HW2: Randomized Optimization

Kunal Agarwal

Objective:

In this assignment, the objective is to compare the behavior and functioning of different randomized optimization algorithms namely, Randomized Hill Climbing (RHC), Simulated Annealing (SA) and Genetic Algorithms (GA).

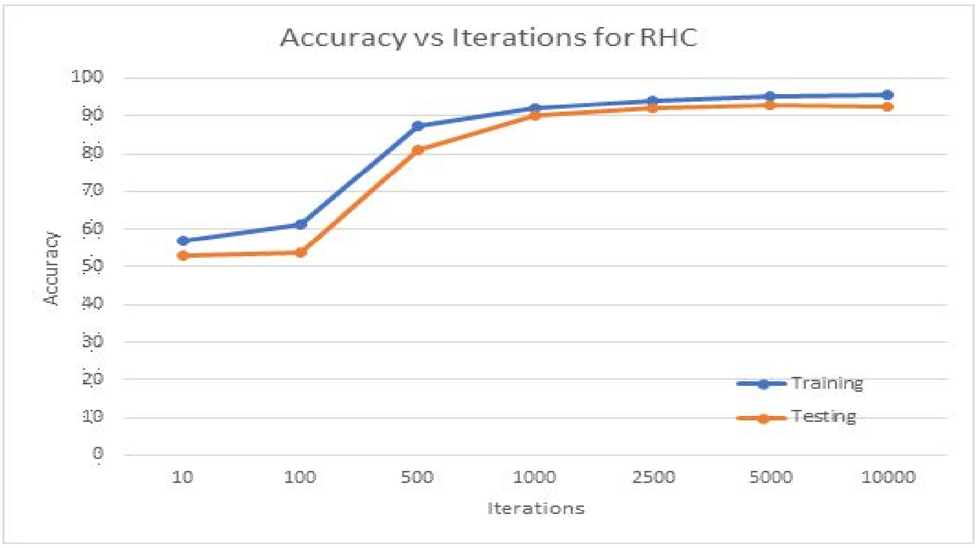
The assignment has two sections. In the first part, we aim to find optimal weights for neural network using different randomized optimization algorithms and compare the performance to traditional backpropagation algorithm. In second part, an analysis has been done on two optimization problems to highlight the differences between the mentioned random search problems, effects of presence of local minima/maxima and how these algorithms perform against them. ABAGAIL library has been used for the implementation of the assignment.

Neural Network:

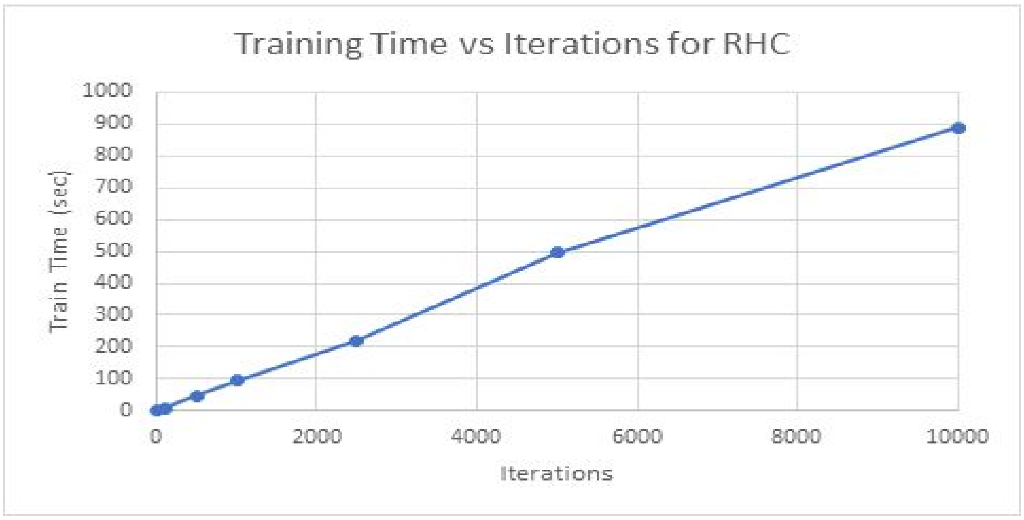
For accomplishing the first section, out of two datasets used in previous assignment analysis has been performed on phishing dataset. The data has been split into training (80%) and testing (20%) set. Please note in the last assignment, I got 96.5% testing accuracy using a backpropagation neural network with 5 hidden layers with 100 nodes. For the purpose of this assignment the neural network structure has been changed to 50 hidden layer size. The analysis of finding optimal weights for neural network using RHC, SA and GA has been detailed below.

