**Georgia Institute of Technology CS 7641 – Machine Learning**

**Randomized Optimization**

**Santhanu Venugopal Sunitha (ssunitha3)**

The purpose of this paper is to explore randomized optimization algorithms namely - Randomized Hill Climbing, Simulated Annealing, Genetic Algorithms and MIMIC.

**Part One**

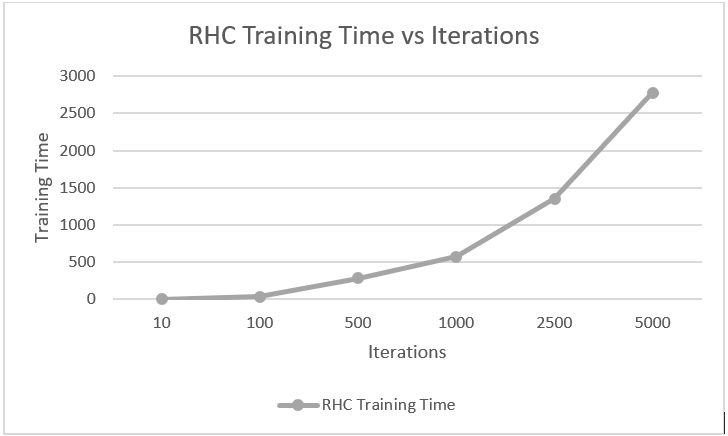
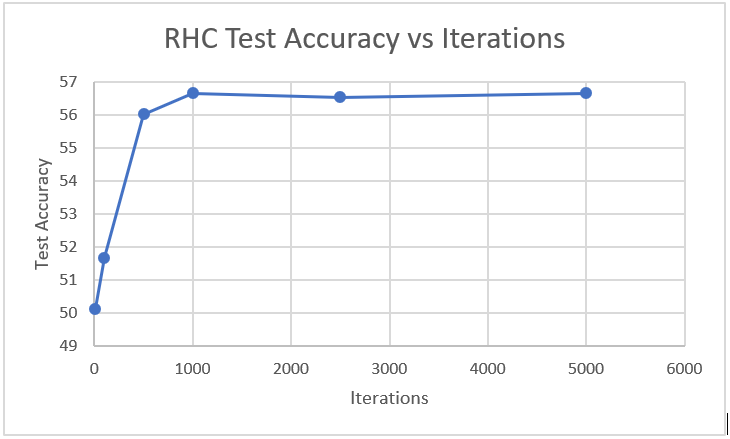
This part deals with applying Randomized Hill Climbing, Simulated Annealing and Genetic Algorithms to find the optimal weights for a neural network on Madelon dataset. The purpose of this part is to compare the Back-propagation technique in a neural network with the above three optimization algorithms.

**Methodology**

Madelon Dataset from UCI ML repository (4400 instances, 500 attributes, 2 classes), which was used in assignment 1, was split into training and test sets in 70/30 ratio and the optimization algorithms were executed on the dataset using ABAGAIL library.

**Randomized Hill Climbing**

As hill climbing algorithm can get stuck at local optima, randomized hill climbing circumvents this by choosing to randomly restart to a different point in input space upon reaching a local optimum.

