Randomized Optimization

Student Name: Sathish Sampath
GT Account Name: ssampath33

Introduction

This report compares the performance of the four random optimization algorithms - Randomized Hill Climbing(RHC), Simulated Annealing(SA), Genetic Algorithm(GA) and MIMIC. In part one, the performance of the first three algorithms, in determining the good weights for a neural network is compared along with the back propagation for the 'Adult' dataset from the Assignment 1. In part two, all four random optimization algorithms are applied to three optimization problems - "Continuous Peaks", "Flipflop" and "Travelling Salesman Problem"., their performances are compared and the best algorithm is selected. The random optimization algorithms are implemented using the ABAGAIL library.

Randomized Optimization Algorithms

Randomized Hill Climbing (RHC)

Randomized Hill Climbing is an iterative algorithm. First, A random point is selected. The algorithm searches the neighbouring points and determines the best one, which has the optimal value. This best neighbour is chosen as the new point. The algorithm then determines the best neighbouring point for the new point. This step is repeated until it reaches the peak, where there is no optimal neighbourhood points available. This point is saved as an optimal solution. Then a new point is selected in random and the whole searching process is repeated to find the optima. The solutions from all the searches are compared and the global optima is determined. The major advantage of this algorithm is, it is easy to implement and faster compared to other optimization algorithms. But this algorithm can get stuck in local optima.

Simulated Annealing (SA)

Simulated annealing is a probabilistic technique for approximating the global optimum of a given function. The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. Simulated Annealing follows an Explore and Exploit approach. When a neighboring point is better than the current point, it becomes the new point. But if all neighbours are not better than the current point, then the next point is determined by the acceptance function results. The acceptance function results are based on the Temperature, current value and the neighbour's value. During each iteration, the value of the temperature is reduced. As long as the temperature is high, the algorithm will search(explore), also taking worse neighbours. But when temperature is low, the algorithm performs standard hill climbing and reaches the nearest optimal position.

Genetic Algorithm (GA)

Genetic algorithm is a metaheuristic approach inspired by the process of natural selection, where the population evolves by iteratively mating and mutating parts to crossover the best traits and to eliminate irrelevant traits. The algorithm operates by iteratively updating a pool of hypotheses, called the population. On each iteration, all members of the population are evaluated according to the fitness function. A new population is then generated by probabilistically selecting the most fit individuals from the current population. Some of these selected individuals are carried forward into the next generation population intact. Others are used as the basis for creating new offspring individuals by applying genetic operations such as crossover and mutation. The main drawback of GA is, it cannot handle huge hypothesis space. Though the GA captures the structure of the problem initially, the offspring generated do no preserve the structure, as the crossover point is chosen randomly.

Mutual Information Maximizing Input Clustering (MIMIC)

MIMIC determines the optima by using the structure of solution space, which it builds using the probability density functions. The better solution space structure is built over multiple iterations and the better solutions points are carry forwarded to the next iteration. The major drawback of this algorithm is the time complexity, as the MIMIC draws from the distribution, samples and reiterates over the points in each evaluation.

Part 1 – Neural Network Optimization

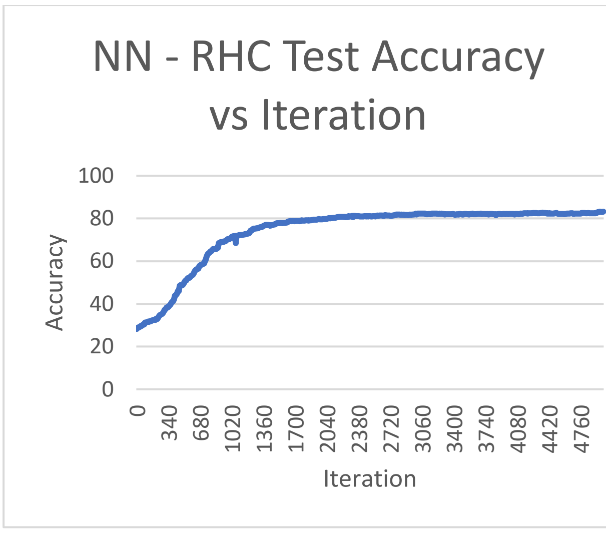
The three optimization algorithms are implemented in the neural network and their performance is compared. The 'Adult' dataset and the corresponding neural network created from the Assignment 1 is used for the comparison purpose. The dataset is a subset from the 1994 US Census, which is used to relate socio-economic attributes like education, heritage and age (among others) against the adult's income, in this case, whether income is above or below $50,000 per year. The data is used to develop neural network for the classification of adult profiles based on their socio-economic attribute values. For the evaluation of the performance of the algorithms, the overall dataset is split into two: 70% as training set and 30% as test set. For cross validation purposes, the K-Fold method with 5 folds is implemented on the training part using the Scikit-Learn's inbuilt function. The backpropagation is used to optimize the weights for the neural network in the Assignment 1. The best results from the Assignment 1 are mentioned below.

