Randomized Optimization
CS 7641 Machine Learning Assignment 2
Siddharth Agarwal, GTID: sagarwal311

Section 1: Implementing three local random search algorithms to find good weights of Neural Networks

Dataset: The dataset that have been used in this set is the same dataset that have been used in the assignment 1. The dataset has been taken from UCI Machine Learning repository. It is a classification problem of predicting forest cover type from cartographic variables. The dataset include four wilderness areas located in Roosevelt National Forest of Northern Colorado. All necessary details are provided in the README.

There was no cleaning and preprocessing required as before and 5K instances were randomly sampled from total half a million instances for running the experiments. Again as stated in the Assignment 1, Principal Component Analysis is used to reduce the dimensions of the dataset (so 54 feature vector was reduced to 5 feature vector to retain 98% of the variance). The dataset is split into two 70-30% ratio for training and testing purposes.

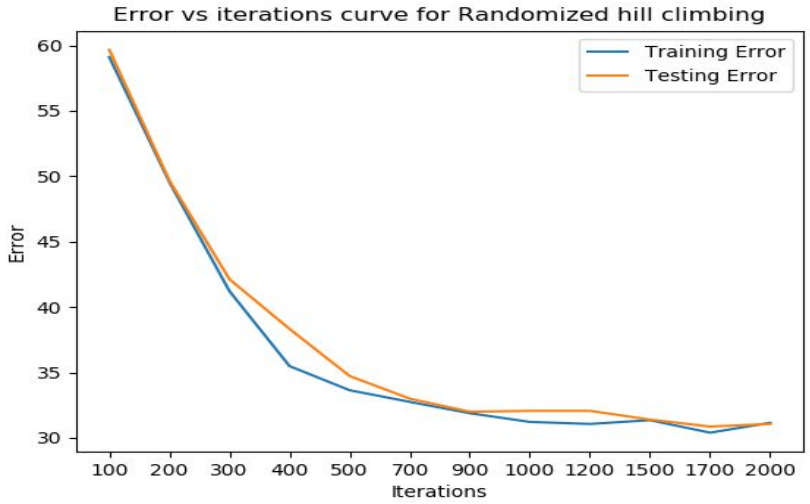
For the purposes of implementing optimization algorithms to find the weights of the Neural Network, $ABAGAIL$[1] library has been used throughout the paper. Now each subsequent section describes the optimization algorithms and then provides corresponding obtained results with the analysis.

All randomized algorithms are used to find weights of neural network of 5 perceptrons in input layer, 10 perceptrons in hidden layers, and 7 perceptrons in output layers.

1.1) Randomized Hill Climbing

Randomized Hill Climbing algorithm is a random restart version of hill climbing algorithm where it first takes a random point and then moves that point to the next position among the neighborhood points by comparing the fitness values. Since, hill climbing algorithms can get stuck in local optima when they encounter a peak, randomized hill climbing algorithm in such case restarts by again taking a random point and moving towards best fitness value point. Plot below shows training and testing error versus iterations curve for Randomized hill climbing algorithm for finding the weights of the Neural Network.

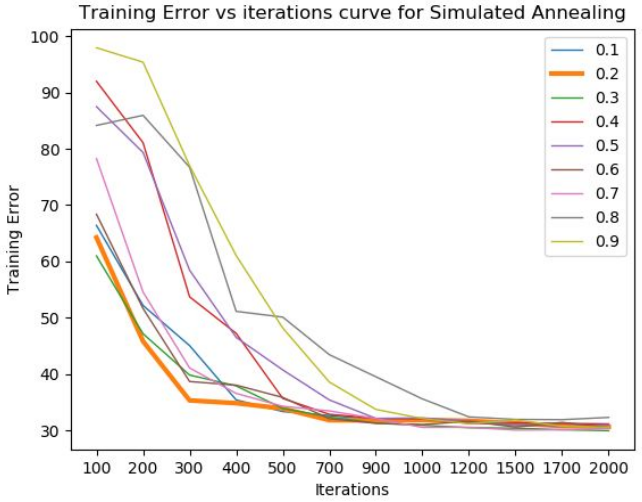
Analysis: It can be seen that training and testing error is almost similar. Algorithm converges at around 1700 iterations. As we train more, that is with more iterations the training error decreases. At around 2000 iterations best training error is 31.156, and testing error is 31.067. Time taken for the algorithm to train for 2000 iterations is 21.647 seconds while testing takes 0.002 seconds.