Randomized Hill Climbing (RHC):

RHC algorithm tries to search for parameters with low error and proceeds in the direction of increasing accuracy till a local optimum solution is reached. The randomness causes the algorithm to give different solutions on running multiple times. Therefore, I ran RHC multiple times and results were averaged to negate the effect of randomness to some extent. For RHC, in ABAGAIL, there were no hyperparameters to vary. As we can see from the Accuracy vs Iterations graph below and as expected, the training and testing accuracy increases on increasing number of iterations. In the starting, iterations are low to reach a optimal solution. A horizontal line means complete convergence and it can be seen that algorithm starts to converge around 1000 iterations and at 5000 iterations, maximum testing accuracy (92.7%) is reached. The final accuracy is four percent lower than the backpropagation along with higher training time. Higher training time is due to high number of iterations for RHC compared to backpropagation 150 iterations. The reason for this can be attributed to learning rate in gradient descent which can help reach the optima faster whereas RHC does not have anything to finish the search faster.



The below plot for training time shows, RHC algorithm train time is very fast and as expected the run time increases with the increase in number of iterations. The algorithm completes 5000 iterations in 500 seconds.



Simulated Annealing (SA):

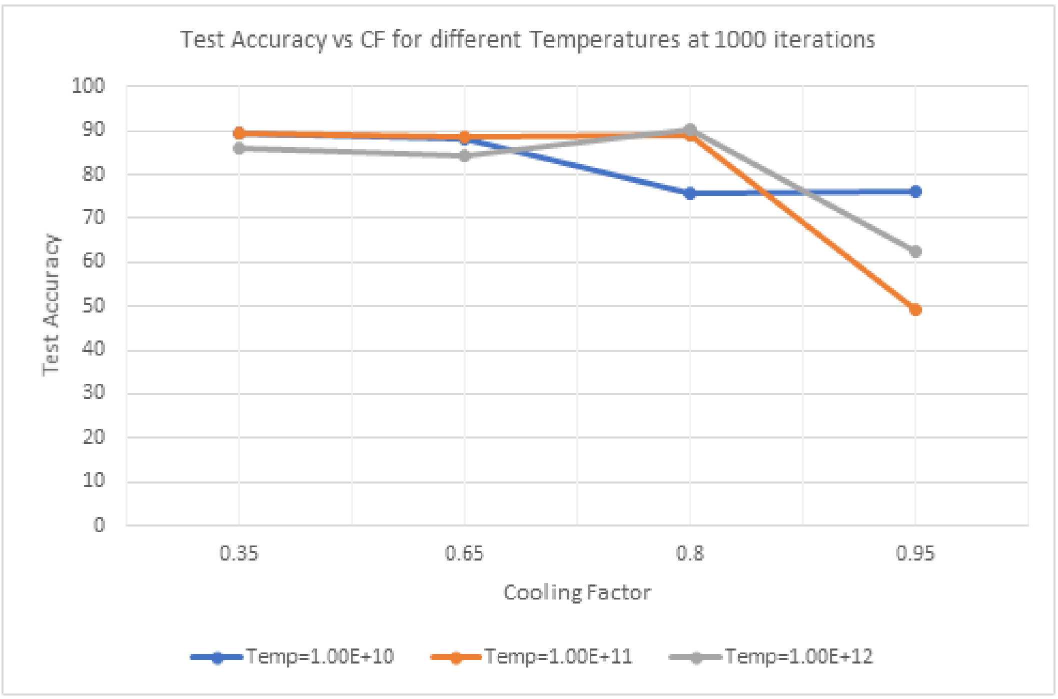
Simulated Annealing is somewhat similar to random hill climbing but here we have a probability of accepting a solution that is worse than the previous one (that is not the case in RHC). The idea of SA is to allow the algorithm to "explore" some paths even if they are in principle not very good . It helps in avoiding getting stuck at local optima.

Parameter Tuning:
There are two parameters which can be tuned in Simulated Annealing algorithm.