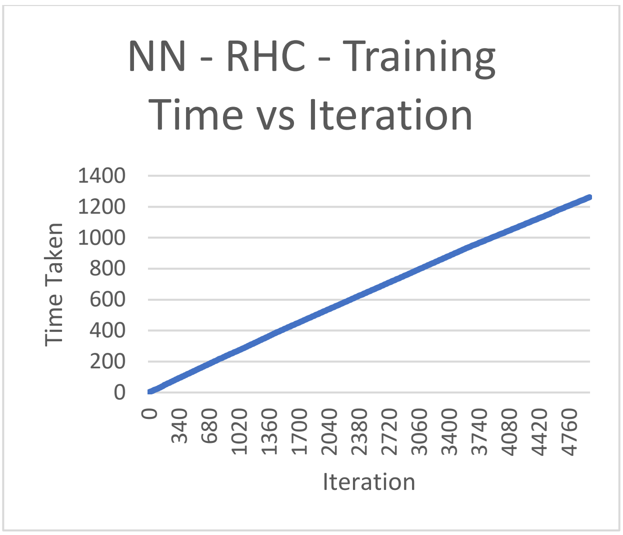
Input Layer size	59
Alpha	0.01
Activation	Logistic
Hidden Layer Size	(58, 58, 58)
Output Layer size	1

The randomized optimization algorithms are run using the same neural network structure, as obtained from the Assignment 1. The algorithms are executed for multiple iterations up to 5000 and the results are saved. The algorithms are implemented using the ABAGAIL Java library. The Jython wrappers are built to run the individual algorithms and save their output.

Randomized Hill Climbing

The RHC algorithm is implemented to optimize the neural network. The resulting test accuracy is plotted against the Iterations. As the iteration increases, initially the accuracy increases, but after 2000 iteration, the accuracy becomes constant. Few distortions in the accuracy is also observed. This mighh be because of the randomness of the algorithm. The points are selected in random, so there is a chance that during certain iteration global optima might not be even obtained. The training time is quite linear with iteration and it is very fast. We can conclude that, in RHC as the iteration increases, there is high chance that we will get global optima and get higher accuracy. But with iteration, the training time also increases.

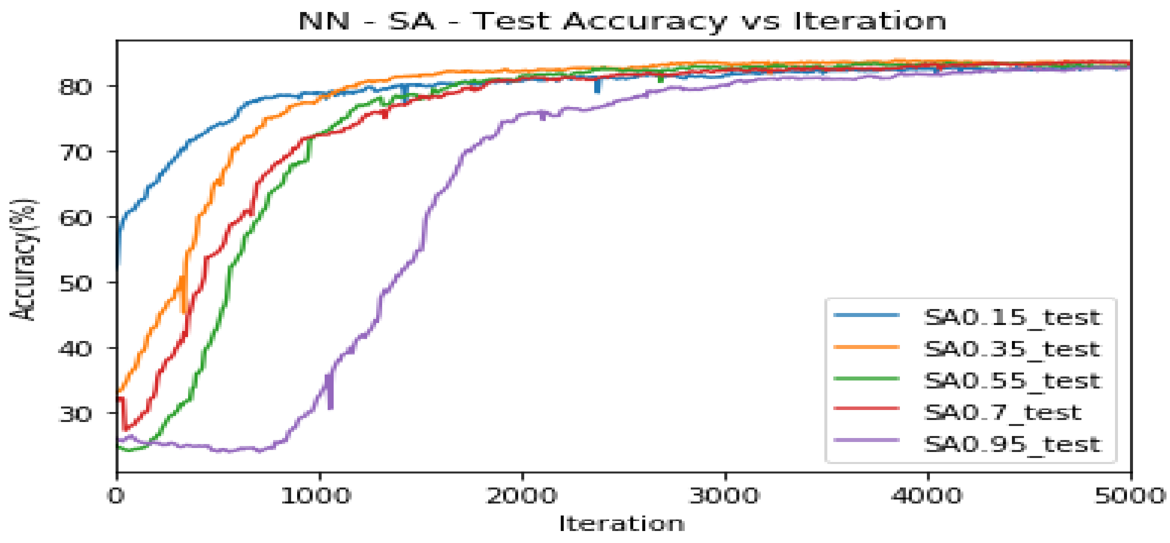




RHC - Best accuracy: 83.196% (Iteration: 5000)

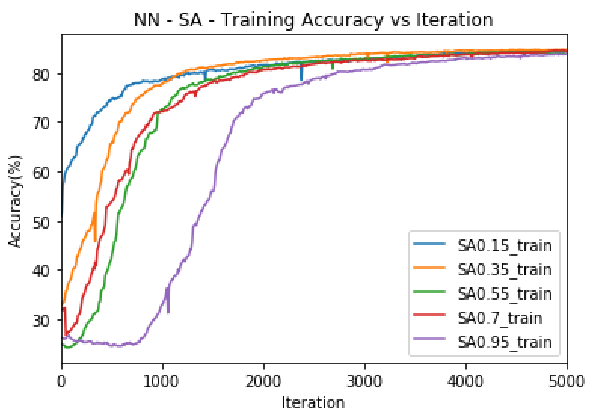
Simulated Annealing

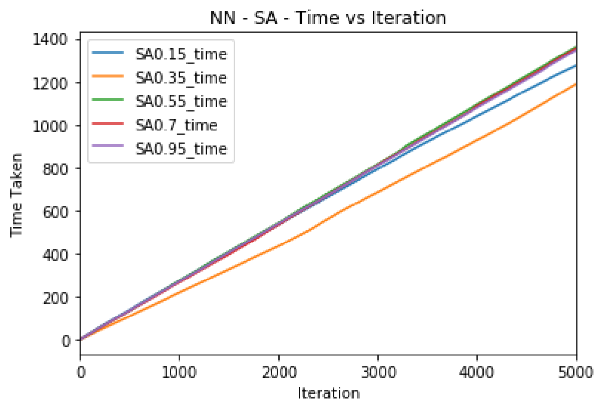
The Simulated Annealing is run for multiple cooling exponents (0.15, 0.35, 0.55,0.75,0.95) and the training accuracy, test accuracy and training time are plotted to show how the NN performs with different values of cooling exponents(CE). The Temperature is set as 1E10.