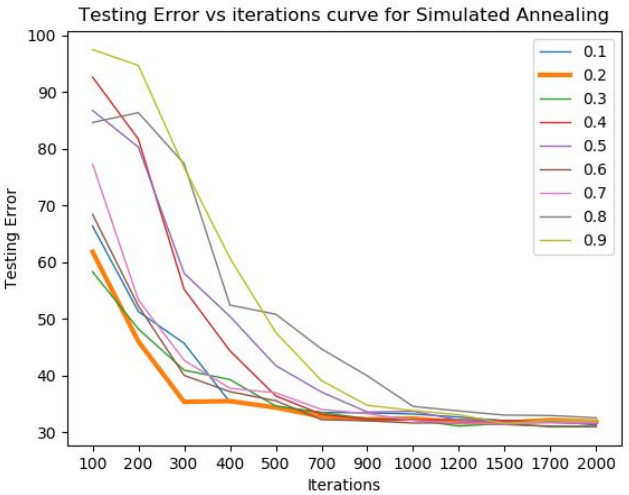


1.2) Simulated Annealing

Simulated Annealing is again a hill climbing algorithm. It is a probabilistic optimization algorithm to find global optimal point. It is often used when search space is discrete. The analogy is derived from annealing metallurgy where heating and controlled cooling is performed to increase the size of crystals and reduce the defects. In this algorithm, at each step based on probability values it is decided if the particle is moved to next particle up the hill or down the hill. Probability is decided based on temperature value, the algorithm starts with high temperature and then temperature is decreased at every step following some annealing schedule provided by the user. If the temperature is high then the algorithm would go more down hill that is going towards more generalization and reducing the chance of being stuck to a local optima. Whereas if the temperature is low, it would always go up the hill, following the neighbor with best fitness values thus increases the chance of being stuck to local optima.

Following two plots shows training error vs iterations and testing error vs iterations curve for simulated annealing algorithm for finding the weights of neural network. Curves are varied with different annealing schedules between 0.1 and 0.9. Number of iterations are varied between 100 and 2000.





Analysis: It can be seen that curve with annealing schedule of 0.2 have the highest negative slope in both the graphs. That means algorithm with annealing schedule of 0.2 converges faster than the rest of the annealing schedule. This curve converges at around 1200 iterations. Reason for convergence is the algorithm used in ABAGAIL library, temperature is decreased according to the equation (temperature * = annealing schedule). Thus, if the annealing schedule is low, temperature is reduced fast and it will always climb up the hill that is choose neighbour with best fitness value and converge faster. But in such a scenario there is high possibility that the algorithm might get stuck in local optima.
The table below shows the training/testing error and computation time with varying annealing schedule for 2000 iterations.

Annealing Schedule/ Cooling exponent	Testing Error	Training Error	Test Time	Train Time
0.1	31.267	30.556	0.002	20.325
0.2	31.867	30.662	0.002	20.002
0.3	**30.933**	**29.933**	**0.002**	**20.552**
0.4	31.467	31.133	0.002	20.643
0.5	31.467	31.00	0.002	20.137
0.6	31	30.511	0.002	20.433
0.7	31.677	30.333	0.002	21.329
0.8	32.533	32.289	0.002	20.213
0.9	32.067	30.556	0.002	20.362