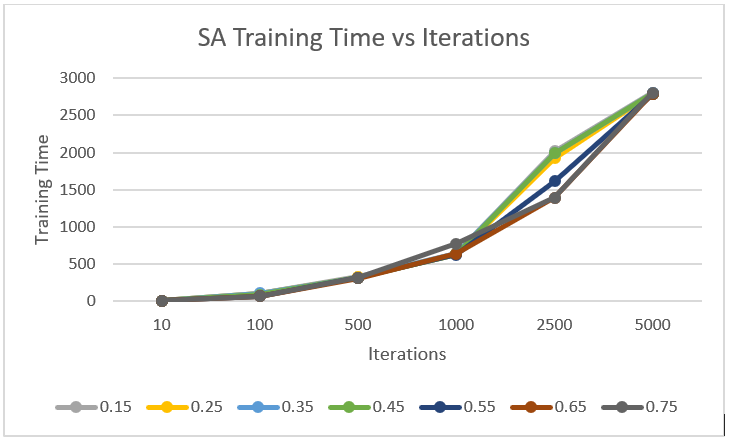
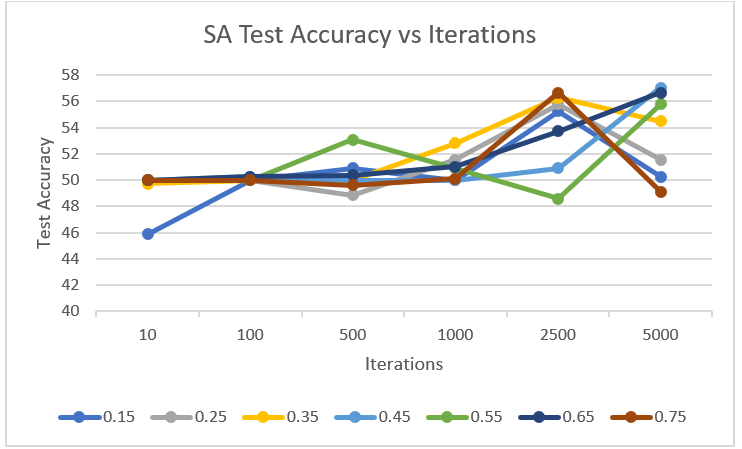


RHC is one of the top performing algorithms among all three. It converged to the optimal weights in 1000 iterations and reached a test accuracy of ~56%. The training time scales linearly with increase in iterations. The reason for RHC converging quickly can be attributed to having fewer local optima in the input space for the weight function or in other words, wide basin of attractions. It is observed that with fewer iterations, RHC possibly got stuck in local optima, due to less no. of random restarts in the space and hence could not converge to optimal solution.

Unlike the other two algorithms, RHC does not have any hyperparameter to tune and hence it can only improve by increasing the iterations up to a certain point.

**Simulated Annealing**

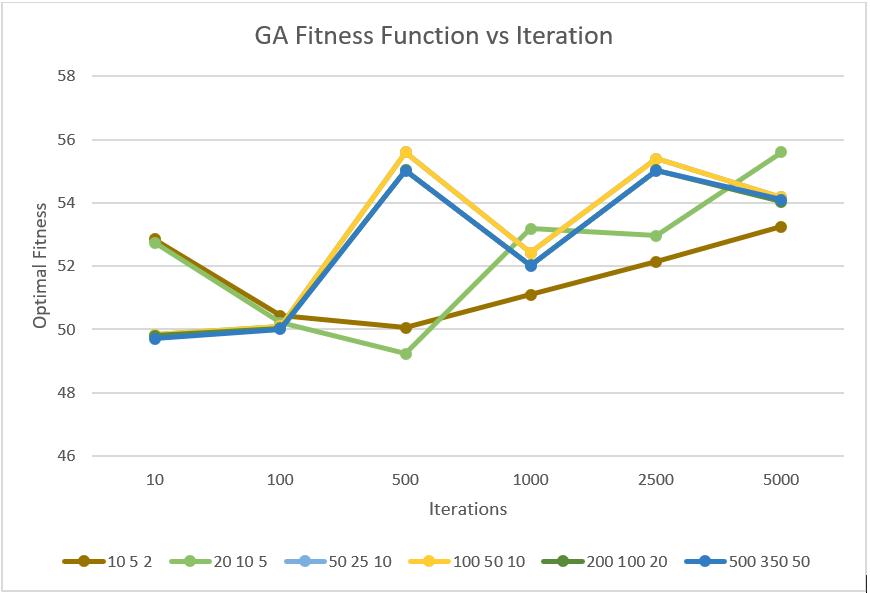
Simulated Annealing algorithm starts with an initial high temperature - IE11 and then slowly cooled based on the cooling factor to converge to an optimal solution. When the temperature is high, the algorithm will accept solutions worse than current best solutions, hence preventing getting stuck in local optima. Subsequently, with reduction in temperature, the acceptance of worse solution reduces, and this gradual cooling helps to find the solution closest to global optimum.



