Randomized Optimization in Machine Learning

Darsh Thakkar (dthakkar33)
Code and Readme.txt: https://github.com/darshthakkar/Randomized-Optimization-Machine-Learning

Introduction:

Implementation and analysis of four optimization algorithms, namely - Randomized Hill Climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA) and Mutual Information Maximizing Input Clustering (MIMIC) has been done and entailed in this report. In the first part of the report, first three algorithms have been implemented to maximize accuracy of the neural network by optimizing the weights of the neural network. For the second part, all four algorithms are used to solve 3 common optimization problems and the results are compared to find which one performs the best in which scenario.

The dataset used here for the first part is the Wine Quality (White) dataset form Kaggle that was one of the dataset utilized in assignment one. The dataset doesn't have missing values. Size of the dataset 4879 samples and 11 features which give description and data required by the classifier to produce quality of the wine (10-outstanding, 1-very poor). Dataset is divided into three parts – test, training and validation using dumper.py to provide all three curves (train, test and validation) for all optimizing problems.

Randomized Hill Climbing (RHC)

Randomized Hill Climbing is an iterative algorithm. First a random point is selected in the feature space. The algorithm searches the neighboring points and determines the best one, which has the optimal value based on some loss function. This best neighbor is chosen as the new point. The algorithm then determines the best neighboring point for the new point. This process keeps on happening for the number of iterations that have been specified to the algorithm.

There is also another variation with random restarts. The algorithm above is run with different initial points in the space. 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 in non-convex optimization functions.

Simulated Annealing (SA)

Simulated annealing is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic to approximate global optimization in a large search space for an optimization problem. It is often used when the search space is discrete. 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 neighbors 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 neighbor'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 neighbors. 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 that 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 algorithm tries to reach the optima by understanding the solution space structure. It is done by utilizing the probability density functions. The solution space structure is built over multiple iterations and the solutions which do better than the defined threshold are taken over to next iteration. However, biggest drawback of this algorithm is the time complexity as MIMIC draws from the distribution, samples and reiterates over the points in each evaluation.

Part 1: Neural Network Optimization

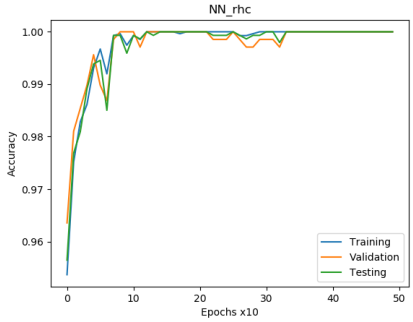
To implement the neural network framework, the ABGAIL library is used. Using this library, the normal backpropagation optimization function for a neural network can be replaced with any optimization algorithm. The details of the neural network are as follows; the dimensions are based on the analysis done in assignment 1:

Dimensions of Neural Network	11 x 8 x1
Activation Function	Tangential Hyperbolic Sigmoid
Evaluation Metric	Accuracy

The neural network itself is constructed using the functions in the ABAGAIL library. Even with a relatively small neural network, the results are quite good on the dataset as concluded earlier and hence a deeper network is not used. When training the neural network, different metrics like MSE error, accuracy, number of iterations and time elapsed is logged into a file which is later used to plot graphs and for the analysis. The results of this optimization for the three algorithms are as follows.

Randomized Hill Climbing

Randomized hill climbing starts from an initial point in the 11D feature space (as there are 11 attributes in this database), evaluates its neighbors and picks the one with the best improvement on accuracy. The randomized hill climbing algorithm in the ABAGAIL works the same way and the graph below shows the training, validation and testing accuracy of the algorithm on the Wine Quality dataset.