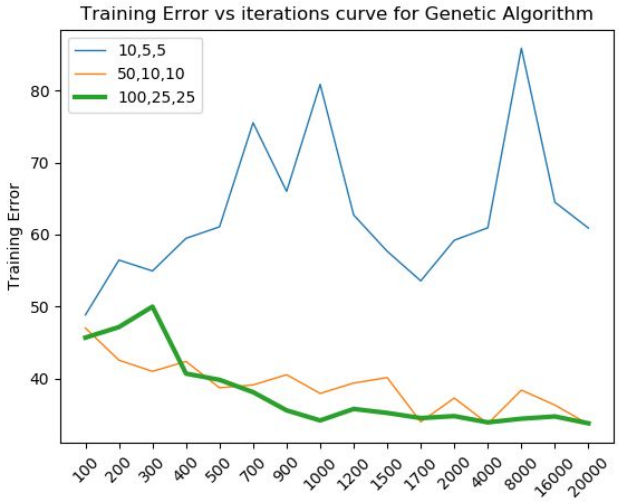
As can been seen in the table annealing schedule of 0.3 has the least testing and training error. Thus, it can be inferred that the search space of algorithm don't have a local optima because it has a least generalization error (testing error) when the temperature decreases relative fast.

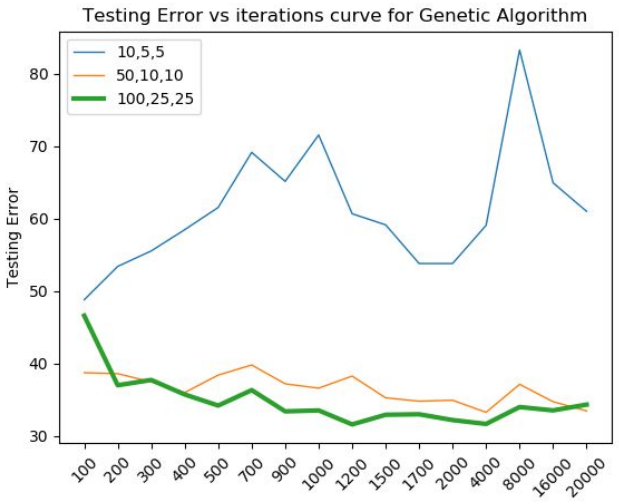
1.3) Genetic Algorithm:

Genetic algorithm is a class of biomimicry evolutionary randomized optimization algorithm. In genetic algorithms population of candidate solutions to an optimization problem is evolved towards a better solution. Each candidate solution has genetic properties and can be mutated and altered. Algorithm starts with a population of randomly generated individuals, and in each generation (iteration) fitness of each individual in the population is evaluated. In each generation portion of individuals with best fitness are breeded to form the new generation. Mutation and crossover operations are used to breed the new generation. And finally, individuals with highest fitness values are selected to be population for the next generation.

Following two plots shows training error vs iterations and testing error vs iterations curve for genetic algorithm for finding the weights of neural network. Curves are varied with different

population size, population size to mutate, and population size to mate respectively. Number of iterations are varied between 100 and 20000.





Analysis:

As it can be seen that genetic algorithm takes much larger number of iterations to converge and sometimes even after 20000 iterations it doesn't converge. The reason for this is since population is chosen randomly and only relatively small percentage of population is breeded using mutation, there's not much difference between populations of different generations. Thus, algorithm takes more number of iterations to converge. If we increase the hyperparameter - number of population to be mutated the convergence would be faster.

Also, it can also be observed that algorithm with more population gives the least error. This is obvious since greater the population, greater the probability that one individual among the population would be closer to the optimal solution.

The following table presents training and testing error, and training and testing times for different hyperparameter configurations for 20000 iterations.

Population Size	Population to mutate	Population to mate	Testing Error	Training Error	Testing Time	Training Time
10	5	5	61	60.889	0.002	1289.668
50	**10**	**10**	**33.467**	**33.644**	**0.003**	**2902.64**
100	25	25	34.333	33.756	0.003	6724.117

From the table it can be observed that training time is much greater than simulated annealing or randomized hill climbing algorithm. Reason for this is in Genetic algorithm fitness function is evaluated for each individual of the population and their children in each generation, so as population increases training time increases. Secondly, testing error for population size of 100 is slightly more that population size of 50. This means that algorithm

with population size of 100 is providing weights that is overfitting on the training data. Since, this is a randomized algorithm it might change as well if we train the network again.