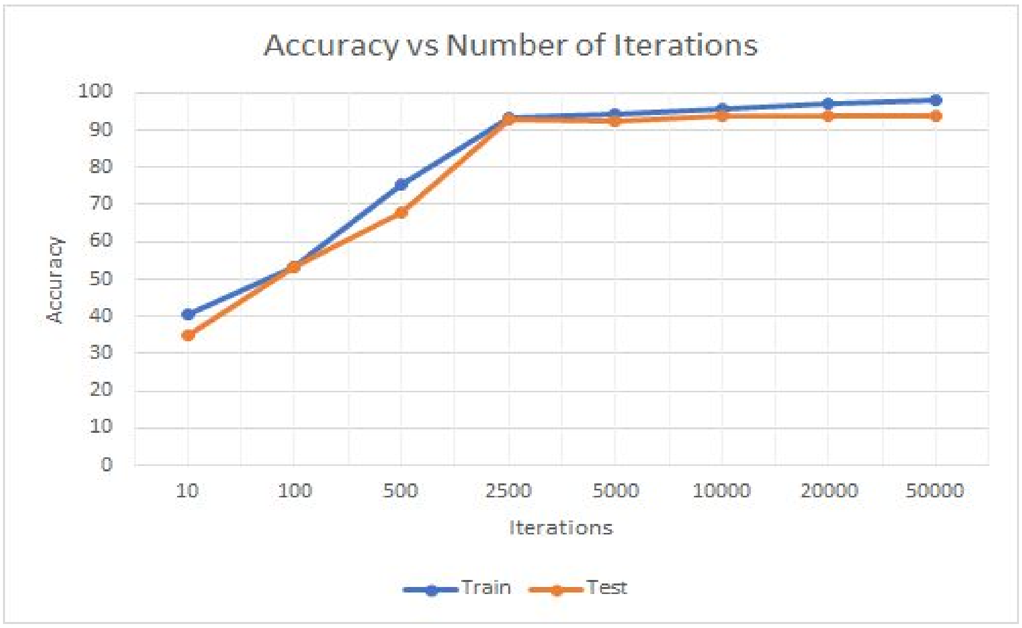
Cooling Rate: The cooling rate is the rate at which the temperature is cooled. Choosing optimum cooling rate is essential because if the cooling rate is too high we won't explore enough and the solution might be a bad one but if the cooling rate is too low then we would never decide on a path and would keep oscillating between different solutions without really optimizing.

Temperature: The second parameter which can be tuned is starting temperature. Temperature affects the rate of exploration. If the starting temperature is too high the algorithm will keep on exploring and vice-versa in case of very low starting temperature.

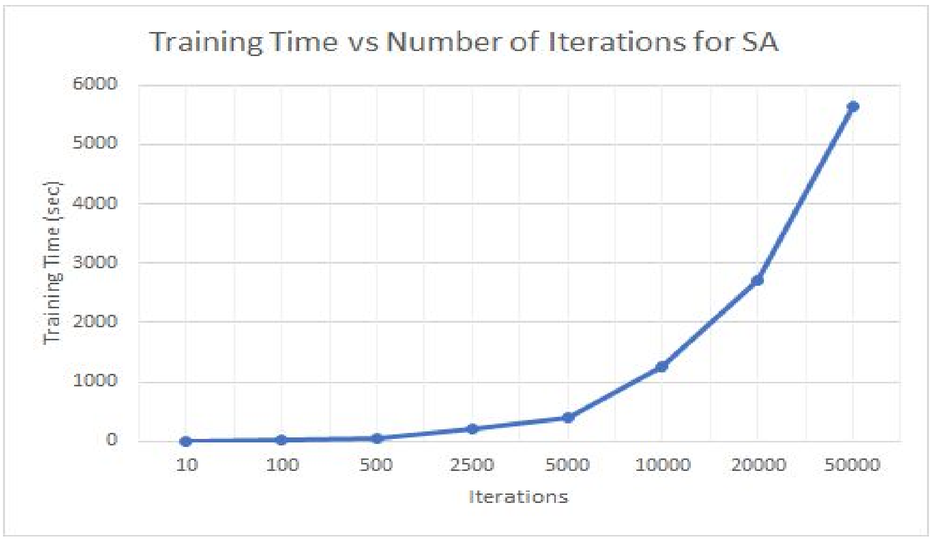
Different combination of cooling rate (0.35, 0.65, 0.8 and 0.95) and starting temperature (1E10,1E11 and 1E12) for the analysis. Testing accuracy was tested keeping 1000 iterations as constant for all different combinations of parameters. It can be seen from graph below, choosing very high cooling rate (0.95) leads to lowest accuracies in each case because of very less exploration. For other cooling rates and temperature combination, accuracy values are quite similar. Based on highest test accuracy (90.45%), cooling rate of 0.85 and starting temperature of 1E12 were chosen as optimal parameters.