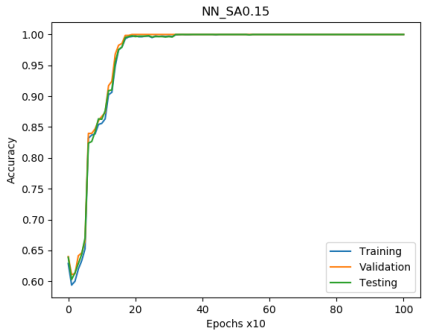


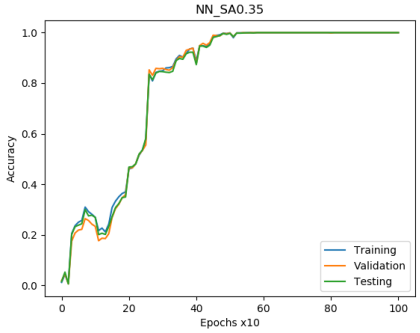
The graph given above for hill climbing shows that overall accuracy increases over iteration but not in a smooth fashion which is expected, as it is taking steps sometimes that might be getting it closer to a local optimum but farther away from the global optimum. After 500 iterations all the three accuracies converge to almost 100%.

Simulated Annealing

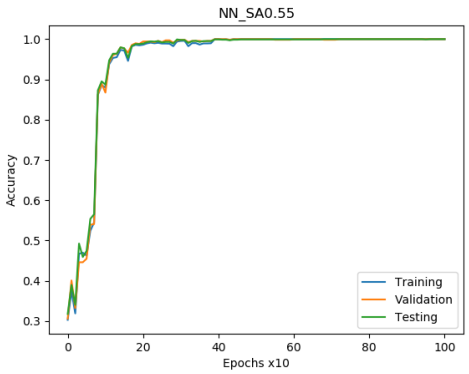
This technique works in a similar fashion to randomized hill climbing but as discussed before if it doesn't find a neighbor that improves accuracy it sometimes takes the step anyway with a certain probability. This probability is a function of the temperature (iterations) which is high at first but then goes down as the algorithm progresses. The simulated annealing algorithm in the ABAGAIL works the same way and the graph below shows the training, validation and testing accuracy of the algorithm on the Bank Note dataset.

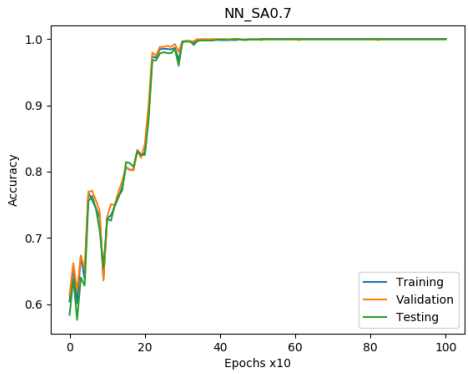
The graphs are ordered in increasing order of the '**cooling component**'. The title on top of the graph indicates the cooling component for the model. The higher this component is the larger the probability of taking a random action and the longer it takes to the probability to go to zero. When the probability goes to zero it behaves in the same way as randomized hill climbing.

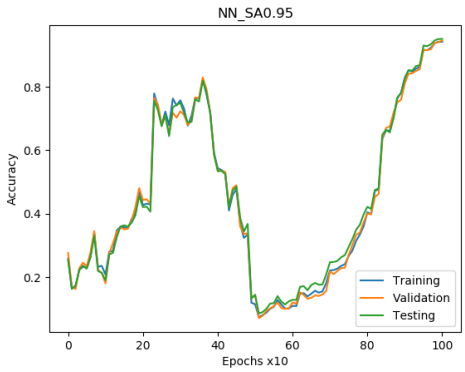




As we can see from the graphs a larger value of the cooling component causes the algorithm to make random decisions and hence there is more erratic change with respect to the accuracy. But with a large amount of iterations regardless of the cooling component all the models converge to an accuracy of close to 100%.