With varying cooling rates, it can be observed that with cooling rate of 0.45, the algorithm reached highest test accuracy of ~57% with 5000 iterations, making it the best among the other three algorithms. It is also observed that different cooling rates behaves differently under varying iterations. The training time scales linearly with iterations here as well.

With a high initial temperature and low iterations, it potentially got stuck at local optima as the temperature would not have been low enough at the end of the iteration, hampering its ability to jump out of local optima. However, with increase in iterations, it became effective in finding close to approximate solution as the temperature is cooled based on the cooling factor. By tuning the cooling factor hyperparameter over varying iterations, the algorithm performance can be improved. Also, by varying the initial temperature, a better solution may be found if there are many local optima for the problem.

**Genetic Algorithms**



**Part Two**

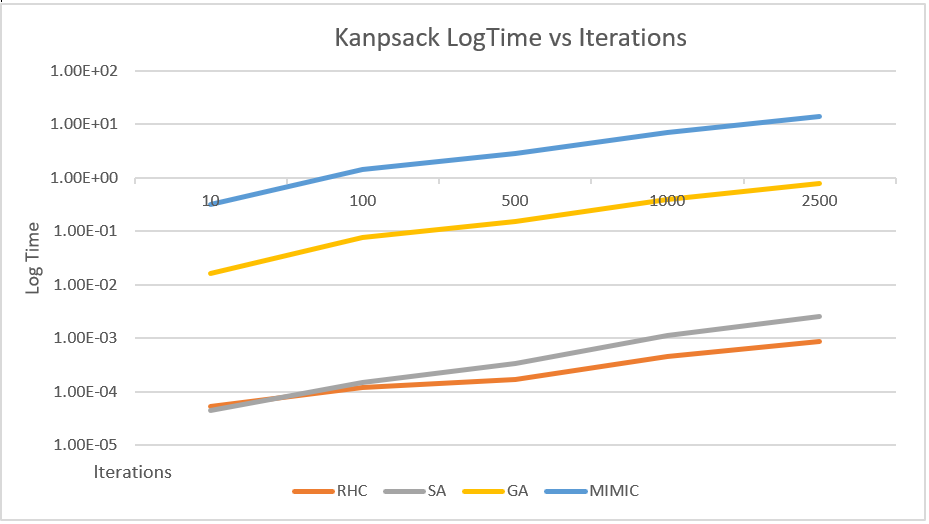
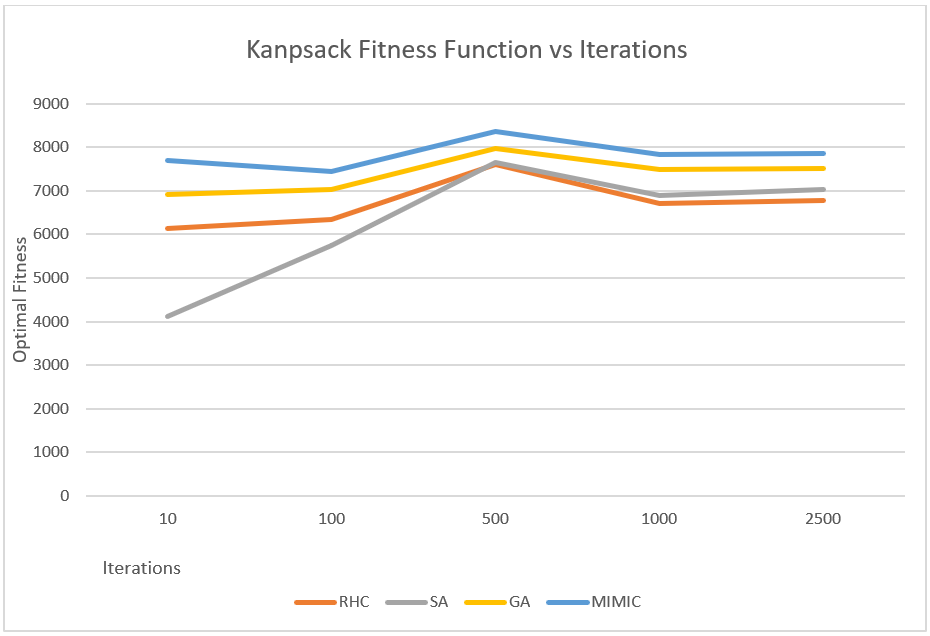
In this part, the four randomized optimization algorithms - Randomized Hill Climbing, Simulated Annealing, Genetic Algorithms and MIMIC, are applied to three optimization problems, namely Travelling Salesman, Knapsack and Continuous peaks.

