Assignment 2: Randomized Optimization

Shailesh Tappe: stappe3@gatech.edu

Note: ***I'm retaking course for this semester, I'm using most part of my previous semester (Spring 2020) report in this report. This time I'm re-exploring same datasets that I used previously, and trying to analyze more for this semester.***

Objective: The objective of the assignment is to analyze performance of four random optimization algorithms i.e. randomized hill climbing (RHC), simulated annealing (SA), genetic algorithm (GA) and MIMIC on various cross functions. In second experiment, random hill climbing, simulated annealing, genetic algorithms performance is analyzed to determine weights of artificial neural networks with "Wine Quality" datasets used in assignment 1 and compared with back propagation.

Optimization Algorithms:

1. **Randomized Hill Climbing (RHC):** In RHC optimization starts with random selection of starting point for search. With this input point in function search space, it then looks for best value for function by choosing a neighbor with improved function value i.e. analogous to hill climbing. Repeat the algorithm until function value does not improve i.e. repeat until algorithm gets optimal value for the function in the polynomial search space. With hill climbing approach there is possibility of search getting stuck in local optima, as it is dependent of where search gets started. Therefore, to get optimal value for function in search space and minimize possibility of getting stuck in local optima, RHC is helpful advancement to hill climbing algorithm. With RHC, hill climbing algorithm again restarts algorithm to widely distributed random point in input space, until it hopes to find global optima for the search space.
2. **Simulated Annealing (SA):** Unlike RHC, where algorithm only looks for improved value i.e. exploiting search space to find optimal value, simulated annealing (SA) explores in neighborhood search space beyond just finding maximum value. Analogous to metallurgical annealing, where in metal is heated and cooled off to improve or strengthen metal, simulated annealing (SA) explores new points in neighborhood and decide probabilistically if current state is worth exploring.
   So iteratively in sets of search space, randomly take input point and look in neighborhood. If fitness value of next point in neighborhood is greater than fitness value of current point then algorithm uses it to find optimal value; otherwise take exponential fraction of difference between fitness functions and temperature to look for probability of function. So, with very high temperature value, probability of accepting fitness value is high, even if value is suboptimal to current fitness value. Likewise, for very low temperature probability of accepting fitness value is low and search algorithm would not look for suboptimal low value and go only for hill climbing search. In other words, if value of temperature is near zero, algorithm looks like hill climbing or if temperature value is leaning towards infinity algorithm does random walk in neighborhood space, instead of looking for optimal value.
   With constant annealing of temperature with slow cooling, algorithm tends to find global optima in search space.
3. **Genetic Algorithms:** Genetic Algorithm (GA) optimization technique is inspired by biological natural evolution such as mutation, crossover and generation. GA computed fitness function for all population by mutating i.e. by local search in neighborhood, crossover from sample points of population with set of different attributes combines together to hopefully get better information, and generation where iteratively generate new population set with hope to get better population. Crossover is what makes GA is useful as it generates new set of population hopefully with better fitness, and deviates from parallelize randomize restarts as this generally behaves.

4. **Mutual Information Maximizing Input Clustering (MIMIC):** All these 3 randomized optimization algorithms are primitive and don't learn anything about continuous space they are searching over and more amnestic about their search history. With MIMIC the concept is to define model from probability distribution, and search through input space to create refined model. This refined model over time to get structure of search space with optimal points in structure.

Experimenting and Analyzing Optimization Problems: In this section 3 optimization problems were evaluated to consider performance of different optimization problems discussed above. Experiments were carried out using python 3 and mlrose library for analyzing optimization problems along with sklearn. NumPy and matplotlib libraries.