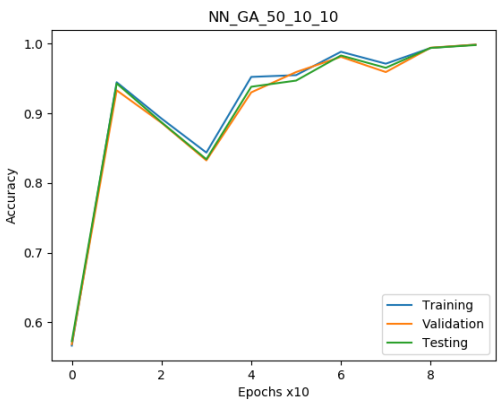


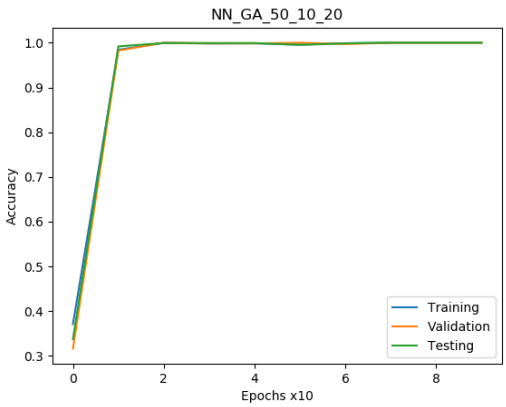


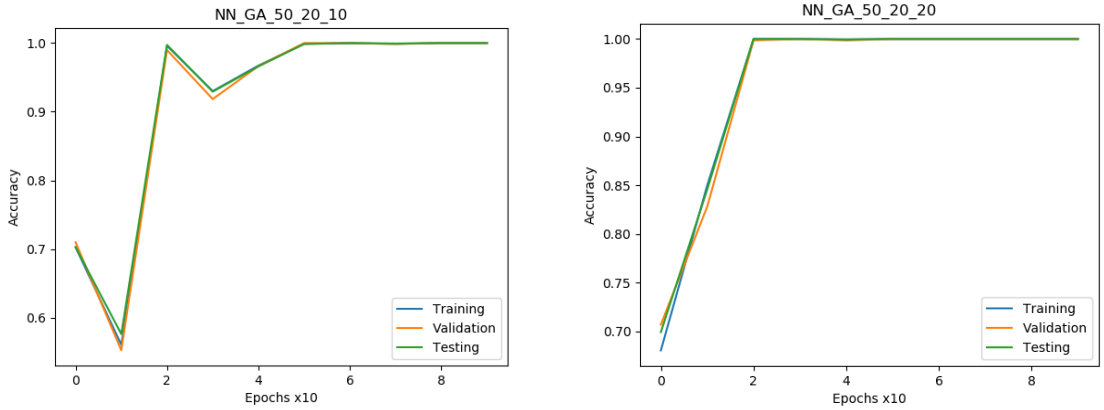


Genetic Algorithm

For the analysis of genetic algorithms to train neural networks 3 hyper parameters have been chosen namely **population size, mating and mutation rate**. These can be changed in the ABAGAIL implementation to train different models. The titles of the graphs show these three parameters respectively. Population size is not changed as it does not have much effect on this data set.







Although all models eventually converge on the global optimum the graphs show that the smaller the value of the mutation rate shows large spikes with respect to accuracy in the initial 100 iterations.
To draw comparison among the 3 algorithms used simulated annealing and genetic algorithms show large spikes in the graph as they are searching more points in the feature space than the random hill climbing algorithm. These results are consistent with the theoretical description and performance of the algorithms.

Part 2 – Optimization Problems

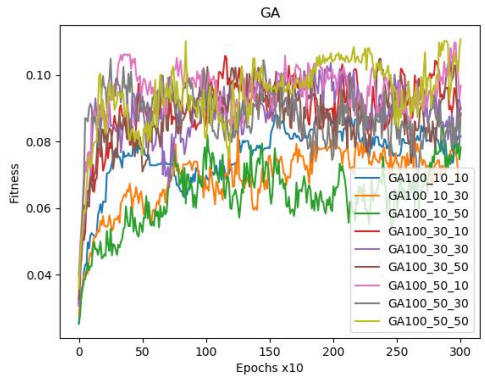
The four randomized optimization algorithms are implemented using ABAGAIL Java library for optimizing three Optimization Problems – Travelling Salesman, Flipflop and Continuous Peak. The Algorithms were executed for different hyper parameter combinations and the average is used for comparison purpose,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

