Randomization Optimization

Swati Prasad
sprasad33@gatech.edu
CS7641-Machine Learning

Abstract:

The purpose of this document to explore randomization optimization, first running on well-known computer science problem and then by applying them to neural network model using pima Indian diabetes data from assignment-1. Finding optimized weight in neural network is done by replacing back propagation with another optimization methods Random hill climbing, simulated annealing and genetic algorithms. After each training iteration, the weights are then updated using optimization or fitness function of the respective algorithms. For the second part, given three algorithms were further explored on computer science problems Knapsack problem, counting ones and Traveling salesman. Each problem will show which algorithm work well for that particular domain and respective analysis will be provided.

Optimization Algorithms

Optimization refers to finding the values of input in such a way that we get the best output values. In mathematical terms finding best mean maximizing or minimizing one or more objective functions, by varying input parameters. In this paper, will focus on maximizing cost function. The following algorithms are compared and implemented using ABAGAIL.

1) Randomized Hill Climbing Algorithm (RHC Algorithm):
   a) Introduction: In Randomized Hill climbing, random point is chosen. It locates local optima by moving towards moving more optimal neighbors until it reaches a peak. With random restarts, RHC randomizes its starting position to locate other local optima and selects the value with the highest value as global optimum. This works well for dataset with broad basin of global optima as likelihood of picking a value in basin is large. Opposite is when basin is narrow of global optima and there are lot of local optima, then it becomes closer to doing a full search across the problem domain.
      RHC is performed using RandomizedHillClimbing in Java (ABAGAIL).
2) Simulated Annealing:
   a) Introduction: Simulated Annealing is similar to RHC is that it is trying to exploit by continuously moving in the direction of improvement with added benefit of exploring the space. Coming from the analogy of heating and cooling metal to make it less brittle, the same idea is applied by starting with high temperature T and cooling exponent C. Initially, when temperature is high, SA moves around more, exploring the problem space. With each iteration, the step size that SA can take decreases as a function of C (cooling off). The longer we run the algorithm, the more it goes from random search to RHC. This feature of exploring gives SA the additional benefit of being able to break out of local optima.

SA is performed using SimulatedAnnealing in Java (ABAGAIL) using input temperature (1E11) and cooling rate (.95).

3) Genetic Algorithm:
   a) Introduction: Genetic algorithm is search based optimization technique based on genetic and natural selection. It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise take lifetime to solve. Initially, we have pool or a population of possible solutions to the given problem. These solutions then undergo recombination and mutation, producing new children and process is repeated over various generations. Each individual or candidate solution is assigned a fitness value (based on objective function value) and the fitter individuals are given higher chance of mate and yield fitter individuals. In this way we keep “evolving” better individuals or solutions over generations, till we reach stopping criteria.

   GA is performed using StandardGeneticAlgoriithm in JAVA (ABAGAIL) using input

   population size (200), toMate(100) , toMutate(10).

4) Mutual-Information-Maximizing Input Clustering (MIMIC):
   a) Introduction: MIMIC algorithm opposed to most optimizing algorithm remembers previous iterations and uses probability density to build structure of the solution space and finding optima. MIMIC is performed in Java (ABAGAIL) using input samples (200) and keep (20).

Implementation

Applying Randomization optimization to 3 problems

The existing cost function in GITHUB repository is used.

i) Knapsack

Knapsack problem is NP hard (at least as hard as hardest problem in non-deterministic polynomial acceptable problems) optimization problem with set of constraints.

Definition: The knapsack problem is defined as the problem with given set of items N, where each item I has corresponding weight w and value v, try to select a subset of N such that we maximize the value of V while not going over a threshold W. For analysis KnapsackTest.java is used with parameters (40 items, 4 instance of each, randomized value not greater than 50 and weight threshold of 50). 10 iterations are done for 4 algorithms and the averages of those runs were used to compare.