Section 2: Optimization problems using four optimization algorithms

In this section three optimization problems are selected: Four Peaks, Travelling salesman, and Continuous Peaks. And then these optimization problems are solved using four optimization algorithms : Randomized Hill Climbing, Simulated Annealing, Genetic Algorithm, and MIMIC highlighting the relative properties of each algorithm.

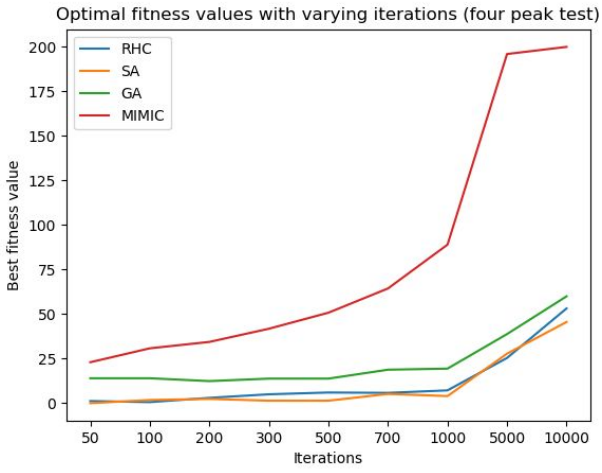
Methodology:
Since, the algorithms are randomized five test runs are performed for each algorithm for different iterations and optimal fitness value and computation time is averaged over each test run. Best performing algorithm for each optimization problem is chosen based on both fitness values and total computation time.

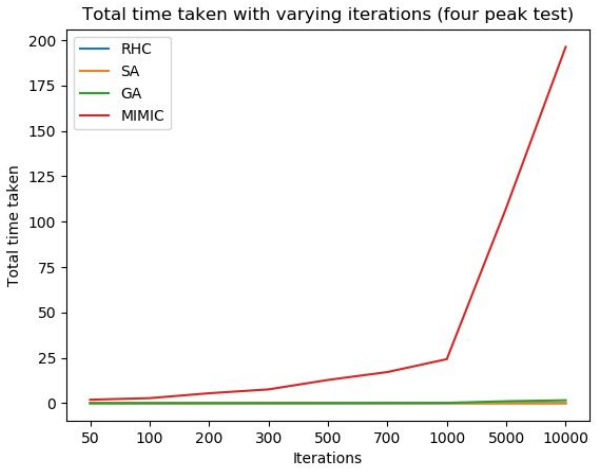
2.1) Four Peaks

Four peaks problem is an optimization problem that tries to find global maxima for the evaluation function given as $f(\overline{X},\ T)\ =\ max[tail(0,\overline{X}),\ head(1,\overline{X})] + R(\overline{X},T)$ where $tail(b,\overline{X})\ =\ number\ of\ trailing\ b's\ in\ \overline{X}$, $head(b,\overline{X})\ =\ number\ of\ leading\ b's\ in\ \overline{X}$ and $R(\overline{X},T)\ =\ N\ if\ tail(0,\overline{X})\ >\ T\ and\ head(1,\overline{X})\ >\ T\ and\ 0\ otherwise.$ This problem has two global maxima and one suboptimal local maxima.

Parameters used in ABAGAIL: N = 200, T = 40

Best fitness value vs iteration curve and total computation time (seconds) vs iteration curve for four optimization algorithms are shown below.





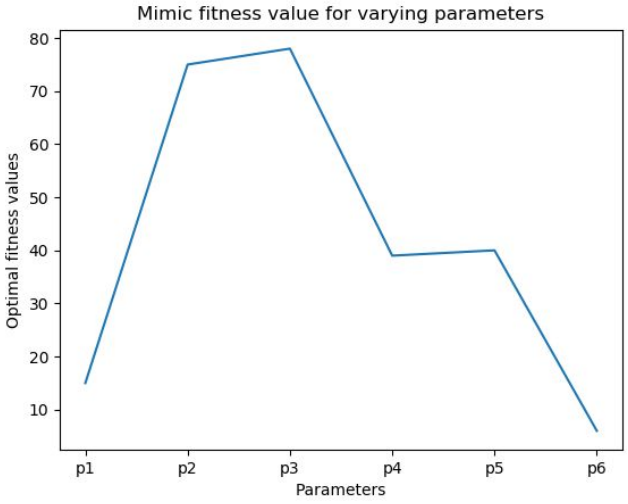
Analysis: It can be seen that MIMIC algorithm consistently maximizes the fitness function among all the optimization algorithms for all the iterations. It reaches the maximum value in (1/50th) of the iterations required for the second best algorithm Genetic Algorithm. Although