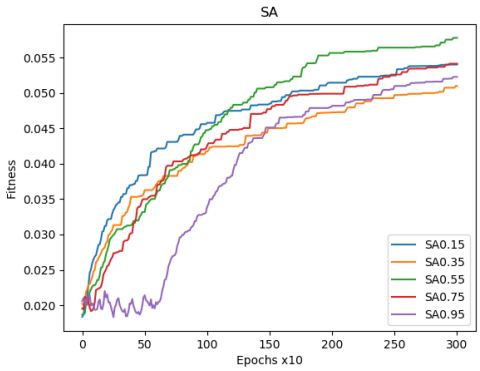
Algorithm	**Parameters**
Simulated Annealing	Cooling Exponents = (0.15,0.35,0.55,0.75,0.95)
Genetic Algorithm	Population = 100, Mate = (50,30,10), Mutate = (50,30,10)
MIMIC	Samples=100, Keep=50, Epsilon=(0.1, 0.3, 0.5, 0.7, 0.9)

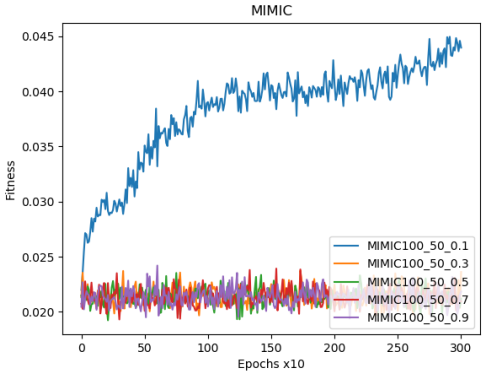
Travelling Salesman Problem

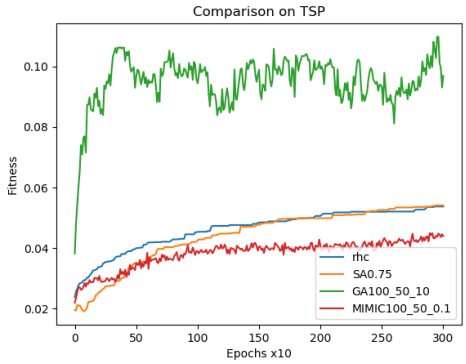
The travelling salesman problem (TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?" It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science. Here, the ABAGAIL implementation of the traveling salesman problem has been used to create models of four randomized optimization algorithms. A graph of 100 nodes has been used to train these models.

Each of the algorithms best hyper parameter combination performance is compared with other algorithms and a comparison between all is also plotted. In these graphs and all following graphs, the fitness on the Y axis relates the performance of the algorithm on the problem. A higher fitness means a better solution for the problem.









From the graphs we can conclude that simulated annealing with 0.75 as the cooling component, genetic algorithm with population size 100, mating parameter 50 and mutation rate 10 and MIMIC algorithm with 100 samples, kept back and 0.1 epsilon work the best. Here a larger value of the cooling component works well as it allows the model to search a larger region of the feature space. Smaller values of the meeting hyper parameter of the genetic algorithm seem to do related really poorly as there is only a small amount of crossover leading to less of the feature space being covered. In case of MIMIC, epsilon of 0.1 allows the algorithm to freely move towards the global optima, while high value of epsilon restricts the movement to optima in this particular structure, MIMIC also takes overall more time compare to other algorithms.