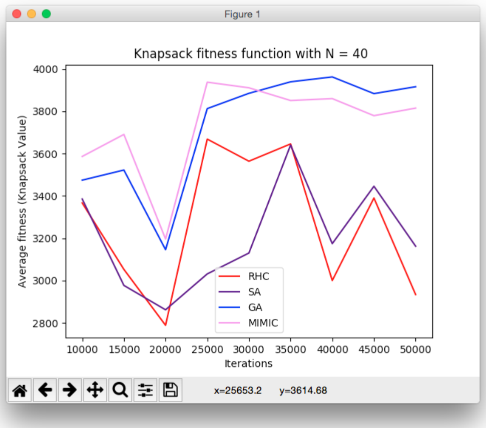
Maximize sum(v_ix_i) for I = 1 to n
subject to sum(w_ix_i) < W and x E {0,1}

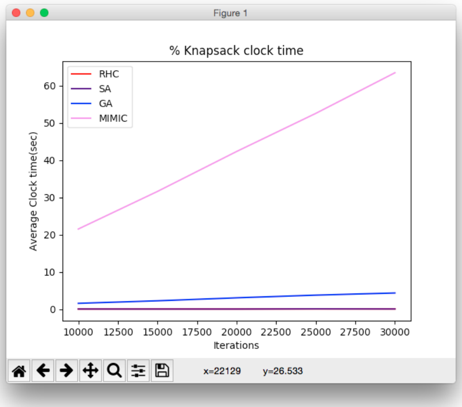
Analysis:

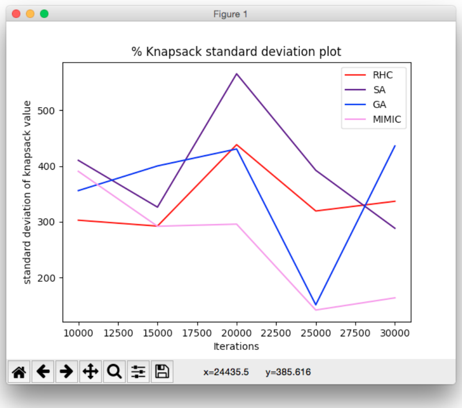
From the diagram, it is evident that MIMIC and GA do better than others. Mimic does well, where probability distribution can be captured. Mimic and GA both learned from their previous structure. Algorithms show peak performance around 25000 iterations. Lower performance is at 20000 iterations due to functions didn’t converge. We can confirm this by looking at standard deviation curve, we see standard deviation is high at 20000 iterations and least deviation at 25000 iterations where functions has displayed

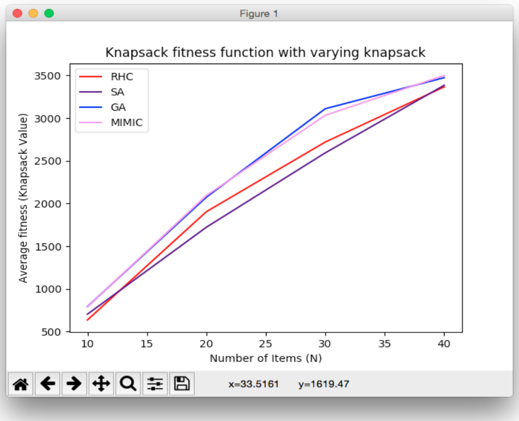
optimization and converged. It has been observed that initially, GA and MIMIC algorithm progresses fast around 25000 iterations with better solution coming in every few iterations except at 20000 iterations, but this tends to saturate in the larger iterations where improvement is very small.
Mimic is most time consuming due to its computation intensive nature. It takes times to create probability distribution function.
We see for lower items all algorithms are comparative. Lower items has wider basin of optimization. RHC and GA started well, but as number of items are increasing or problem is getting more complex they start lagging.
Mimic and GA may performs better where scope of learning from their previous space search instead of randomly wandering.

It is faster and more efficient compared to the traditional methods. It always answer to the problem, which gets better over time. Useful when search space is very large and there are large number of parameters are involved.