To understand how these optimization algorithms work, an experiment was carried out with these input parameters for all 3 optimization problems

First set of experiment was to analyze fitness, time, number of evaluations vs problem size using these parameters

Parameters for Fitness/Time/Number for evaluations VS Problem size				
Optimization Algorithm	**max iterations**	**max attempt**	**problem size**	**input parameters**
RHC	20000	200	20-80	
SA	20000	200	20-80	{ 'decay_schedule' : 'geomatric decay', 'decay_rate' : '0.99', 'initial_temp' : '1.0', 'min_temp' : '0.001' }
GA	20000	200	20-80	{'population_size' : '200', 'mutation_probability' : '0.1' }
MIMIC	20000	200	20-80	{'population_size' : '200', keep_size : '0.2' }

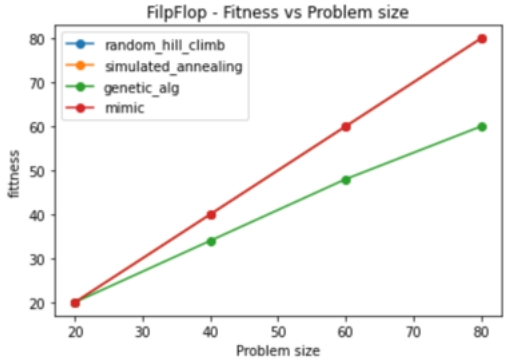
Second set of experiment is to analyze fitness vs iteration using these parameters.

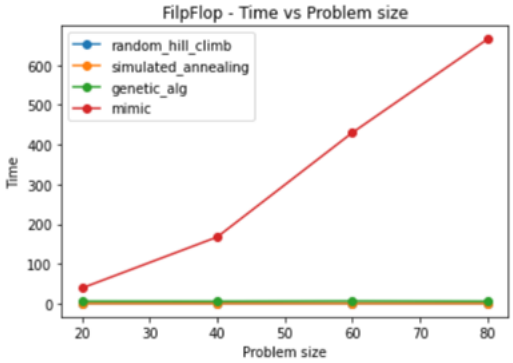
Parameters for Fitness VS Iterations				
Optimization Algorithm	**max iterations**	**max attempt**	**length**	**input parameters**
RHC	5000	5000	100	
SA	5000	5000	100	{ 'decay_schedule' : 'geomatric decay', 'decay_rate' : '0.99', 'initial_temp' : '1.0', 'min_temp' : '0.001' }
GA	5000	5000	100	{'population_size' : '200', 'mutation_probability' : '0.1' }
MIMIC	5000	5000	100	{'population_size' : '200', keep_size : '0.2' }

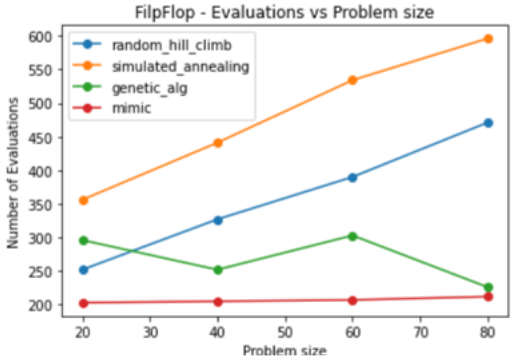
Temperature scheduling for simulated annealing was done using geomatric decay to cool temperature gematrically. Cooling factor in SA is deterministic in optimizing algorithm. If cooling is too fast then it affects ability to find global optima and more probability to get stuck in local optima, on the contrary if cooling is too slow, then probability of algorithm to find global optima is high, but computationally expensive. For example, logarithmic decay converges to global optima, but temperature cools slowly and hence take longer time to compute; whereas geomatric cooling is comparatively faster to logarithmic cooling but probabilistically lower in finding global optima.

For genetic algorithm set of population mutate together to get better population set. That makes setting optimal mutation probability to converge GA algorithm to optimal solution. In other words, to get better crossover population mutation helps in getting better solution.