**Knapsack Optimization Problem**

The premise for Knapsack problem is as follows - Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. The optimization problem is [NP-hard](https://en.wikipedia.org/wiki/NP-hard) and there is no known polynomial algorithm which can tell, given a solution, whether it is optimal.

For this analysis, we apply the four optimization algorithms to this problem with following attributes:

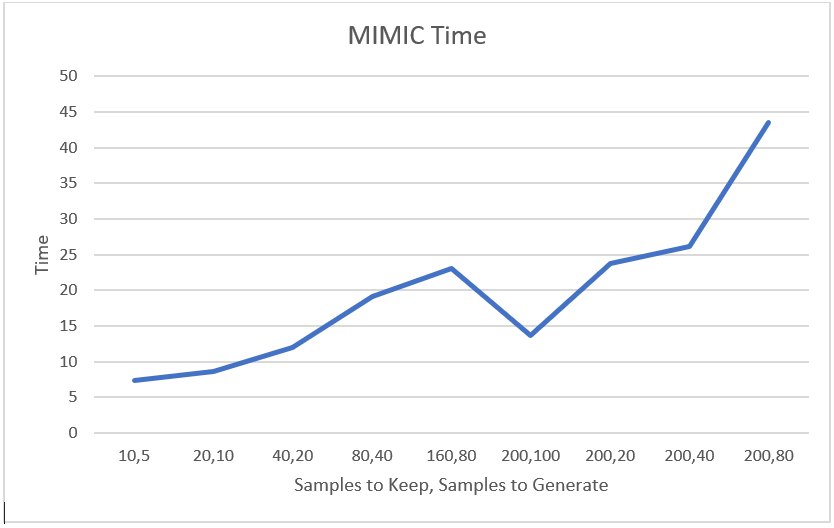
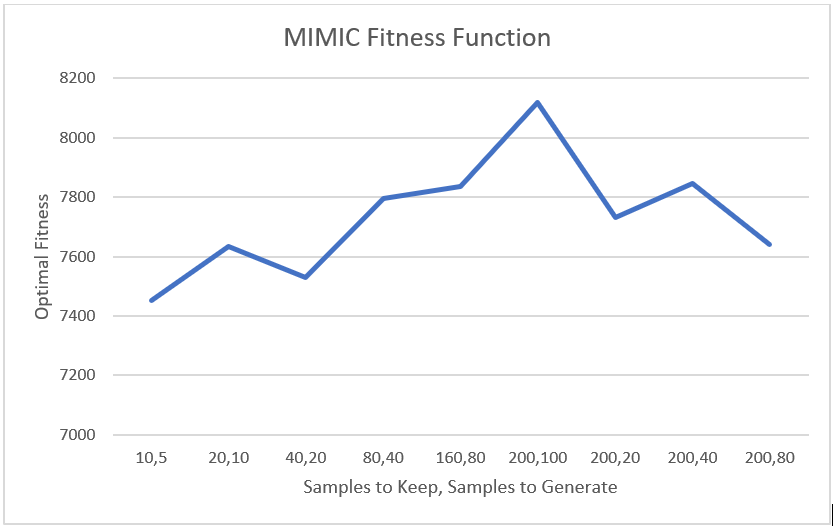
Number of Items: 60 with 4 copies of each, Max Weight: 70, Max Volume: 70, Max Knapsack Volume: 2800.



Based on the above plots, MIMIC algorithm has the upper hand as it provides the best fitness score at 500 iterations, but at a higher run time in comparison to other algorithms. It seems to have converged around 1000 iterations. MIMIC algorithm was successful in capturing the underlying structure of this problem and use that information to perform better optimization search. It uses joint probabilities to create dependency trees to understand the dependencies within the data and was able to convey the same to subsequent generations.