From the plots, it is clear the performance is lower for lower iteration, but it gradually increases with the iteration. Also, we can see that, as the cooling exponent is lower (0.15,0.35), the performance is much better. The accuracy increases faster. As the Temperature is higher in many evaluations, due to low cooling exponent, the algorithm would have chance to explore more and finally settle in the global optima position. Also, for lower cooling exponent, training time is lesser compared to higher cooling exponent value.

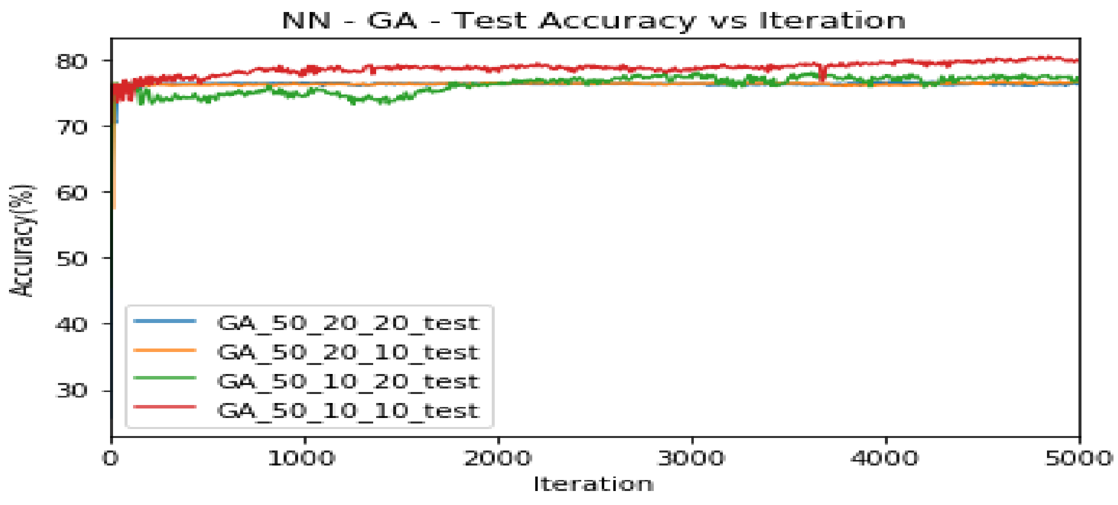
Best Test Accuracy SA: 84.74% (Cooling Exponent: 0.35, Iteration: 5000)

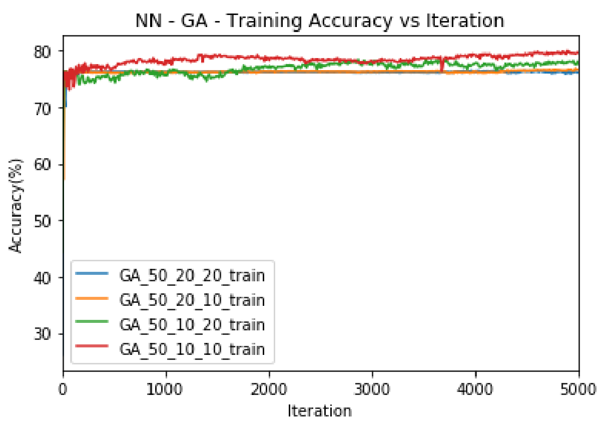


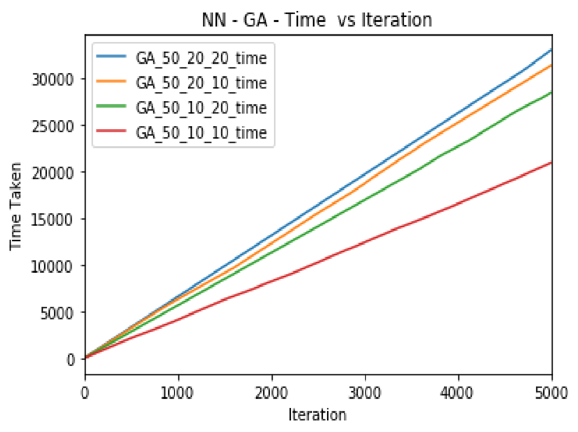


Genetic Algorithm

The GA is run for the population value of 50, mate values (20,10) and mutate values (20,10), to analyse the performance of the algorithm using various combinations of hyperparameter. From the plots, it is clear that lower value of mate and mutate are performing well in optimizing the neural net. The time taken is also faster compared to higher values. This might be because, as the crossover is not high, the optima might be faster to reach in this particular problem.