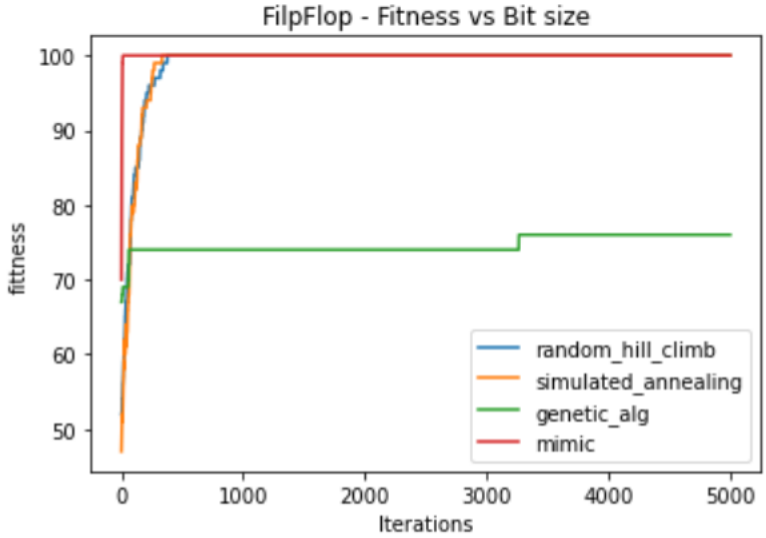
1. **Flip flop problem:** Flip flop is a problem where bits are flipped to maximum bits by counting bits from input bit string. That means fitness bits are alternative bits from what is in input string. As attribute of each bit in string is independent, it can have multiple local optima.







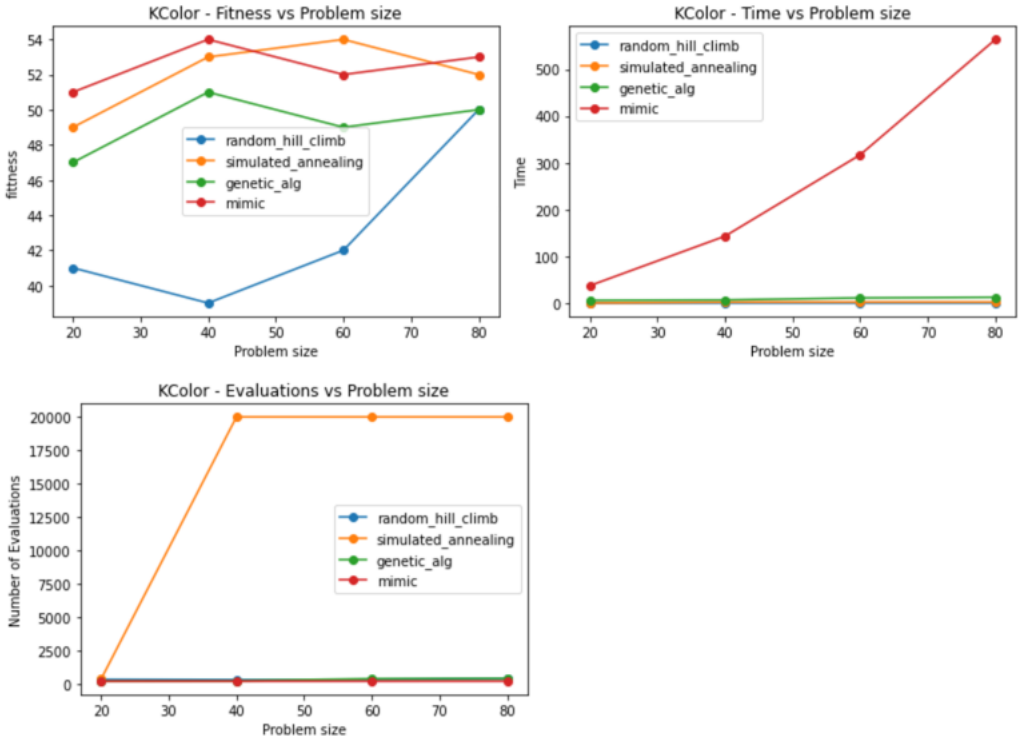
Observing fitness of algorithm with problem size, it can be observed that RA, SA and MIMIC is having better fitness compared to genetic algorithm (GA). Observing at time taken vs problem size, MIMIC is much slower to converge to find optima, as compared to other 3 algorithms i.e. RHC, SA and GA. But in terms of maximizing function with evaluations, MIMIC does much better job as it had better structural stability compared to other 3 algorithms.