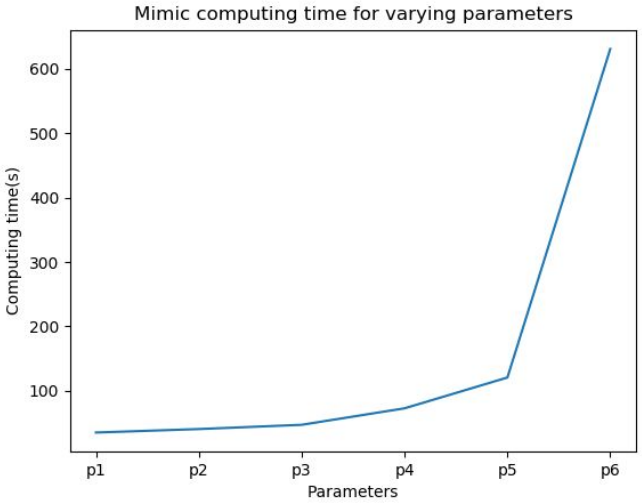
it returns maximum fitness function value among the three algorithms, it takes the most computation time. MIMIC is more reliable and better in this case as it tries to find the hidden underlying structure about the optima by computing second order statistics and sampling from conditional probability distributions that are consistent with the second order statistics.

The plots shown below show optimal fitness value and computing time for MIMIC algorithm with varying parameter values for fixed iteration of 2000.
Parameter values p1, p2, p3, p4, p5, and p6 are:

	p1	**p2**	**p3**	**p4**	**p5**	**p6**
Samples	10	50	100	200	500	1000
Random	5	10	20	50	100	500

Where Samples are the number of samples to take in each iteration and Random is the number random samples in that population of samples. It can be seen that p3 has the highest optimal fitness value which means approximately MIMIC would perform better if there are approximately 20% random samples in the total samples in each iteration. And as total number of samples increases time taken to perform the optimization also increases drastically.





2.2) Travelling Salesman Problem

Travelling Salesman is a NP hard problem which tries to answer the question that given the list of cities and distance between each pair of cities what is the shortest route that visits each city and in the end return back to the original city. Travelling salesman problem can be modelled as undirected weighted graph problem such that cities are graph vertices and route between cities are the edges and distance between cities are the edge weights. It can also be formulated as integer linear program.

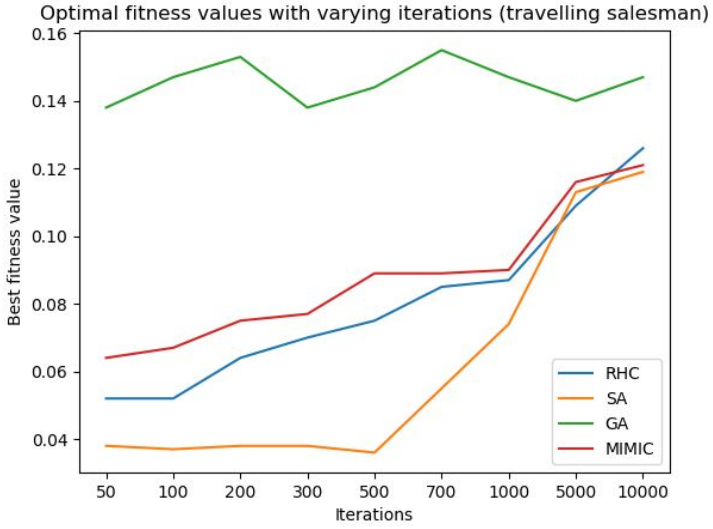
$x_{ij} = 1$ *when path goes to city i to city j and* 0 *otherwise*