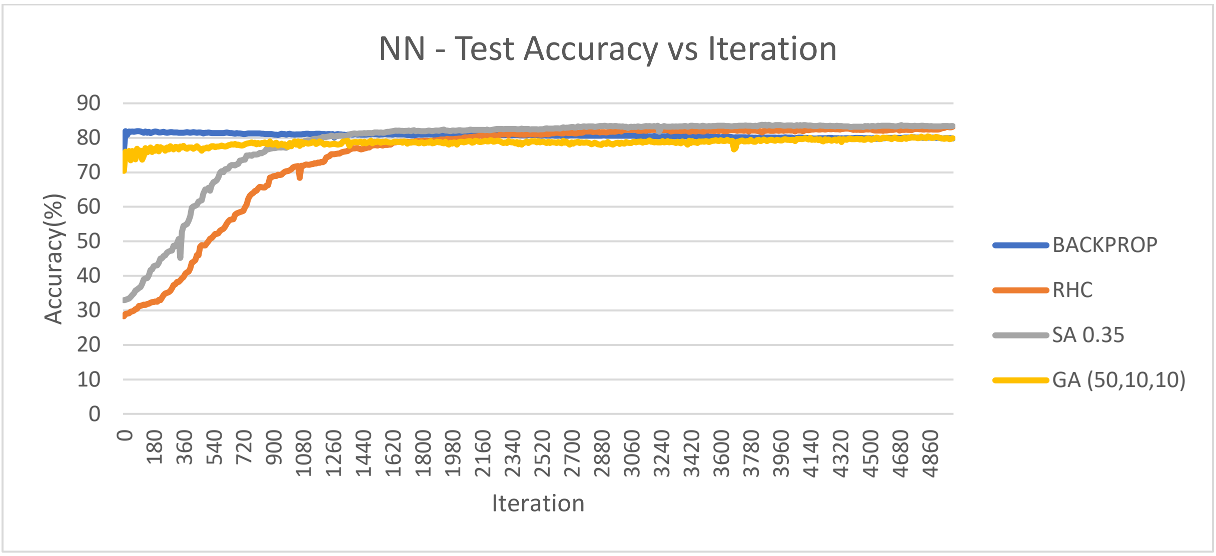


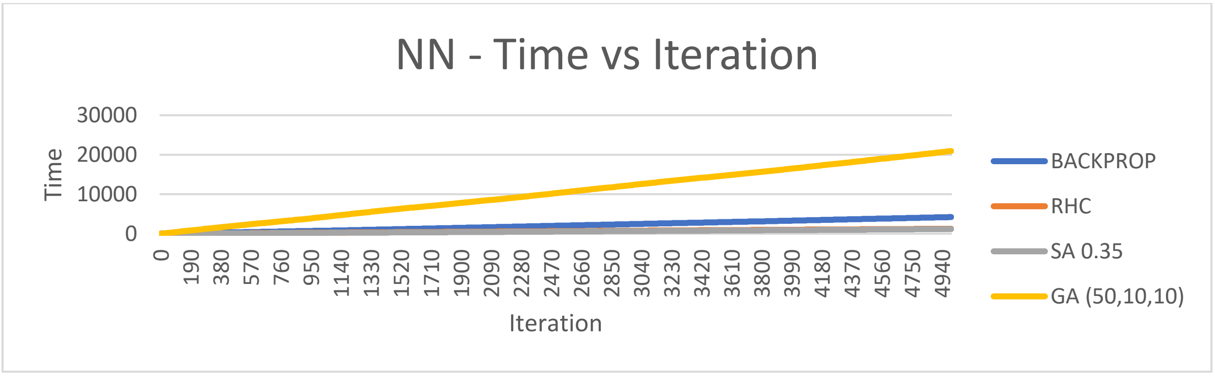


Best Test Accuracy GA: 80.36654 (population: 50, mate: 10, mutate: 10, iteration: 4810)

Comparison of Randomized Optimization Algorithms

The Test Accuracy and Time taken of the best solution from each randomized optimization algorithm and Backpropagation are plotted and the performance is compared.





Backpropagation and GA have reached higher accuracies with lower iteration, but they are slower as shown in the Time vs Iteration plot. RHC and SA have lesser accuracies for lower iterations but reach higher accuracies as the iteration increases. RHC and SA also consume less time. The results are shown in the table below. The Simulated Annealing has better accuracy and is also faster compared to the other optimization algorithm for the Neural Network. SA is also little bit faster compared to RHC in higher iteration, by few seconds.

Algorithm	**Parameter**	**Accuracy**
Backpropagation	Nil	81.9263
RHC	Nil	83.1936
Simulated Annealing	CE = 0.35	84.74
Genetic Algorithm	Population: 50, mate: 10, mutate: 10	80.36

Part 2 – Three Optimization Problem

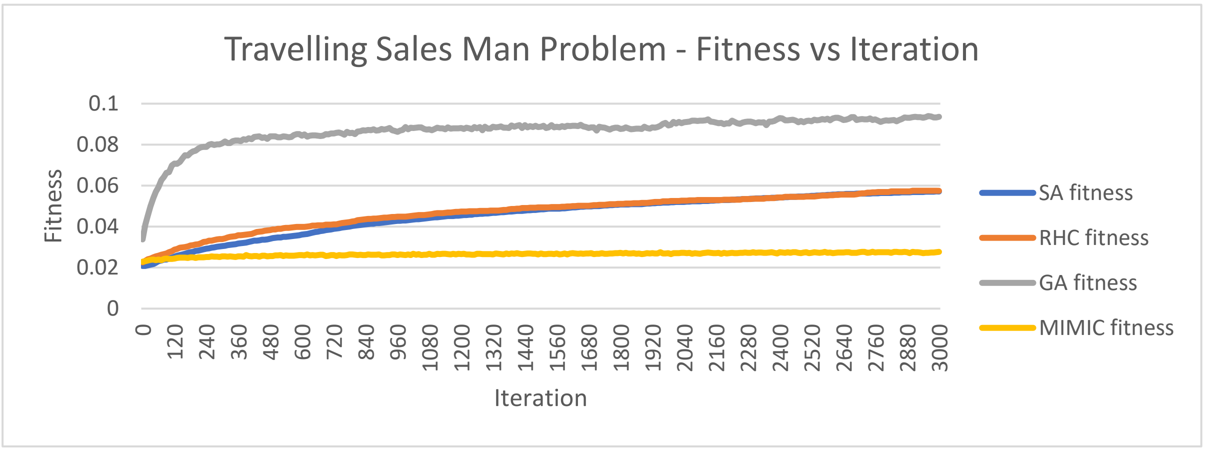
The four randomized optimization algorithms are implemented using ABAGAIL Java library for optimizing three Optimization Problems – Travelling Sales Man, Flipflop and Continuous Peak. The Algorithms were executed 5 times for different hyperparameter combination and the average is used for comparison purpose, to reduce the effect of randomness. The fitness function and time taken are used for the comparison purpose.