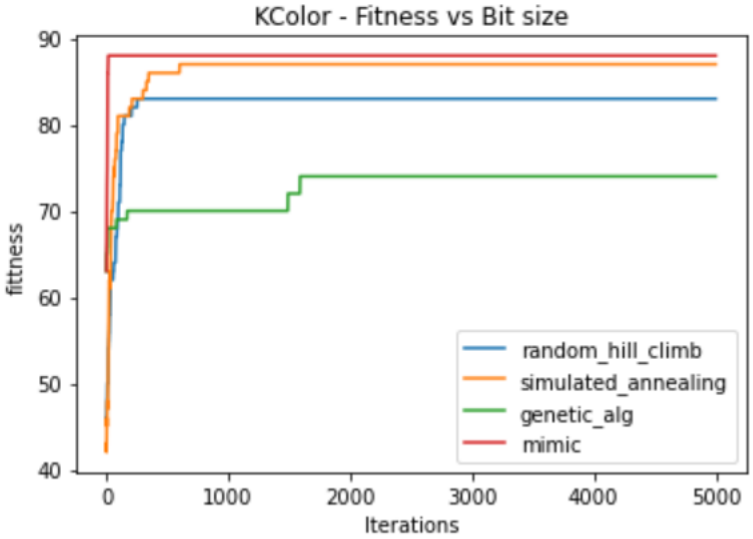
Comparing fitness (i.e. accuracy) over iterations, MIMIC is converging fast as compared to RHC, and SA. GA fitness compared to other algorithms is suboptimal and can get stuck in local optima

flip problem. Overall MIMIC is doing better with flip flop problem, as it performs better with structure, such as problem like flip flop.

2. **Max K-Coloring problem:** The objective of max K-coloring is to use k colors to color maximum number of vertices. K-coloring is NP-complete problem and it is K-colorable if it assigns k-color to other nodes and have different colors to each adjacent node.



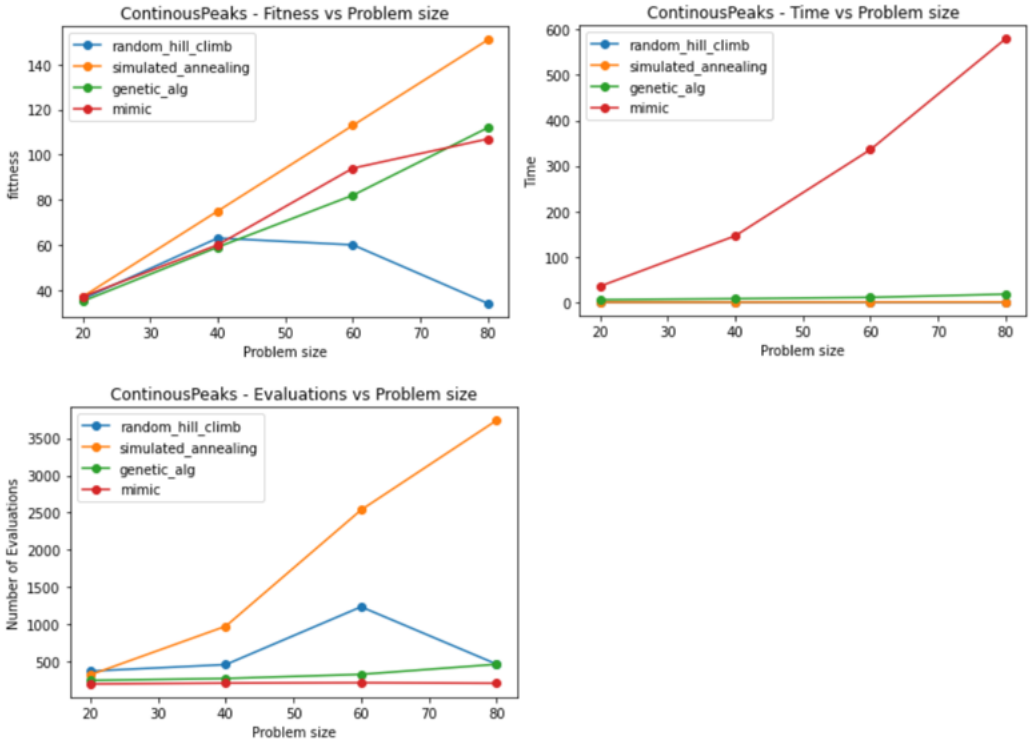
Observing fitness of algorithm with problem size, it can be observed that MIMIC and SA are having better fitness as compared to RHC and GA. Although SA is substantially less time compare to MIMIC to achieve similar level for functional fitness. In terms evaluations vs problem size, SA is inferior in performance as compare to RHC, GA and MIMIC. But overall MIMIC is better as it provides structural stability in solving the problem.