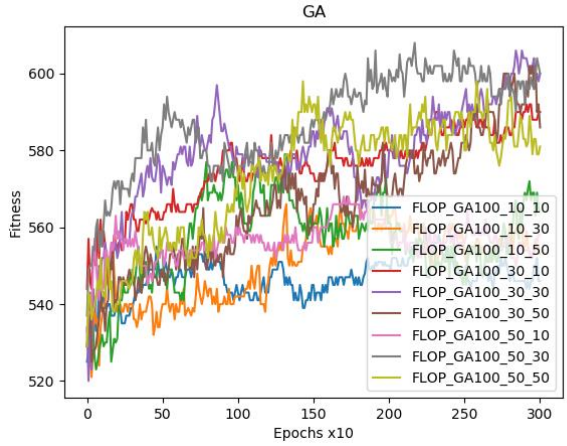
From the comparison graph we can see that the genetic algorithm clearly outperforms other 3 algorithms. The genetic algorithms performance is almost twice as good as the other 3 and simulated annealing performs similar to randomized hill climbing as number of epochs increase. Genetic algorithms can search a larger portion of the solution space and are less prone to being stuck in local optima's and hence it performs much better for the travelling salesman problem. Thus, the best algorithm and hyper parameters for this case are:

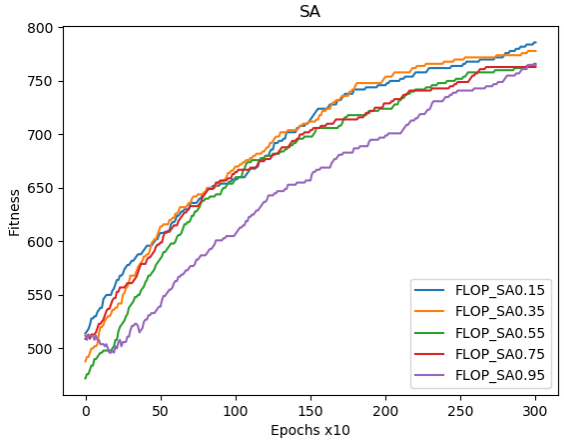
Algorithm	Genetic Algorithm
Hyper parameters	Population=100, Mate=50, Mutation rate=10
Highest Fitness	0.109

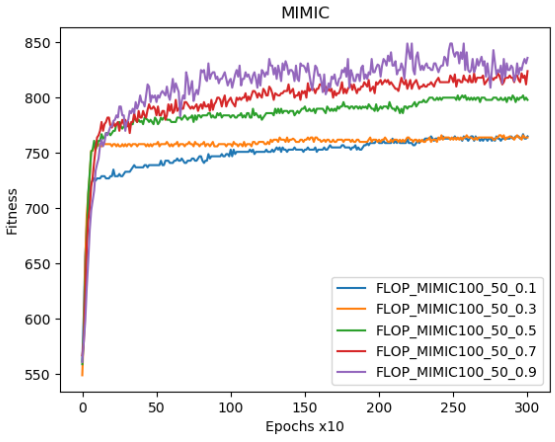
Flipflop Problem

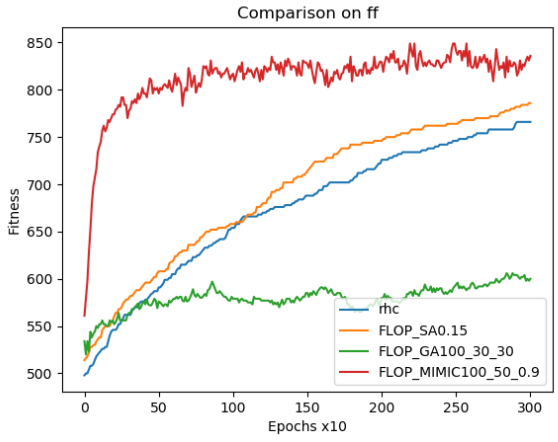
This is a bit string problem where the evaluation function counts the number of ordinating bits until it finds a consecutive pair of non-alternating bits. The fitness function can be improved by alternating a single bit in every iteration and so algorithms like MIMIC, randomized hill climbing and simulated annealing perform better here. Genetic algorithms can eventually find the solution but might not be suited well for this problem as if a parent is split close to the start of the bit string causing a consecutive pair of non-alternating bits that can lead to a lower value of fitness function for the offspring.