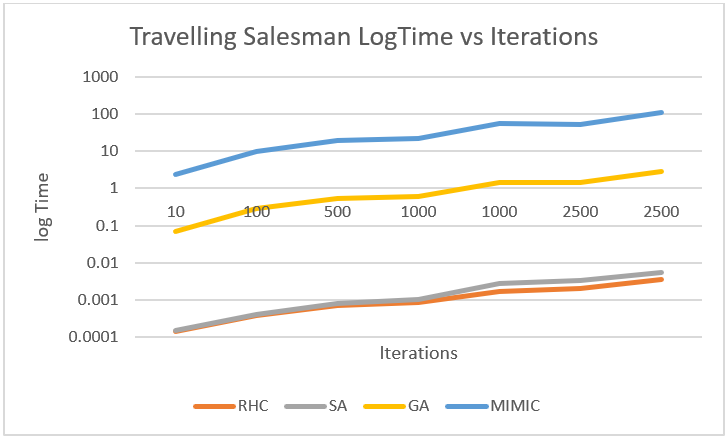
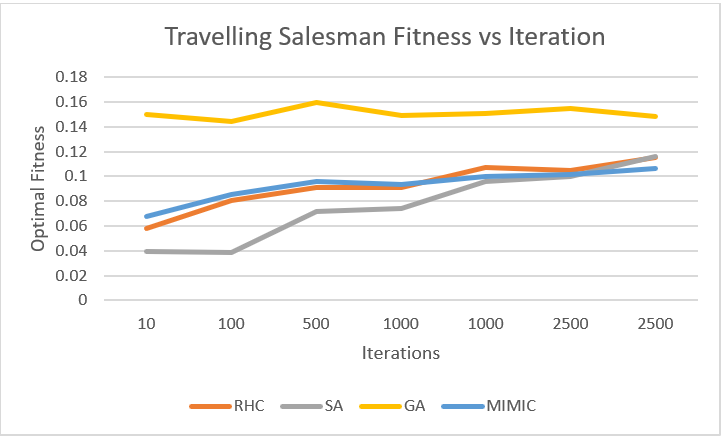
RHC seems to have performed the worst followed by SA. SA could not work well here as this problem may not be suited for exploratory search and knowing the structure and passing the information from one generation to another could only result in the best fitness score, as done by MIMIC and GA.

The below plots depict the hyperparameter tuning performed for MIMIC, which shows that 200,20 (Samples to keep, Samples to generate) was the best in terms of higher fitness with lower training time and hence the same was used in the above experiment.



**Travelling Salesman Optimization Problem**

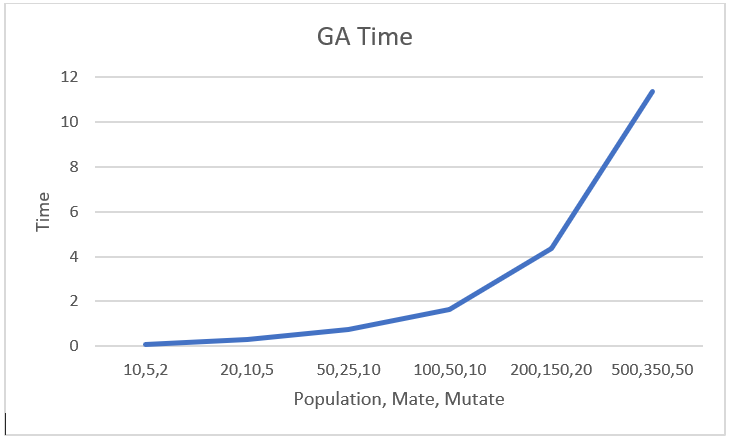
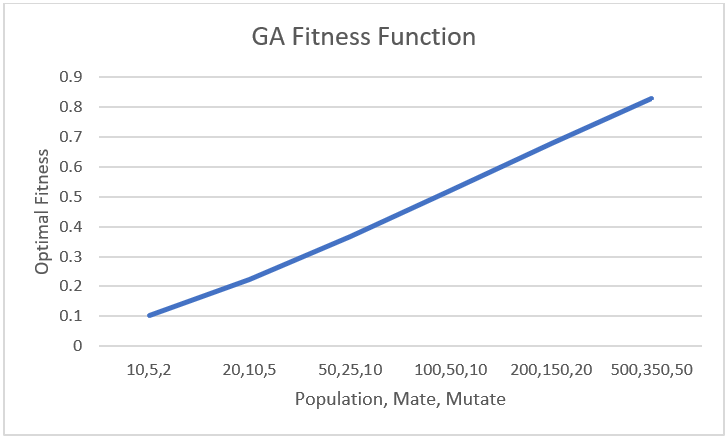
Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point. Here, we set the no. of customers for the salesman to visit as 50 (N = 50), which means that there are 50! ways the salesman can visit all the customers. This being an NP Hard problem, evaluation of randomized search algorithms may not provide the best solution to this problem, but somewhere close to it. However, doing do would throw light on which algorithm would perform better under this circumstance.