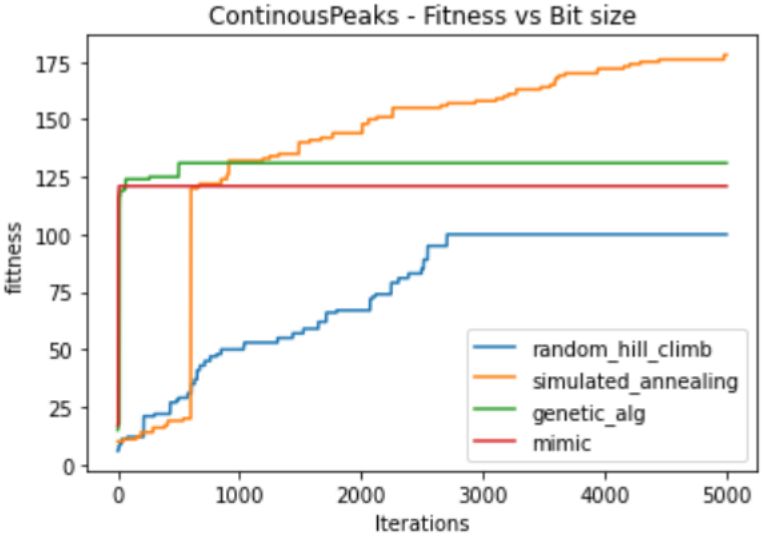
Compratively MIMIC is converging fast to get better fitness (accuracy) over the iterations, SA is also performing, but less comapre to MIMIC. RHC and GA is subpotimal, and can get stuck in

local optima for max K-coloring problem. Over MIMIC is doing better for K-coloring problem, but SA is close second. So if maintaining strucctural stability is not important factor then SA is better as it takes substentially less time to get close to global optimal solution.

3. **Continuous Peaks problem:** The continuous peaks problem is extended version of four peak problem, where with randomized vector of integers of peaks and valleys, finds optimal value. The problem can have numerous local optima in 2D space.



Observing fitness of algorithm with problem size, it can be observed that SA is performing better compare to RHC, MIMIC and GA. Also, time taken by SA is less compare to all other algorithms i.e. RHC, GA and MIMIC. In terms evaluations vs problem size, MIMIC is better comparing to other algorithms, that is because MIMIC is better structural stability as compare to other optimizers

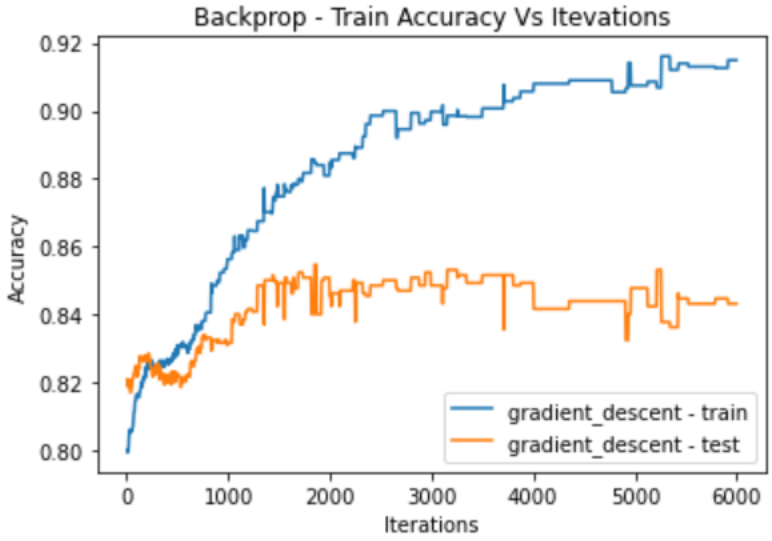


Comparing fitness (accuracy) over iterations, it is observe that SA is peforming much better comapre to other optimizers. This means to find global optimal value in problem where more local