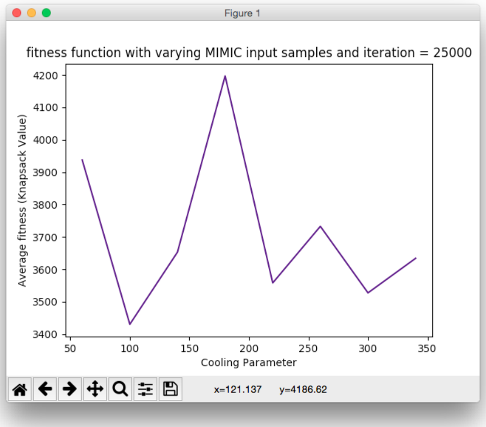


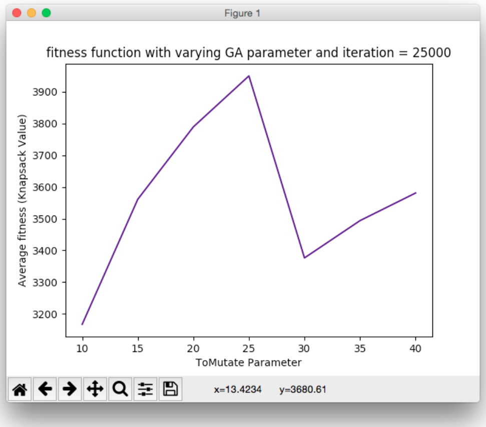






Finding Hyper parametrs for MIMIC and GA: After selecting the best optimization algorithm, grid search for best hyper parameters. We had got MIMIC has peak at 25000 iterations. For 25000 iterations, input parameter is varied and found at (180, 20) reaches maximum with maximum value of 4197.12. Similary for GA, we see hyper parameter toMutate =25 with population = 200 and toMate(150).



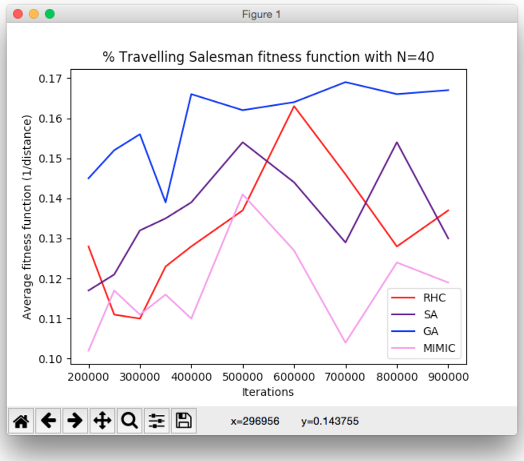


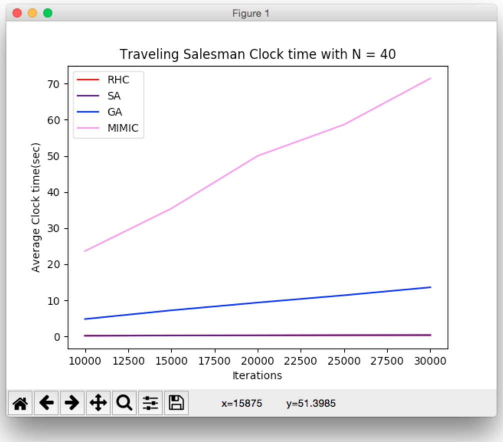
ii) Travelling Salesman Problem (TSP)

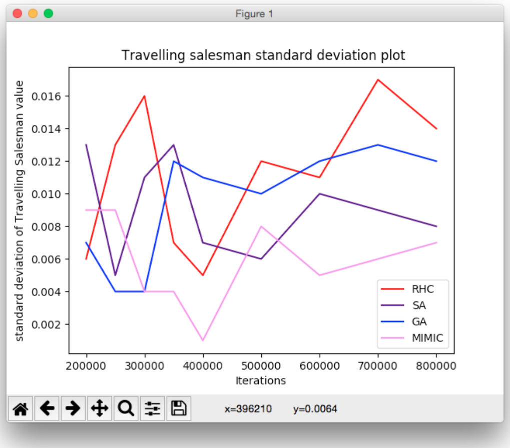
TSP is another NP hard problem.

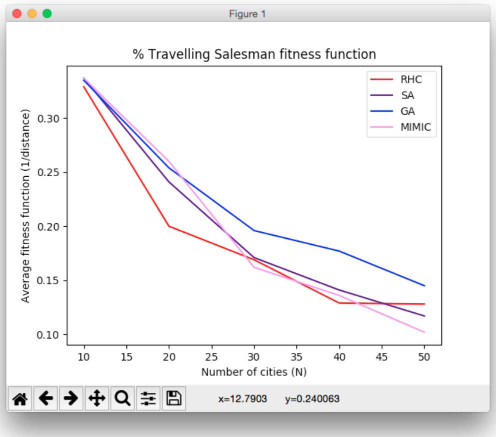
Definition: Given a list of N cities and their distance between each other, find the shortest possible path that will visit each location exactly once and ending at starting location. Since we are trying to minimize the distance, we use (1/distance) as fitness function to train on. The code is given from ABAGAIL.

Analysis: A valid solution would need to represent a route where every location is included at least once and only once. Genetic algorithm is effective due to its mutation and crossover methods. Mutation method is capable of shuffling the route. Crossover method should able to produce a valid route. In the crossover method, select a subset from the first parent and add that to offspring.