Based on the above plots, it is evident that GA performs the best for this problem. Though comparatively GA took more time than RHC and SA, it was faster than MIMIC to find the optimal fitness score. It can also be noted that RHC and MIMIC found similar fitness values at each iteration.

Genetic algorithms are based on the process of evolution by natural selection which has been observed in nature. They essentially replicate the way in which life uses evolution to find solutions to real world problems. A randomly generated initial population is evaluated to determine their fitness, followed by selecting the best members and performing crossover and mutation to induce more randomness and this gets repeated until an optimal solution is reached.

GA was quick to find the best solution at lower iterations. In this problem, as the salesman should not visit the same city twice, GA is implemented using swap mutation and special crossover techniques. Hence, only the best routes are passed on from one generation to another as the members with most fitness score has higher chance of survival as well as being selected for mating and mutation. This is the reason for the out performance of GA over other algorithms.

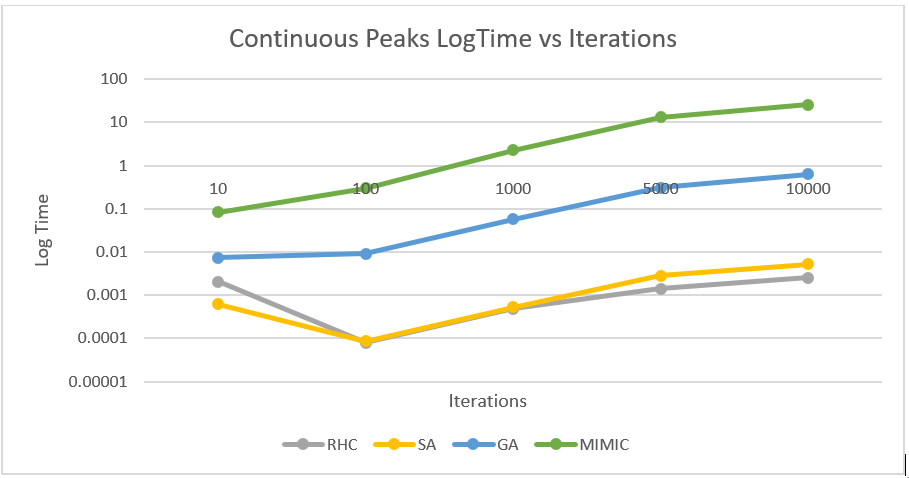
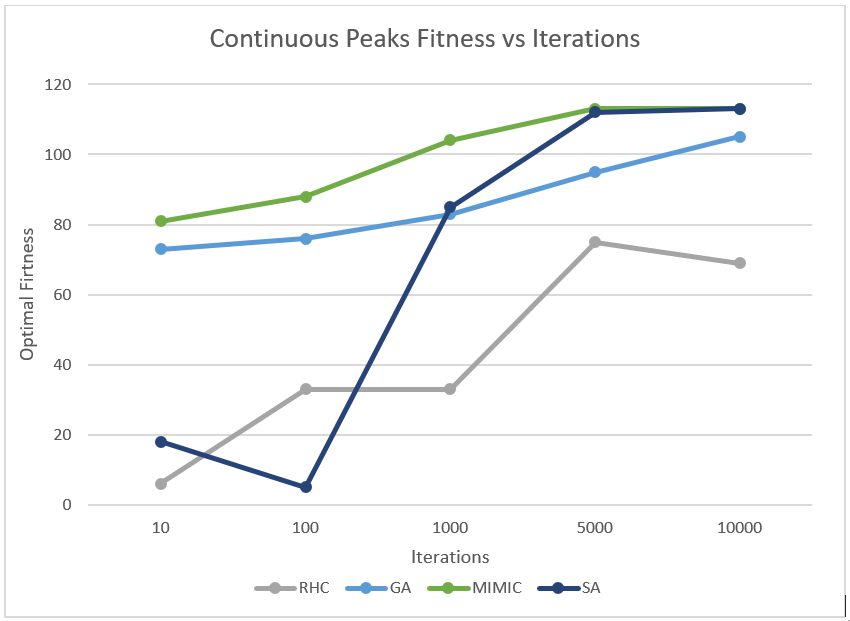