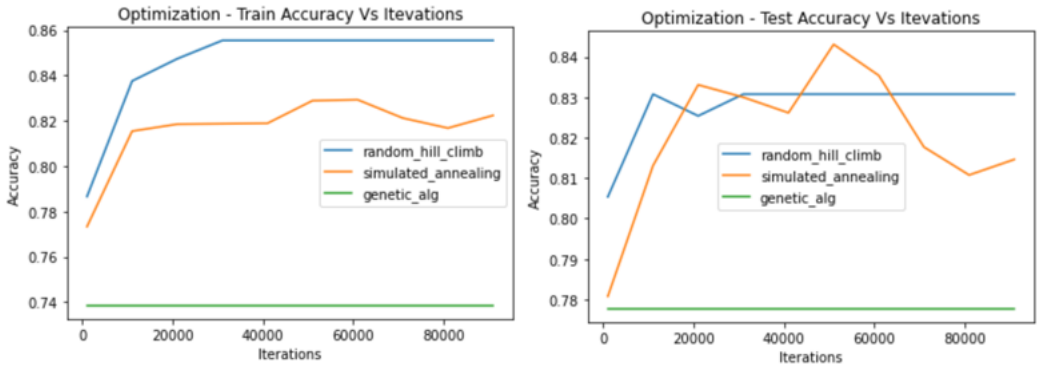
optima, as SA alogorithm heats and cools temprature and wonder freely to avoid getting stuck in local optima, as it can be observed for RHC where it got stuck in local optima and getting suboptimal fitness function.

Neural network and optimization algorithms: The purpose of this experiment is to evaluate back propagated neural network from assignment 1 with feed-forward neural network and implement it with randomized optimization algorithms random hill climbing (RHC), simulated annealing (SA), and genetic algorithm (GA) against dataset (wine quality dataset) used in assignment 1 for supervised learning.

1. **Dataset :** For this experiment wine quality dataset is used as it was in experiment 1. The wine quality datasets are variant of Portuguese 'Vinho Verde' wine, sample taken from a study performed by University of Minho in Portugal. Data contains 6497 instances and 12 attributes with classification attribute "*quality*". In preprocessing quality rating of wine is set to low quality (i.e. 0 for rating less than 7) and high quality (i.e. 1 for rating 7 or above). Dataset was randomly split with ration of 80:20 for training and testing sets.
2. **Experiments:** Experiments were carried out using python 3 and mlrose library for analyzing optimization problems along with sklearn. NumPy and matplotlib libraries.
   Same like assignment 1, "RELU" activation was used with 3 level hidden layer of 20,20,10 and learning rate of 0.0001 was used in experiment with over 100000 iterations. Training and testing accuracy against iterations was plotted to analyze performance of optimization algorithms.
3. **Analyzing backpropogation for gradient descent for neural network:** In this experiment backpropogation neural network was used to update weights in neural network. It is found that optimal number of hidden layer is 20,20,10 with learning rate of 0.0001. Accurayv VS iteration obtained by backpropogation suggest training accuracy of 0.91 and test accuracy of 0.84