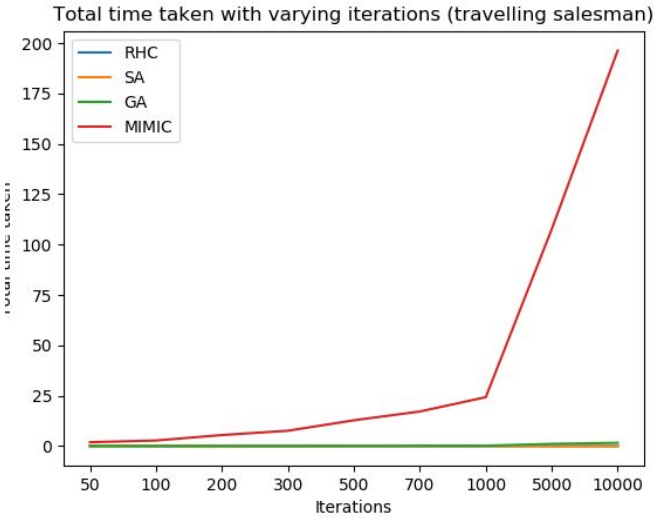
Let u_i *be dummy variable and* c_{ij} *be distance between city i to city j*

Then Travelling salesman problem can be written as $min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij}$ subject to constraint that $0 \leq x_{ij} \leq 1, \quad \sum_{i=1, i \neq j}^{n} x_{ij} = 1, \quad \sum_{j=1, j \neq i}^{n} x_{ij} = 1$ *and* $u_i - u_j + n x_{ij} \leq n - 1$

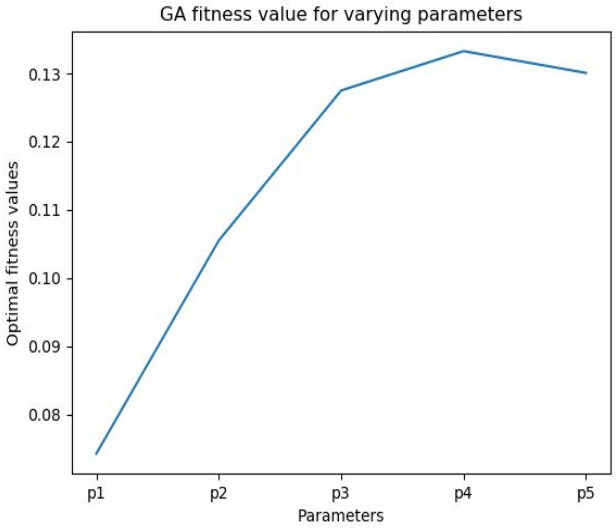
Best fitness value vs iteration curve and total computation time (seconds) vs iteration curve for four optimization algorithms are shown below.

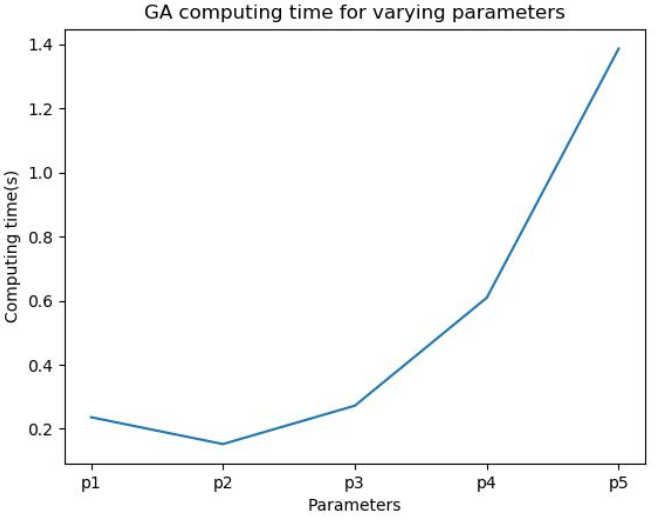
Parameter used in ABAGAIL : N = 50





As can be seen that Genetic Algorithm has the highest optimal fitness value as compared to different algorithms. It reaches to a fitness value of approximately 0.14 that other algorithms reach at around 10000 iterations, so it is considerable faster than other algorithms. It is considerably faster than MIMIC algorithm. Thus, Genetic algorithm performs best on Travelling Salesman Problem both in terms of fitness value and computation time.





