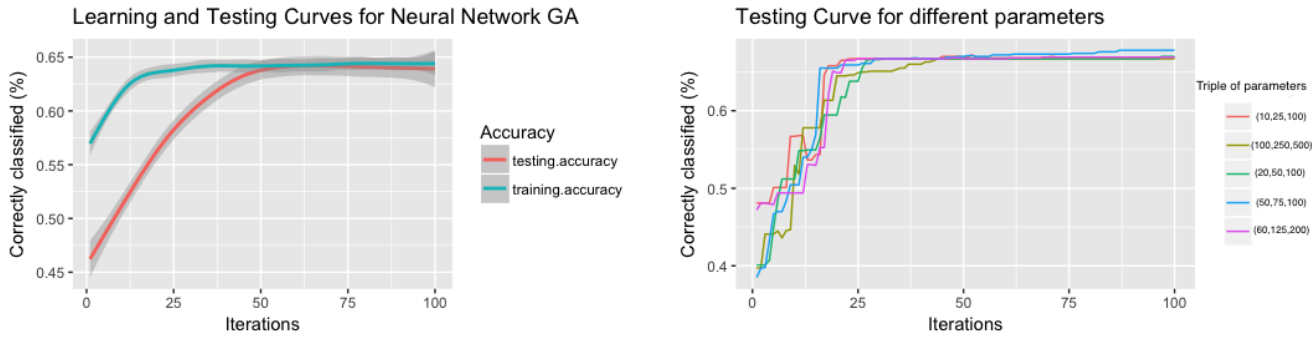
a b

Figure 7: Simulated Annealing
*a.*Learning and Testing Curves. *b.* Map over the two parameters of SA

The inquiring testing accuracy leads to the tuning of the parameters which are for the simulated annealing, the initial temperature and the cooling factor. The map 7b. shows no consistency in the choice of these parameters since no trend can be detected but for one launch best parameters can be identified and the final comparison will be based on this choice.

4.2.3 Genetic Algorithm

The learning and testing curves provide good results since we can observe a converging behaviour and a good testing accuracy on the 100th iteration. In order, to obtain better results, we tune the respective parameters of Genetic Algorithm. Indeed, the population size which corresponds to the surviving element of the fittest, toMate which is the crossover (mating between individuals) and toMutate which is the mutation, were tuned to find the ideal combination.



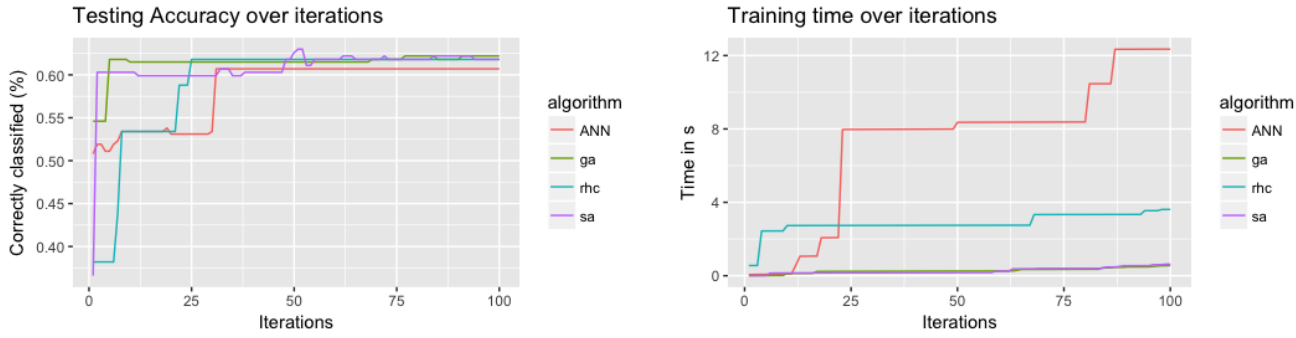
a b

Figure 8: Simulated Annealing
*a.*Learning and Testing Curves. *b.* Tuning of the parameters of GA

As the map 8b. shows the best triple is a population size of 100, a toMate rate of 75 and a toMutate rate 50. But it seems that all combinations provide the same result so we can't really conclude.

4.3 Comparison

Finally, the most important observation has to be made accordingly to the comparison between the testing curves over the algorithms.



a b

Figure 9: Comparison for Neural Network over the four algorithms
*a.*Testing Accuracy. *b.* Training Time

The graph 9a. was obtained by picking the best run. We can see that they all perform well with a slight maximum for Simulated Annealing. As the comparison was done also with the backpropagation neural network, it helps us compare randomized optimization methods with supervised learning. We can observe that they are quite equivalent. But during the experimenting we saw that whereas the performances of ANN were constant, the other ones were always changing. The results of graph 9b. were quite expected, since ANN has the highest computation time, rhc is the second one, SA and GA are combined which is unusual since GA is more costly than SA.

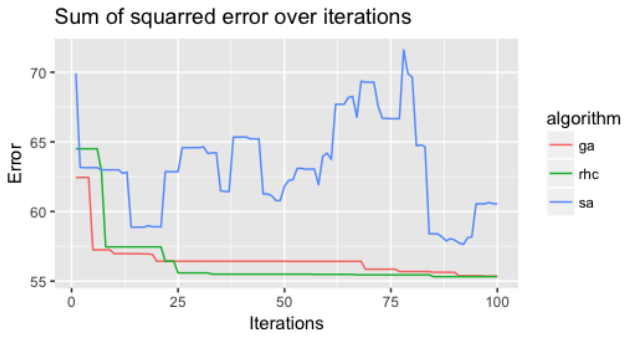


Figure 10: Comparison of sum squarred error between the 3 algorithms

The graph 10 shows the evolution of the sum of squarred error which is another error measure but which is not really adequated to our binary output. But it is informative of the decreasing trend og both RHC and SA and the totally random behaviour of SA.

The conclusion of these 3 graphs is that although SA is provide results around 63%, it is too unstable to use it, more runs would be useful and then the step of selection of the best one is necessary. But the three algorithms perform better than the backpropagations algorithm with learning rate.

5 Conclusion

This assignment was focused on radomized optimization algorithms while staying connected to supervised learning through the backpropagation neural network. The first part helped to understand the properties of each algorithm as it highlighted different advantages along the optimization problems. It also strikes me that all was about tuning the different parameters in order to obtain good results without guaranty of success, since we have to select ourselves the best run.

The second part was the link between the two assignments. It allows to compare the performances of the 3 neural net along GA, SA and RHC with the ANN. And the comparison is interesting since along runs sometimes ANN is better.