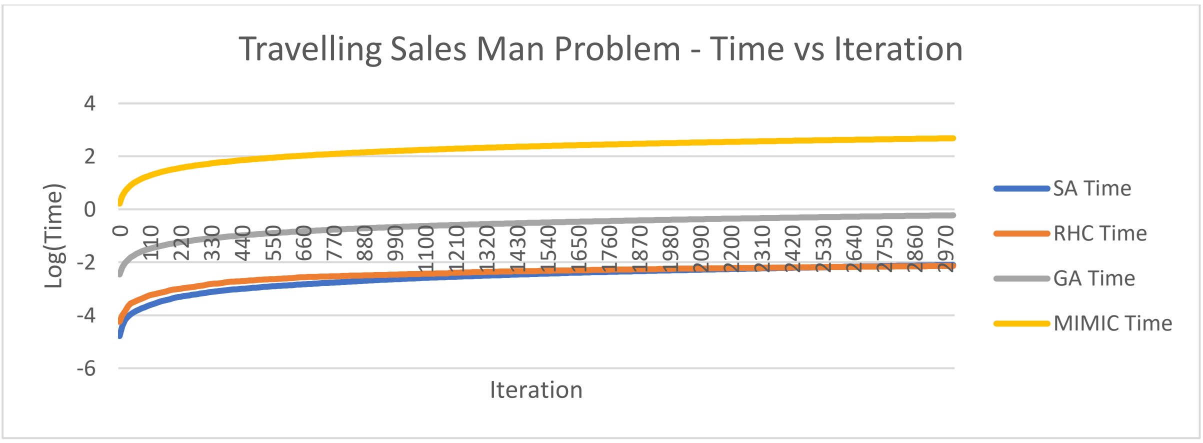
Algorithm	Parameters used
Simulated Annealing	Cooling Exponents = (0.15,0.35,0.55,0.75,0.95)
Genetic Algorithm	Population = 100, Mate = (50,30,10), Mutate = (50,30,10)
MIMIC	Samples = 100, Keep = 50, Epsilon = (0.1,0.3,0.5,0.7,0.9)

Travelling Sales Man Problem

Travelling Sales Man problem is a NP-hard problem, where the travel distance between multiple points has to be minimized and optimal route has to be decided. The TSP has many applications in the field of transportation, logistics, shipping and travel industry, as most of the destination points might not be in the same line and it requires computation to determine the optimal route with minimal distance for reducing the overall cost. The four optimization algorithms are run for iterations (1,10,20, to 3000). The best fitness from each of the optimization algorithm and the time taken by each algorithm were plotted against iteration count to analyse and compare the performance of the algorithms. The time taken is plotted in log scale as time taken by MIMIC was much higher.

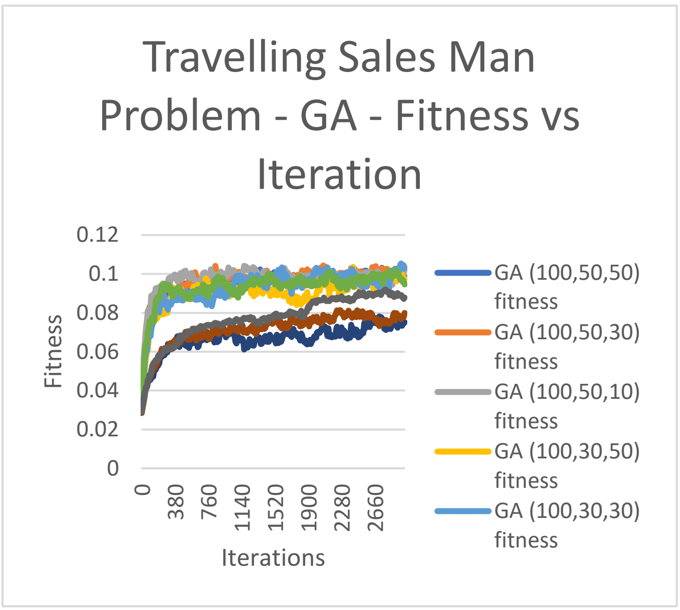
From both the plots, it is clear that three algorithms perform better as the iteration count increases. The RHC and SA fitness increases with iteration and GA has reached a stable fitness score within few iterations (<250). The MIMIC has the least fitness.