After selecting optimal parameter values, training and testing accuracy was tested with different number of iterations. The accuracy vs Number of iterations graph below shows increase in training accuracy continuously but testing accuracy drops slightly at 5000 iterations and then rises again to reach maximum at 20000 iterations. At 50,000 iterations training accuracy is still increasing but testing accuracy drops indicating overfitting. The algorithm starts to converge around 1000 iterations and maximum test accuracy (93.98%) is reached at 20,000 iterations. The final accuracy is 2.5% lower than backpropagation algorithm.



The training time vs number of iterations graph for SA is similar to RHC and can be seen that SA is also quite fast. The training time increases with increasing the number of iterations.



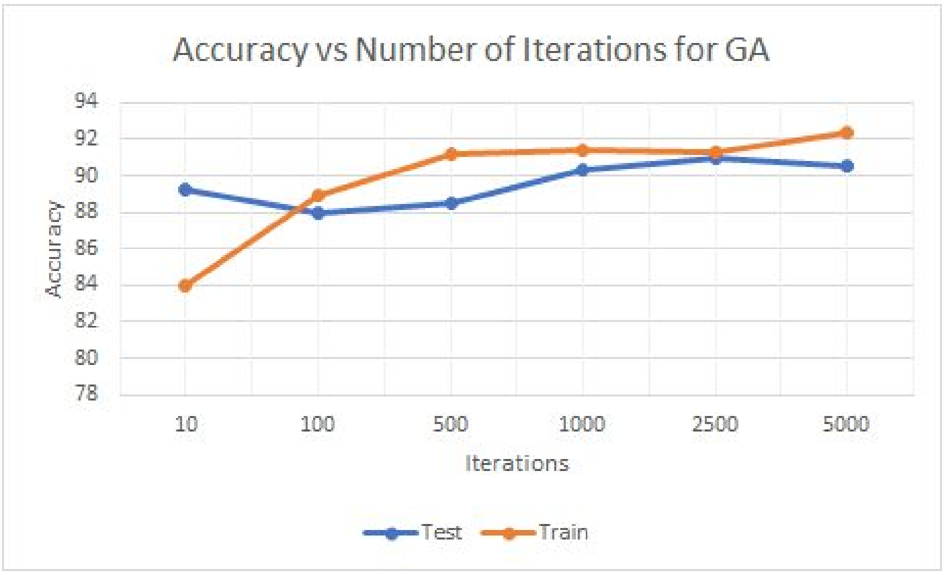
Genetic Algorithm (GA):
The final randomized algorithm we analyze in this section is Genetic Algorithm which is inspired by the concept of biological natural selection.

Parameter Tuning: Here, we have three parameters to tune – the population Size, portion of the population to mate (toMate) and the portion of the population which has to be mutated (toMutate) as mentioned in ABAGAIL. I varied toMate and toMutate parameters keeping population size constant at 100. I also kept iterations number constant at 100 for parameter tuning. The results have been tabulated below:

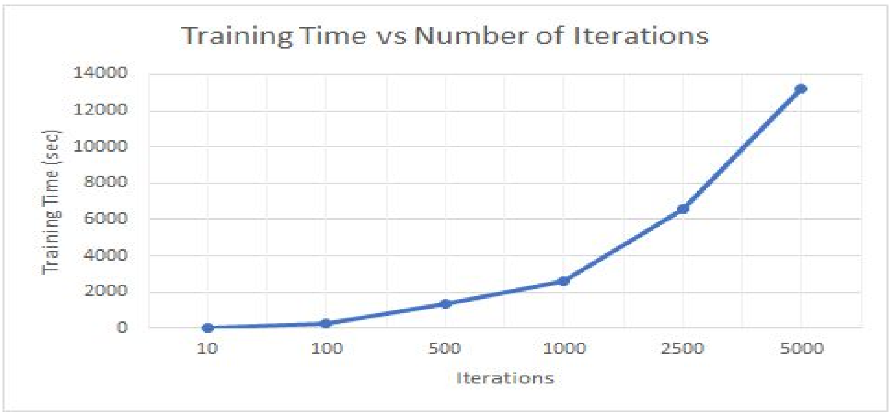
toMate	toMutate	Testing Accuracy	Training Time (sec)
30	25	86.08 %	169.60
30	50	89.02 %	213.22
50	25	89.00 %	228.16
50	50	89.65 %	261.99

The testing accuracies were almost similar but based on highest accuracy I chose 50:50 to be the optimal value for all further analysis.