The plots above shows Genetic Algorithm's performance on Travelling salesman problem with varying parameter values and fixed iterations. Total 2000 iterations are taken for optimization of Travelling salesman problem with varying parameters p1, p2, p3, p4 and p5.

The values of these parameters are shown in the table below.

	p1	**p2**	**p3**	**p4**	**p5**
Population	10	20	50	100	200
To Mutate	5	10	25	50	100
Mate	5	10	25	50	100

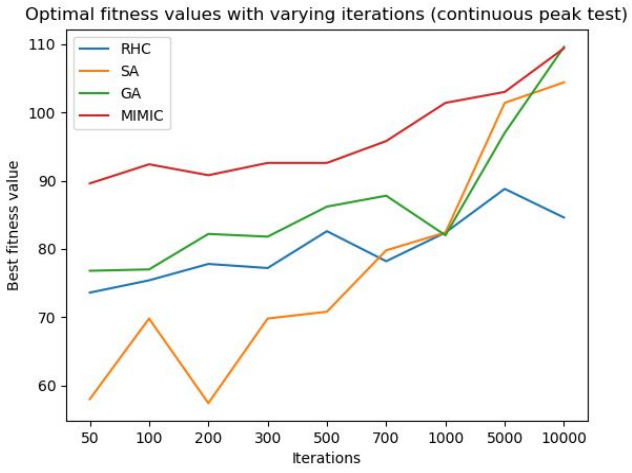
Where Population is the total population size in each generation. Mate is the crossover number representing the mating between individuals. Mutate is the mutation that causes random modifications. In general greater the population better would be the performance of the Genetic Algorithm. In this case, Genetic Algorithm performs best when mutate and crossover points are approximately 50% of the total population.

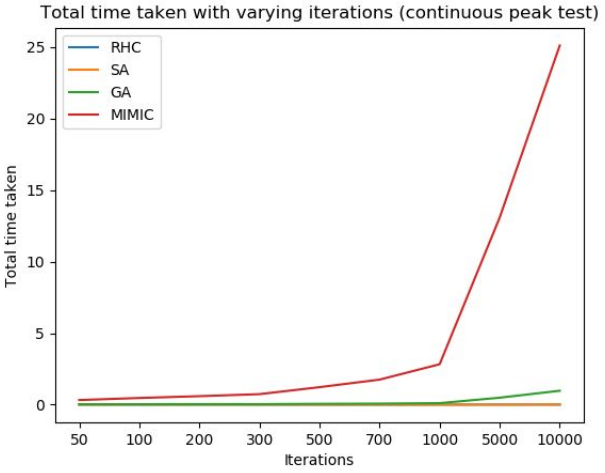
2.3) Continuous Peak Problem

Continuous Peak problem is one of the type of peak problems. In this problem rather than forcing 0's and 1's at the opposite end of string solution, they are allowed to form anywhere in the string. Reward is given when there are greater than T contiguous bits set to 0 and greater than T contiguous bits set to 1. It tries to find the highest peak with respect to local peaks (local minima).