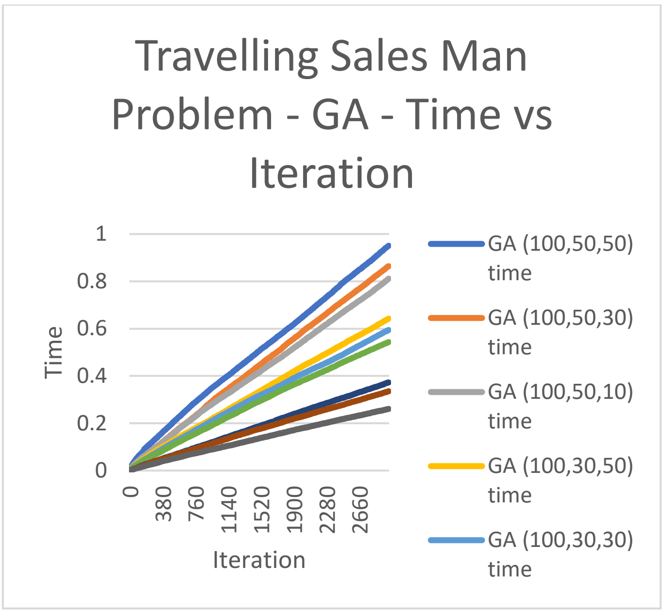




Genetic Algorithm performs better in terms of fitness and also reached stable fitness score in few iterations. The time taken is higher compared to RHC and SA but it is also considerably lower than MIMIC, so it is selected as the best optimization algorithm for TSP. Let's explore how it performs for

different hyperparameters (Population, Mate and Mutate).
Note: The graph labels are mentioned in (population value, mate value, mutate value) format



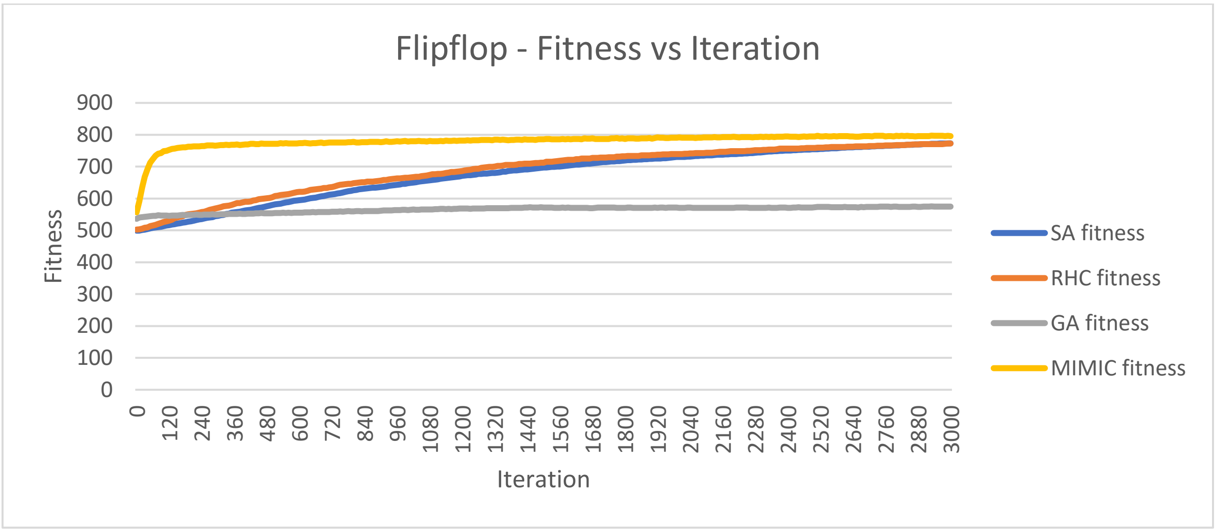


The Hyper Parameter Combination (100,30,30) performs well compared to other parameters in the fitness and is also comparatively faster to few combinations. So, this combination is selected as the best one for the GA implementation of optimisation for TSP.

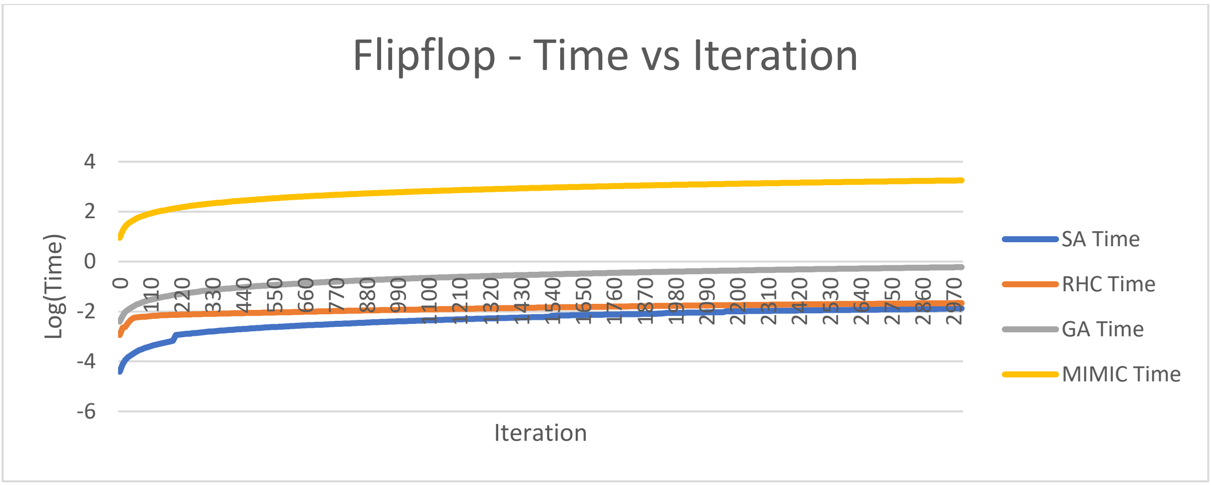
Optimisation Algorithm	Genetic Algorithm
Hyper Parameter	Population: 100 Mate: 30 Mutate: 30
Highest Fitness	0.105