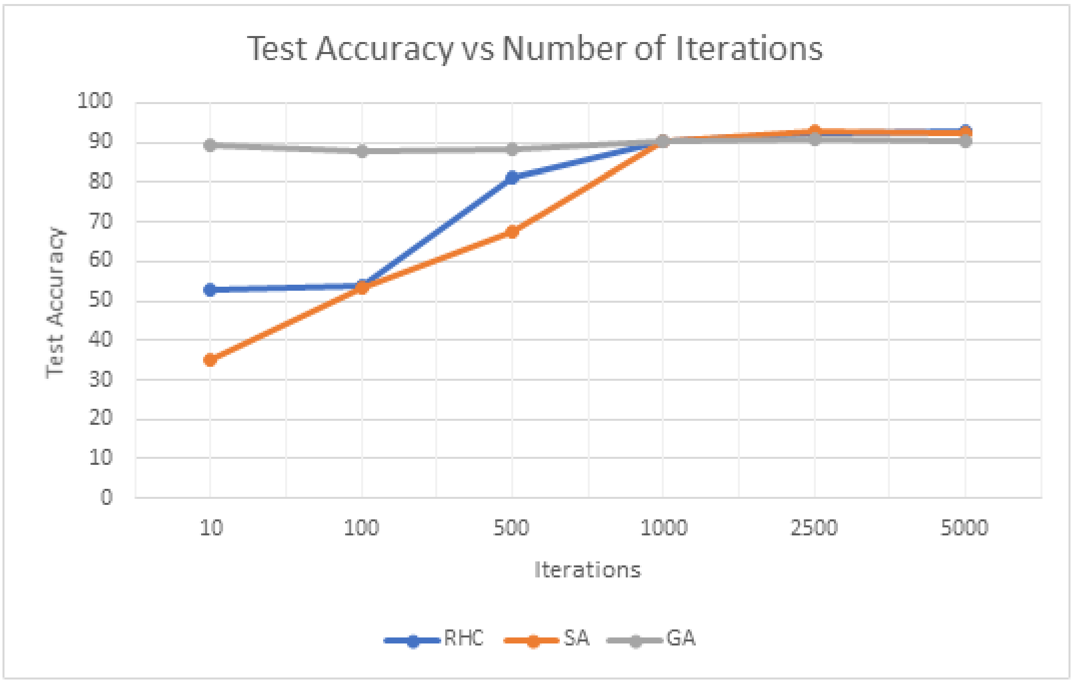
After selecting the optimum parameter values, accuracy was compared based on number of iterations. We see Genetic algorithm quickly reaching near highest accuracy. For 10 iterations it gives 89% testing accuracy. This is in sharp contrast to SA and RHC which doesn't reach high accuracy so quickly. This can be attributed to high crossover and mutation size selected.



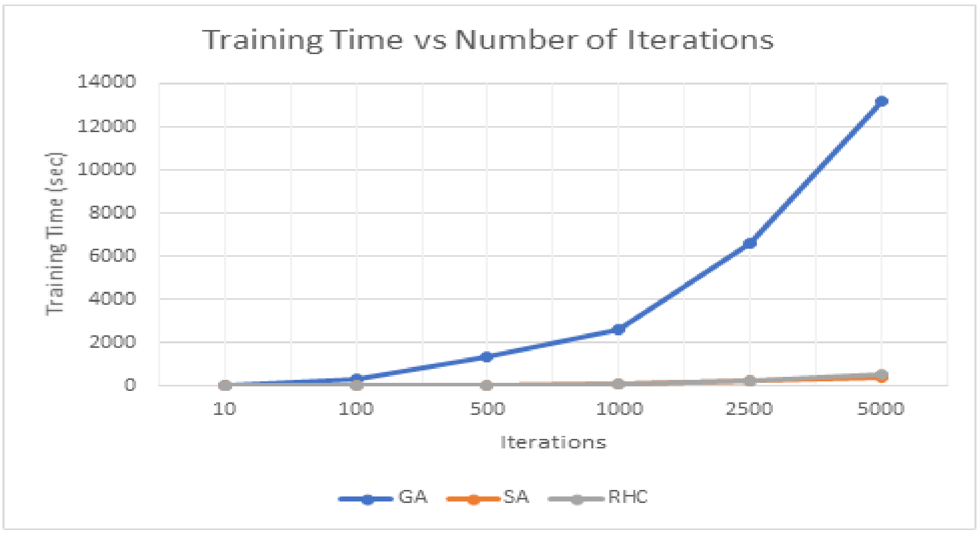
Maximum accuracy (91%) is reached at 2500 iterations and decreases at 5000 iterations. The final accuracy is 5.5% lower than backpropagation. GA takes a lot of time to train even for small iterations compared to RHC and SA.