The above plots indicate that with increase in hyperparameters – Population, Mate and Mutate, the fitness score linearly increases for GA at the expense of computation time. Hence, the algorithm can be improved by increasing these values.

**Continuous Peaks Optimization Problem**

This problem deals with finding the global optima among the many local optima in the search space. For this experiment, below attributes were set for the algorithms :

N = 60, T = 6

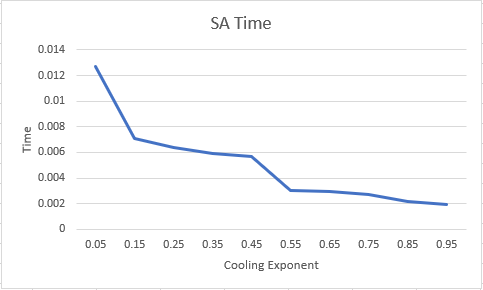
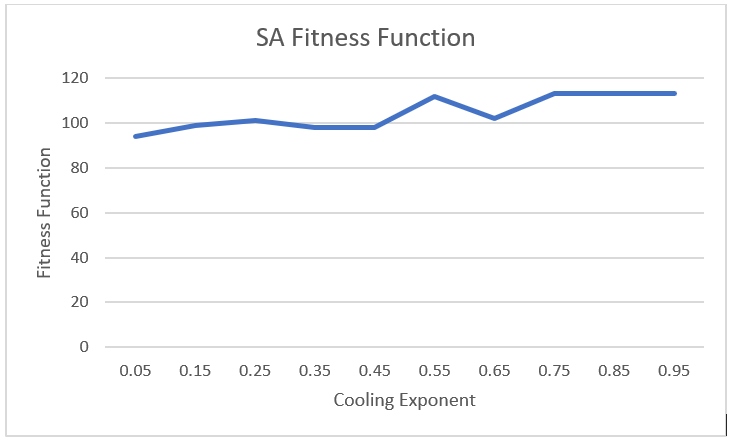


Both MIMIC and SA performed well but MIMIC took more time for training and hence SA is the best algorithm for this problem. Both SA and MIMIC converged to the best fitness score in 5000 iterations. SA was set to an initial temperature of 1E11 and a cooling factor of 0.95

RHC, being the worst performing algorithm, mostly got stuck at one of the many local optima initially and even with increasing iterations, it still could not find the solution closer to global optima. GA was quick to find a good solution with fewer iterations but could not reach the best solution, which could be attributed to mutation of the chromosomes, which in turn might reduced the overall fitness of the population, as there are many local optima in this problem. However, based on the plot, increasing the iterations can be promising for GA to improve its fitness score.

SA performed well here as it is well suited for large search space to find a solution closer to global optima. The fact that it started with a high temperature and performed cooling at a rapid rate, it was able to discard the worst solutions quickly from the distribution and converged to the optimal solution in very less time.

Based on the below plots, the cooling factor hyperparameter was found optimal at 0.95 as it provided the best fitness for the least amount of time.



Conclusion