Flipflop Problem

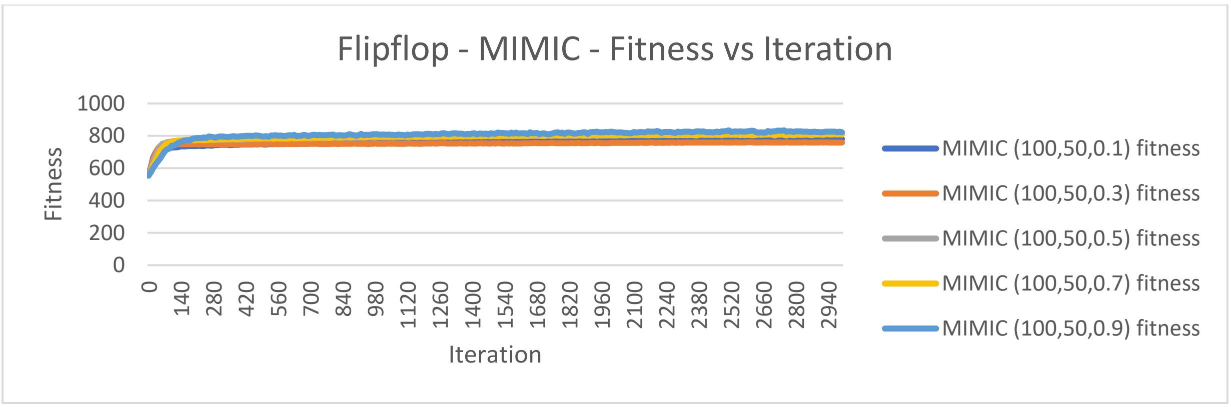
The flipflop optimization problem is an interesting optimization problem. The four-optimization algorithm are executed for iterations (1,10,20, to 3000). The fitness score and time taken for each of the algorithm is plotted for comparison purposes. The Time Taken curve is plotted in logarithmic scale.



All the algorithms have poor performance with lower iteration, but as iterations increase the fitness increases. The MIMIC fitness score raises faster when compared to other algorithms and reaches a stable score. The GA has the worst fitness for most of the iteration out of the four algorithms. The SA and RHC almost reached the MIMIC's fitness score with higher iterations (>2700 iteration), but still lesser than MIMIC. The MIMIC is comparatively slower than other four algorithms, because it has more number of evaluation with each iteration. The MIMIC is selected as the best optimization algorithm for Flipflop problem.



The hyperparameter combinations are varied for MIMIC and their corresponding performance is plotted. *Note:* The graph labels are mentioned in (Samples, Keep, change(Epsilon)) format

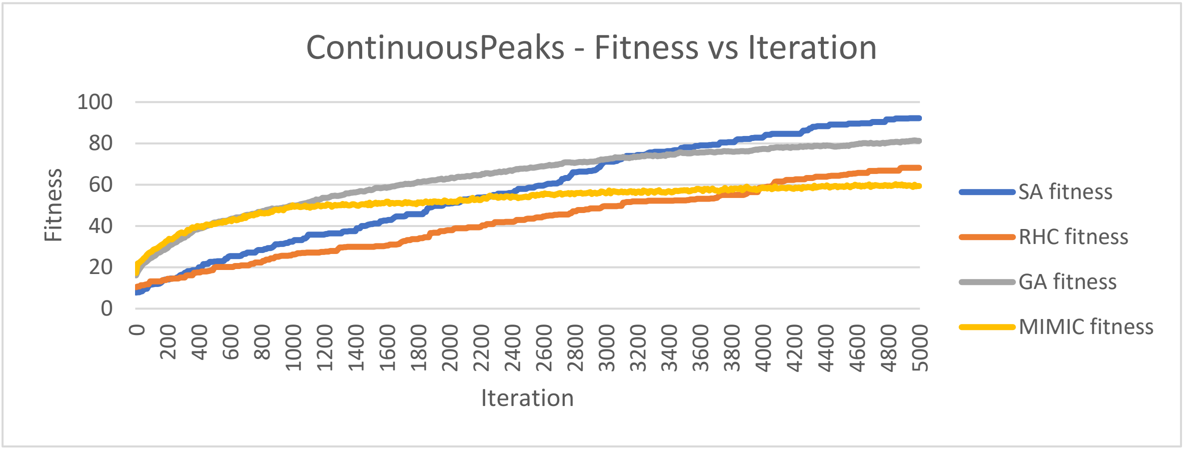


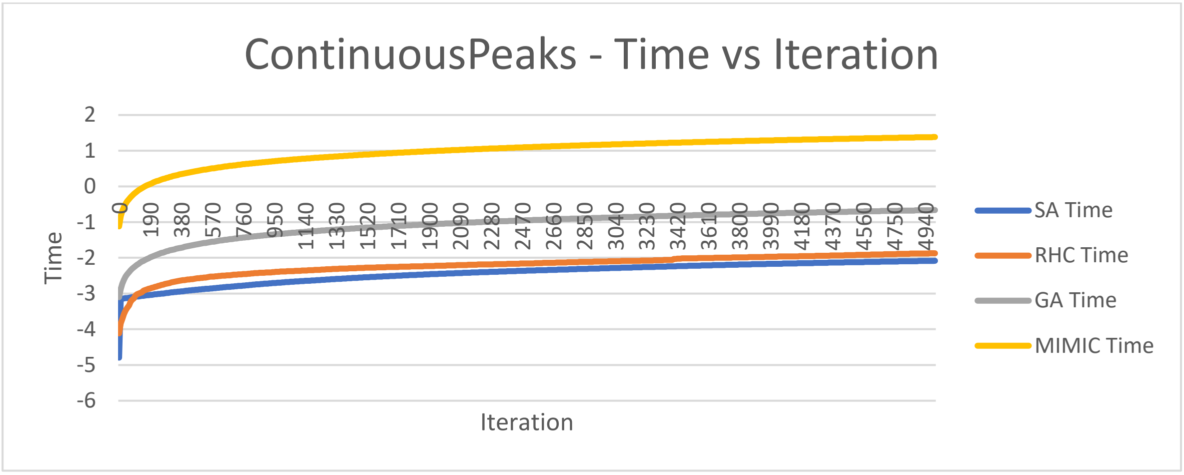
From the plots, it is clear that combination (100,50,0.9) performs better. It has better fitness and also reaches higher fitness value is lesser iteration.