Comparative Analysis:
Comparative analysis of different RHC, SA and GA with hypertuned parameters is shown below:



As evident from the Test accuracy vs number of iterations graph above, RHC and SA perform very similarly. After 1000 iterations, the lines for both are almost overlapping. GA converges very quickly to a high accuracy, it gives 89% accuracy at 10 iterations. After that it fluctuates a little and reaches maximum accuracy (91%) at 2500 iterations but the downside of GA is that it takes a lot a time for each iteration as shown by the Train time graph below. Therefore, for our dataset SA and RHC seems to be the ideal choice.



On comparison with backpropagation, the randomized optimization algorithms come close to the backpropagation accuracy but they take a lot of time compared to backpropagation. Difference in time between backprop and other algorithms can be attributed to different hidden layer sizes which was just 5 in case of backprop whereas it was 50 in other algorithms. Test accuracy for randomized algorithms doesn't change much after 2500 iterations.

	Backpropagation	**RHC**	**SA**	**GA**
Testing Accuracy	96.5%	92.7% at 5000 iterations	93.98% at 20000 iterations	91% at 2500 iterations
Training Time (sec)	3.8286	495.6	2702.92	6617.43

Optimization Problems:

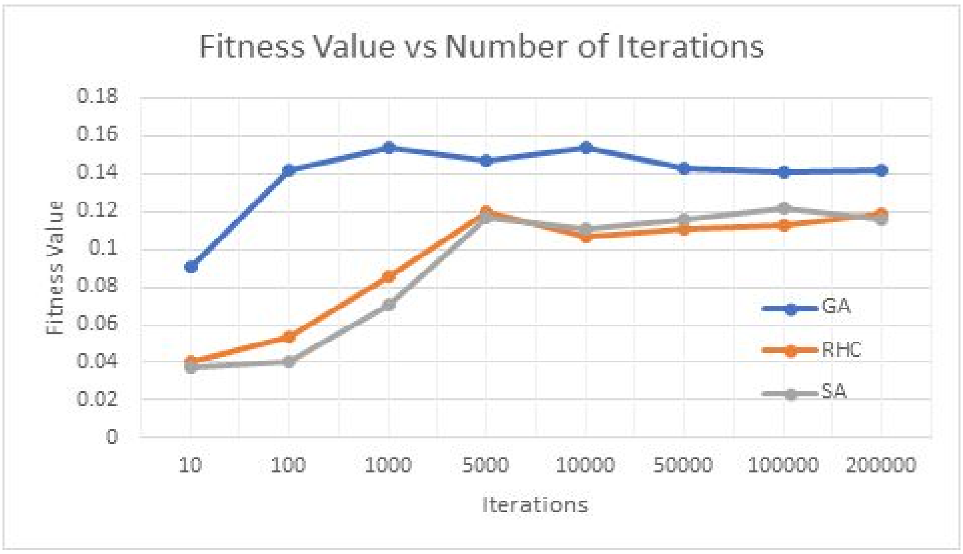
In this section, we have considered two optimization problems to analyze the behavior of algorithms such that one problem shows the advantages of Genetic Algorithm while second problem shows the advantages of Simulated Annealing. The performance has been compared based on fitness function value and the time taken to reach optimal solution.