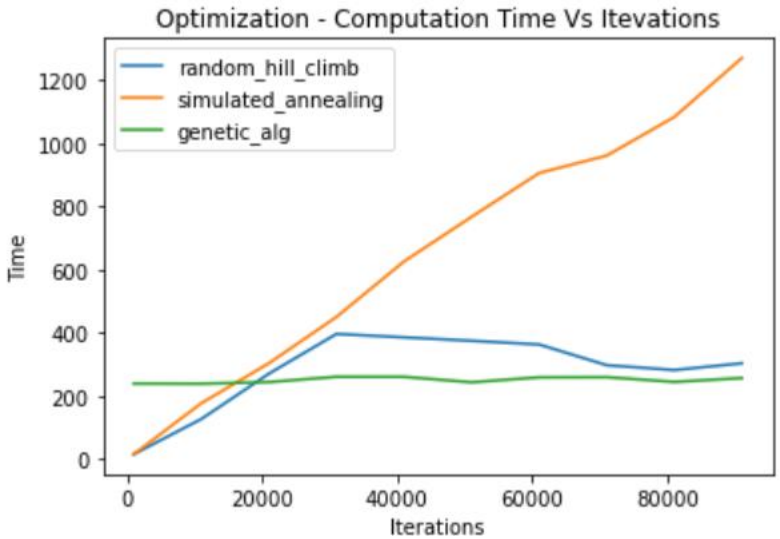


4. **Analyzing Training and Testing accuracy for optimized algorithms:** In this experiment for both training and testing data in wine quality datasets was tested for RHC, SA and GA algorithm using relu activation. An experiment used hidden layer as 20,20,10 and learning date as 0.1



Algorithm	Iterations	Train Accuracy	Test Accuracy
gradient descent	6000	0.91	0.84
RHC	80000	0.86	0.83
SA	80000	0.82	0.81
GA	80000	0.74	0.74

Obeserving performance of each algorithms it is found that both RHC and SA accuracy is better comapre to GA and converging well with training and cross validation data. Principal of RHC to determine direction of global optima by randomly starting new states in search space work better with neural network, as it works on similar objective of gradient descent to find optima. SA algorithm is also perfomring in line with RHC, as both of them are similar algorithm in the sense of randmoizing in search space, except SA have more tunning parameter of tempurature and hence works well with neural network. Performance of GA algorithm is flat with this test even with high accuracy rate of 0.74, and need to tune more with population size and mutation probability to analyze performance with different sets of hypertunning parameters.



Observing computation time with number of iterations it is found that RHC is converging faster comapred to SA and can get stuck in local optima.'GA computational time is flat across number of interation and no significant variant in computational time.

Although time matrics was not captured for backpropogation gradient descent, it is observed that time taken to by back propogation gradient descent was substentially high comapre to feed forward optimization algoriothms for the neural network problem. Also accuracy for SA and RHC

is close to backpropogration algorithm. In conclusion neural network solved with optimization alogrithms can be more effective with tunning parameters and faster as compare to neural network with back propogation gradient descent. With hyperparameter tunning, randomize optimization algorithms can be more effective in solving NN problems.
Experiment needs more time and effort to find out optimized hyperparameters for each randomize algorithm, specially for GA, as it was flat out with set of tunning parameters used in experiment.

Conclusion: Overall project experience was unique and interesting. The analysis of all 4 algorithm enhance my understanding of optimization of machine learning algorithm. Each set of algorithms have its own strengths and weakness. RHC is simplest algorithm for exhaustive search with small run time per iteration. But it can get stuck in local optima. SA is similar to RHC, but it prevents getting stuck in local optima by adjusting temperature parameter. GA is good when local optima is immense for complex problem is search space. While RHC, SA and GA work better in problem space, it lacks structure and forgets everything where it was previously available in search space. To address this structural problem, MIMIC is useful where structure is important. MIMIC converges fast with iterations, but take longer computational time per iterations.

With these experiments I was not able to successfully optimized GA for neural network. In future I intent to explore more hyperparameters and capture metrics to understand more not only GA for neural network, but explore more of other algorithms to find best optimal solution for given problem.

References:

- Tim Mitchell: book – Machine Learning
- Dr. Charles Isabell and Dr. Michael Littman: video lectures on randomized optimization
- Paulo Cortez, University of Minho, Guimarães, Portugal
- UCI Machine Learning Repository
- Openml.org
- scikit-learn library and documentation
- mlose library and documentation
- Jeremy Boner, Dr Charles Isabel, Jr. Paul Viola: MIMIC: Finding Optima by Estimating Probability Densities
- Baluja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. Technical report, Carnegie Mellon University.
- TA's during office hour and piazza forum