Optimisation Algorithm	MIMIC
Hyper Parameter	Samples: 100 Keep: 50 Epsilon: 0.9
Highest Fitness	833.4

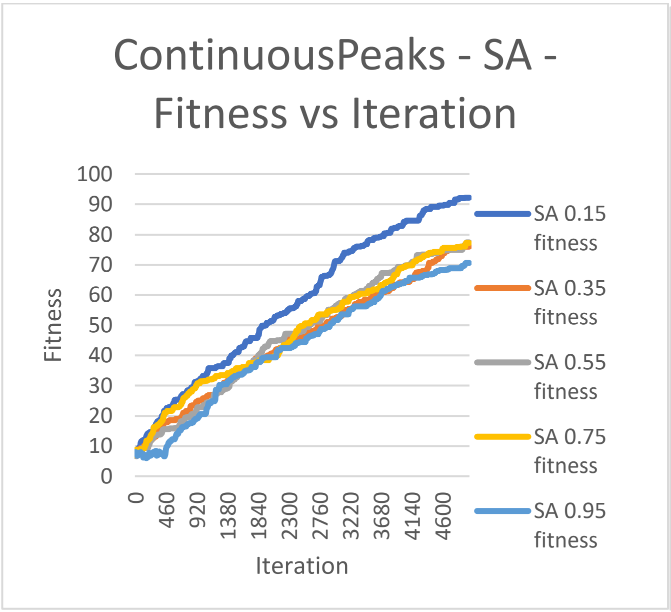
Continuous Peak

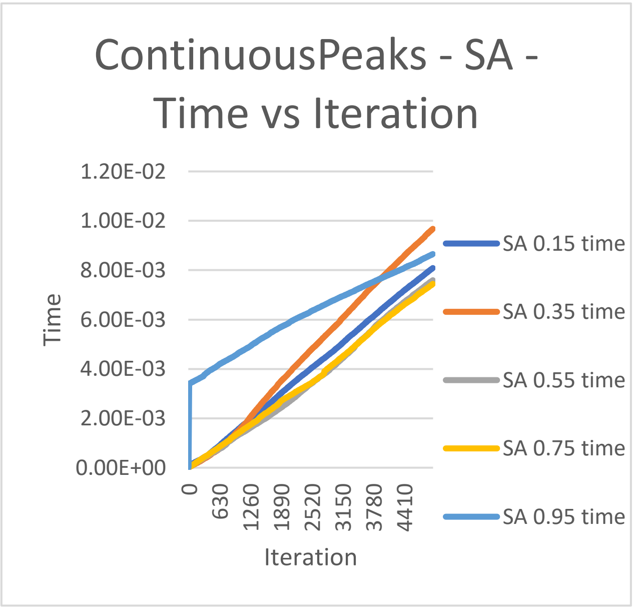
This problem has many local optima (peaks). It helps in highlighting the difference between multiple random optimization algorithms. The major application is in the surface optimization and topography analysis. The global optima is surrounded by many local optima. This makes it challenging for the optimisation algorithms are there is a chance that few may get stuck in local optima. The four algorithms are run for different iterations (1,10,20, to 5000) and the fitness score and the time taken is compared in the each algorithm.





From the plots, it is clear that all 4 algorithms improve their fitness score with iteration. As more and more iterations are added, the effect of randomness will reduce and there is a high chance that global optima are determined, this might be the reason. The Simulated Annealing(SA) performs better compared to other algorithms after 2500 iterations and is improving and also requires lesser time. So, SA is selected as the best algorithm and the performance is studied for different hyper parameter.





The SA with Cooling Exponent 0.15 performs well compared to other Cooling exponent curves and also has consumes lesser time. So, it is selected as the best hyperparameter.

Optimisation Algorithm	Simulated Annealing
Hyper Parameter	Cooling Exponent: 0.15
Highest Fitness	92.2

Conclusion

The different Randomized optimisation algorithms are compared and their performance in a NN implementation and three interesting optimization problems are studied. The following conclusion is derived.

Algorithm	**Parameters**	**Computation Speed**	**Comments**
Randomized Hill Climbing	Nil	Very Fast	Good for continuous value, simple structure, low cost problems. Performs well with higher iteration.
Simulated Annealing	Cooling Exponent, Temperature	Fast	Good for continuous value, simple structure, low cost problems. Comparatively better than RHC
Genetic Algorithm	Population, Mate, Mutate	Medium	Can handle both complex structure and simple structure.
MIMIC	Samples, Keep, Epsilon	Slow	Good for complex structure, but computationally costly.

Reference

1. Machine Learning Book, Tom M. Mitchell
2. Adult Dataset, UCI ML Repository, https://archive.ics.uci.edu/ml/datasets/adult
3. Simulated Annealing Wikipedia, https://en.wikipedia.org/wiki/Simulated_annealing
4. Code Source: Jonathan Tay Github(https://github.com/JonathanTay/CS-7641-assignment-2)