Travelling Salesman Problem:

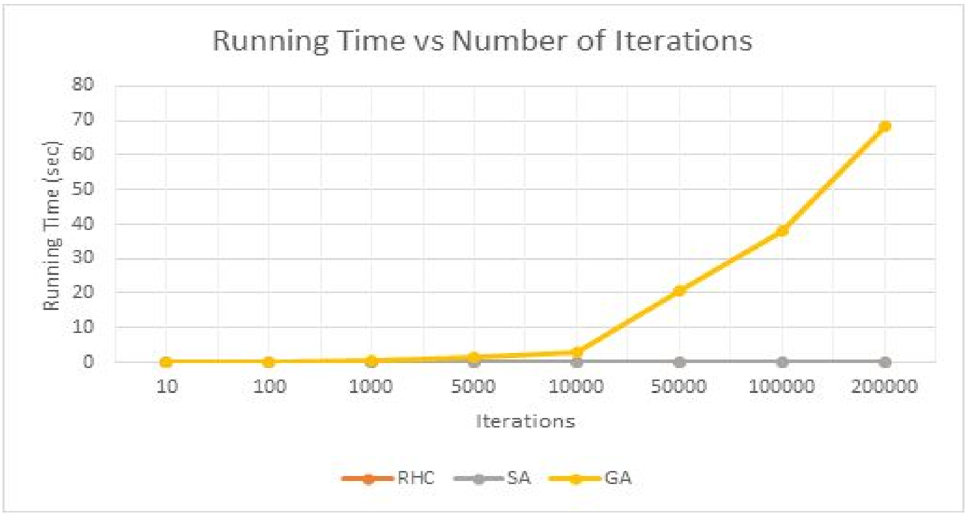
Travelling Salesman Problem (TSP) is a classic NP-hard problem, where given a set of cities and the distances between each pair of cities, the task is to find the optimal (shortest) route such that each of the city is visited exactly once. For the purposes of this assignment, I took the no. of cities (N) = 50. The coordinates were then randomly generated for each of the city, and the distances calculated between each pair.

The fitness function here is inversely proportional to the length of the route. Therefore, lower the length, higher would be the score for fitness function. Fitness value of all the algorithms have been checked for different number of iterations varying from 10 to 200,000. The experiment has been repeated for 3 trials for each iteration number and the average value has been recorded. For parameter values, default values in ABAGAIL library TSP example were used, cooling rate as 0.95 and starting temperature as 1E12 for Simulated Annealing and for Genetic Algorithm parameters value chosen were as follows: population size (200), mate size (150) and mutate size (20).

From the below fitness value vs number of iterations, it can be clearly seen the genetic algorithm performs best for this problem. RHC and SA though have lower training times at high iteration value, evident from running time vs number of iteration graph below but they never reach the same level of performance shown by GA. Genetic algorithm is best also because it reaches the highest value at 1000 iteration (here running time for all algorithms is almost similar) and their is very less effect on increasing number of iterations.