Parameters used in ABAGAIL: N = 60, T = 6

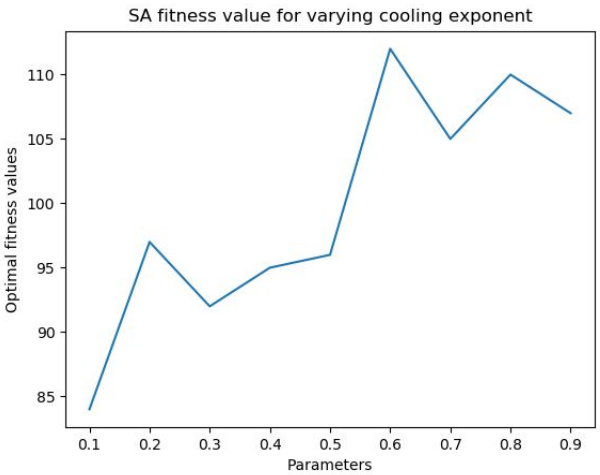
Best fitness value vs iteration curve and total computation time (seconds) vs iteration curve for four optimization algorithms are shown below.

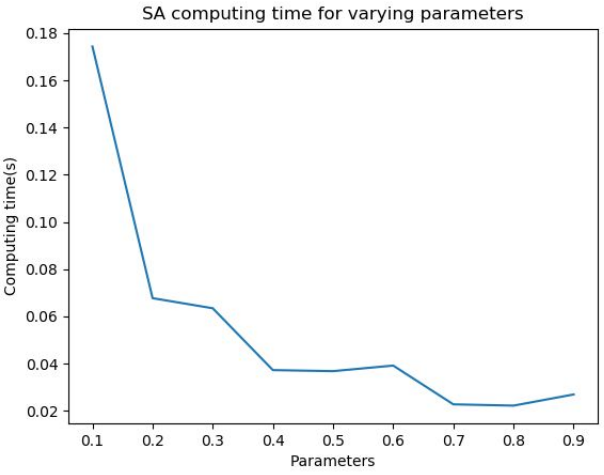




It can be seen that although MIMIC and GA do slightly better than Simulated Annealing in terms of optimal fitness values obtained. They perform considerably bad when we consider their total computation time. Simulated Annealing has lower total time taken than MIMIC and GA. Thus, if we consider the tradeoff between best fitness values and total time taken then we can conclude that Simulated Annealing performs better than other three optimization algorithms for Continuous Peaks Problem.

The following plots shows the Simulated Annealing performance with varying hyperparameter setting taking a fixed number of iterations. Both optimal fitness values and total computation time is plotted.





Parameter that is varied in these plots for accessing the performance of Simulated annealing is cooling exponent or annealing schedule. In ABAGAIL library the equation to decrease the temperature is (t* =annealing schedule). As expected optimal fitness is obtained when annealing schedule is at approximately 0.7 that is in the higher side. If the annealing

schedule is low, temperature will decrease fast and next solution chosen will always up the hill in which it may converge faster but it may get stuck in local optima. Whereas as in this case when the annealing schedule is higher it will explore the search space completely and in the end will reach to global optima.

Conclusions

We can draw following conclusions from the above analysis:

1. In the first section a neural network was trained using three optimization algorithms: Randomized Hill Climbing, Simulated Annealing and Genetic Algorithm.
2. It can be concluded that Gradient Descent using backpropagation performs better than these three algorithms.
3. In terms of total time taken, Randomized Hill Climbing, and Simulated annealing are faster algorithms whereas Genetic Algorithm and MIMIC are slower algorithms. The reason for this is because Genetic Algorithm and MIMIC consider a population of possible solutions whereas other two just work with one solution at a time.
4. Randomized Hill climbing and Simulated Annealing are good for solving simple problems whereas Genetic Algorithm and MIMIC are designed to solve complex problems with structure in the search space.
5. Randomized Hill Climbing and Simulated Annealing can get stuck in local optima if hyperparameters are not set properly. There is less probability of Genetic Algorithm and MIMIC to get stuck in local optima.
6. In the second section three optimization problems were solved: Four Peak test, Travelling Salesman Problem, and Continuous Peak Problem.
7. Each optimization problem showed relative advantage of one particular optimization algorithm: MIMIC performed best on four peak test problem, Genetic Algorithm on Travelling Salesman Problem, and Simulated Annealing on Continuous Peak Problem.
8. Tradeoff between optimal fitness obtained and computing time was used for finding the best performing optimization algorithm.

References

[1] ABAGAIL Library: https://github.com/pushkar/ABAGAIL