Like the traveling salesman problem, we will first tune the hyper parameters for MIMIC, simulated annealing and genetic algorithms to find the best model to solve this optimization problem. Randomized Hill climbing is used as it is. The graphs given below shows the performance for these models in terms of the fitness function for the flip flop problem and the hyper parameters of the models are in the legend of the graph and are the same as in the traveling salesman problem.









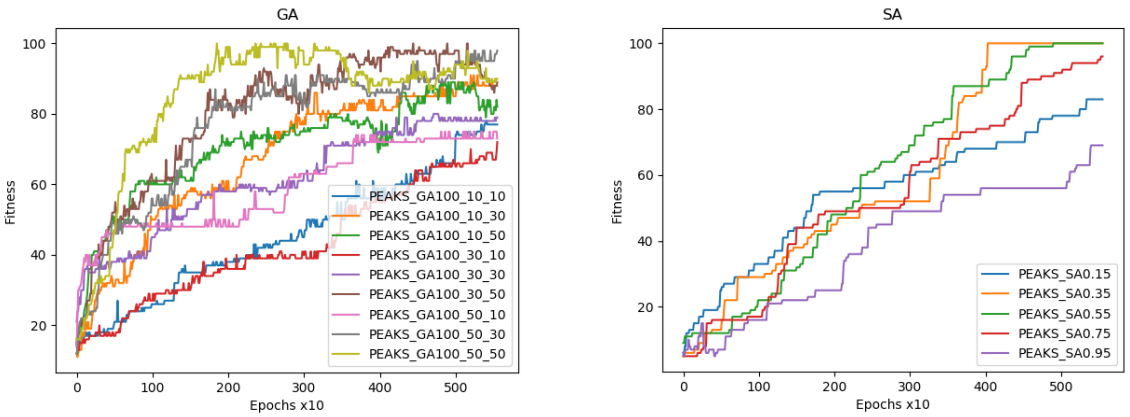
From the graphs we can see that simulated annealing and randomized hill climbing have similar performances, with SA having a better performance later on with 0.15 as the cooling component. Here a smaller value of the cooling component works well as there is only one direction in the feature space that improves the solution (alternating the next same bit) and a genetic algorithm with population size 100, mating parameter 30 and mutation rate 30 works the best in all GA algorithms but overall it works as worse amongst the four, mainly because of the same reason as to why lower value of cooling component in SA works better, that is it does not require random crossovers.

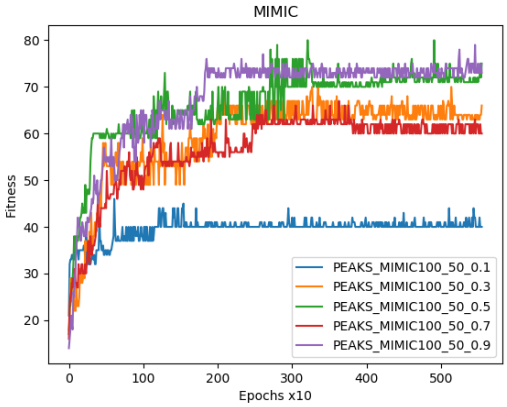
Most importantly, MIMIC algorithm works the best amongst all four algorithms in less number of iterations (around 50) as it as a high fitness function and it gets to the optima using solution space, which in this case is only one direction as mentioned earlier. However, the major drawback is that MIMIC algorithm takes **3308 seconds** to perform while simulated annealing takes **3 seconds**, this is because each iteration of MIMIC evaluates the whole solution space. Thus, the best algorithm and hyper parameters for this case are:

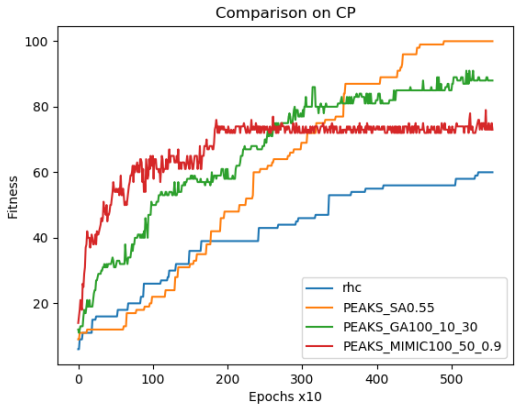
Algorithm	MIMIC Algorithm
Hyper parameters	Samples=100, Keep=50, Epsilon=0.9
Highest Fitness	836