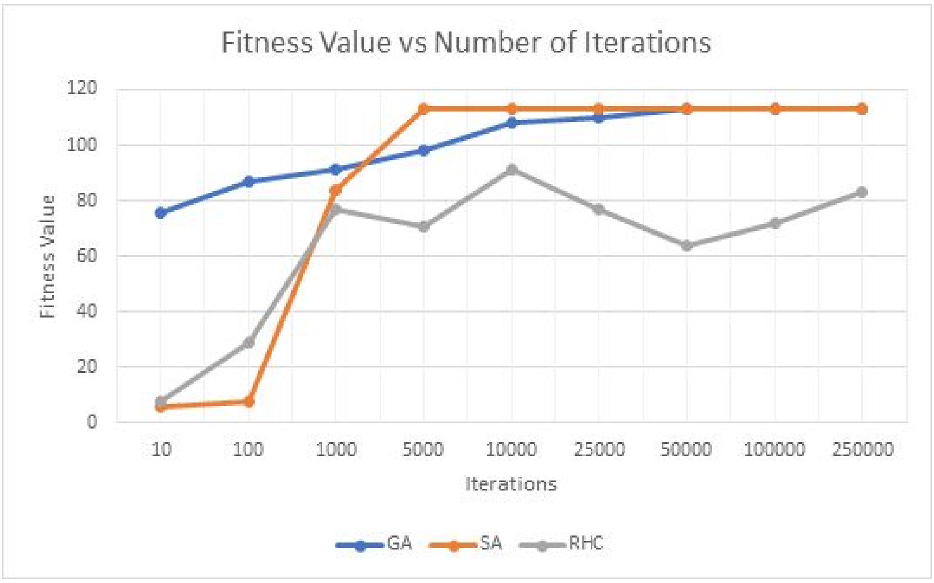


One reason for GA consistently being the best performer in case of TSP can be attributed to presence of underlying optimal substructure. This idea is similar to genetic algorithm where the members with highest fitness function are crossed in order to find an offspring that has the best parts of both the parents. For each pair of cities, an optimal path exists between cities and thus there are suboptimal paths present between pair of cities. It is highly possible that a crossover happens between two suboptimal paths, leading to a resulting path, which is better than both the suboptimal paths used for crossover. Thus, GA is going to be the best performer on problems which have such optimal substructures. RHC and SA being greedy algorithms (looking for neighboring cities greedily) can get stuck at local optimas.

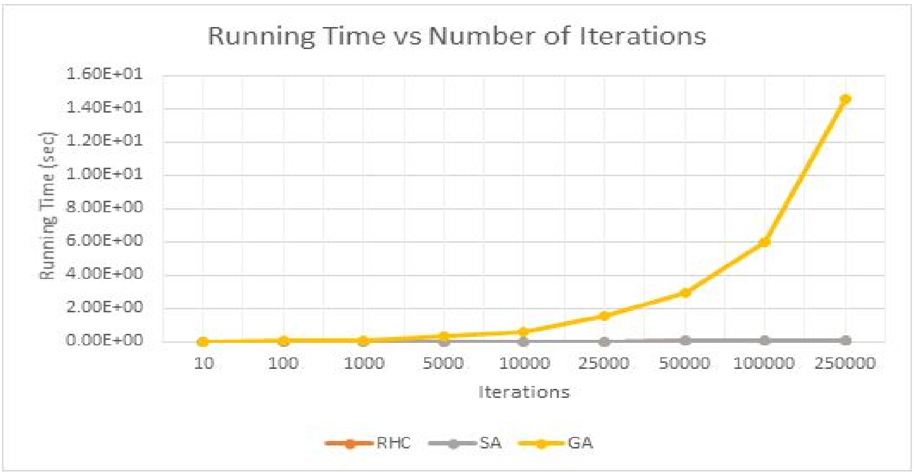


Continuous Peak Problem:

The continuous peaks problem is a problem where there are many peaks in a single dimensional space and our aim is find the highest peak among them. Due to presence of multiple peaks, exploring each and every part of the space becomes more important to reach global optimum. The value of N was taken as 60 and T was set to 6. In continuous peak problem, there are high chances of getting stuck at some local optima. For parameter values, I used default values in ABAGAIL library continuous peak example, cooling rate as 0.95 and starting temperature as 1E11 for Simulated Annealing and for Genetic Algorithm parameters value chosen were as follows: population size (200), mate size (100) and mutate size (10).



As evident from the graph of fitness value vs number of iterations above, RHC doesn't perform well because of getting stuck at different local optimas.It never reaches the global optimum value. Simulated annealing because of it's exploring nature and tendency to accept even worse paths based on probability makes it perform better than RHC for this problem. Genetic algorithms also reach the global optimum value of 113 but it takes around 25000-50000 iterations whereas Simulated Annealing just takes 5000 (**around 1/10th of GA iterations)** iterations. The time taken for GA varies based on the population size, number of members that cross and also number of mutations. But it is substantially high compared to both random hill climbing and simulated annealing. Therefore, considering both running time and fitness value, Simulated Annealing is the best algorithm for this problem.



Conclusion:

According to the results obtained for different problem statements following conclusions can be made:

- On using randomized optimization algorithms, testing accuracy was found to be lower than backpropagation.
- RHC and SA are ideal for problems which requires solution at low-cost (time).
- Random hill climbing can get stuck at local optimas and requires multiple restarts to increase the chances of getting global optimal solution
- Simulated annealing explores the search space more initially gradually reducing the exploration as temperature begins to cool but choosing the parameters is more work as the performance completely depends on it.
- Genetic Algorithms are best suited for complex problems like Travelling Salesman Problem which has optimal substructures but GA is computationally intensive as it requires more training time compared to RHC and SA.

References:

1. Phishing Dataset : https://www.kaggle.com/akashkr/phishing-website-dataset
2. ABAGAIL Library: https://github.com/pushkar/ABAGAIL