Continuous Peak Problem

The problem set contains multiple local optima and thus the goal is to reach the global optima. The problem is simpler compared to the other two problems, however, it highlights differences between the four optimization algorithms that are being considered. The major application of this problem is in the surface optimization and topography analysis. The simplest solution can be to process ach data input point and find out the optimum but if we want to avoid it and run our optimization algorithms, their performances become quite evident. This becomes difficult as there is a good chance of an algorithm being trapped into a local optimum.







All the algorithms except randomized hill climbing are measured for their different hyper parameter values. In case of genetic algorithms, it is observed that optimizers with low mutation rate operate poorly, probably because the optimizer ends up moving for the local optima as its not able to randomize as much as required. For MIMIC optimizers, the fitness initially increases, giving a promising output but in all hyper parameters it stabilizes before reaching a global optimum as it gets stuck on a local optima and is not able to identify that it is in fact at a local optima and can reach a better state. In case of simulated annealing, optimizers with a medium cooling component performs better overall compared to a very high or a very low cooling component, again the reason behind this is probably because for low cooling component, optimizer just travels around the local optima and for very high it keeps jumping and reaches another local optimum, this is because there are many local optima's present in the solution space.

When comparing between best versions of different algorithms for 5500 iterations, we find very interesting insights. Firstly, as expected randomized hill climbing performs poorly, however it keeps learning with increasing iterations. Secondly, MIMIC initially provides the best and most promising result (less iterations and more time) because it learns about the structure of solution space and updates the threshold of fitness functions but ultimately, it keeps moving in the local optima and fails to reach the global optimum. Similarly, genetic algorithm is also promising at the start and keeps increasing the fitness with number of iterations, contrary to MIMIC though, it does not get trapped in the local optima and keeps moving to a better optimum, mainly because of the combination of crossover and mutation. However, after 3500 iterations the rate of increasing fitness reduces drastically while simulated annealing still increases its fitness function at a steady pace and is more successful than genetic algorithm in reaching global optimum. The reason behind this is that simulated annealing takes chances by going towards a less optimum state depending on the probability function with cooling component, thus even if it is an optimum, the algorithm is allowed to take chances in order to find a better solution.

Algorithm	Simulated Annealing Algorithm
Hyper parameters	Cooling Component = 0.55
Highest Fitness	100

Conclusion

The performance, advantages and drawbacks of the four randomized optimization algorithms that are discussed before have been summarized in the table below:

Algorithm	Relative Computation Speed (based on log files)	Hyper Parameters	Comments
Genetic Algorithm	Medium	Population, Mate, Mutate	-Much better at searching different parts of the solution space -Not likely to get stuck in local optima's -Can perform poorly if ordering is important in the population (flipflop problem)
Mutual Information Maximizing Input Clustering Algorithm	Slow	Samples, Keep, Epsilon	-Computationally costly -Performs well for complex structure of solution space -Reaches a better fitness value in less number of iterations
Randomized Hill Climbing Algorithm	High	None	-Keeps learning with increasing number of iterations -Simple to run and speed is very fast -Prone to getting stuck in local optima
Simulated Annealing Algorithm	High	Cooling Exponent (Temperature)	-Less likely to get stuck in local optima than randomized hill climbing -Searches a larger region of the feature space than randomized hill climbing and fast computational speed -Works well with continuous values

References

- Simulated Annealing Wikipedia: https://en.wikipedia.org/wiki/Simulated_annealing
- Code Source: JonathanTay Github: https://github.com/JonathanTay/CS-7641-assignment-2
- Dataset source: www.kaggle.com/