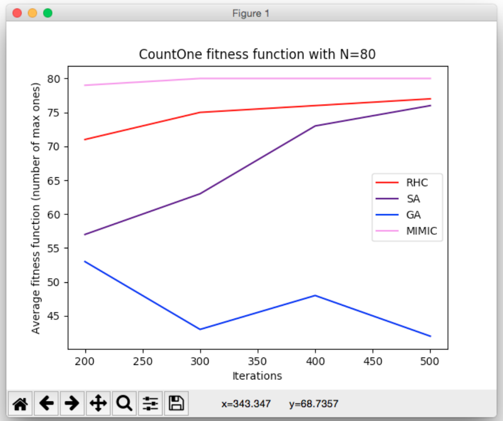
Using this domain knowledge in GA, TSP is the example where GA performs very well. From the diagram it is evident that in very few iterations, GA close to an optimum value of 1/distance. At around 550000 iterations, GA has found optimized solution and from standard deviation curve it is evident that GA has less deviation after 550000 iterations. Other Algorithms are still struggling for optimization. MIMIC didn't do well, as there is no definite distribution in the travelling salesman problem. RHS has found global optima at around 600000 and GA at around 500000 iterations, else settled for local optimum solution.
As expected MIMIC has taken much more time to converge due to its computation intensive structure.
With increasing number of cities, cost function of total travelled distance increases. This makes decreasing 1/distance, which is clear from last graph.

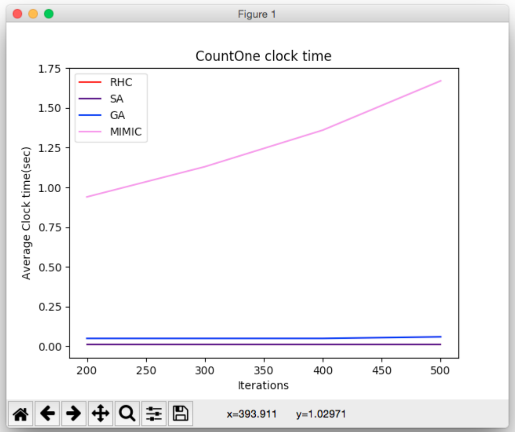
(iii) CountOnes

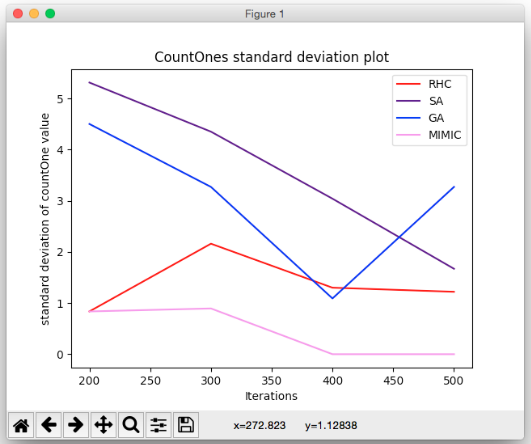
Definition: Given a bit string S of length n, we are trying to maximize the number of 1's in that bit string. Fitness function simply counts the number of 1s and returns that count. Each bit is independent of other bit and delivers no structure. We are trying to find a string x = {x1, x2, .. xn}, where x E {0, 1} which maximizes the following equation:

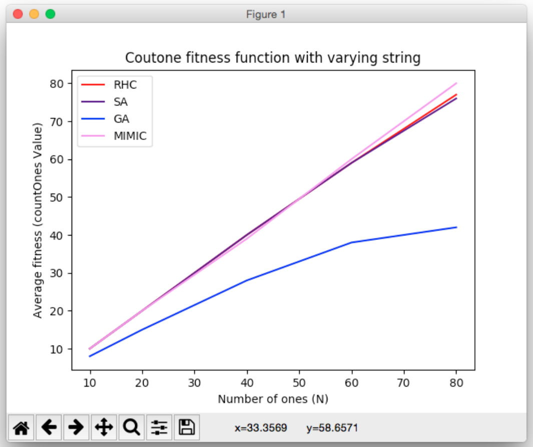
F(x) = sum(xi) (1 < i < n)

Analysis: Fitness function simply count the number of ones and return the count. This problem is interesting, because each bit is independent of the other and hence deliver no structure. We can see the diagram, RHC, GA and MIMIC performs best. Simulated annealing is not able to converge initially with low iterations due to heat parameter (100), but we see for higher iterations it start converging with high fitness value. We also see with higher iterations, standard deviation of SA start decreasing. RHC and MIMIC has low standard deviation means they are not deviating much to converge.









Mimic is computation intensive and hence takes maximum time to converge and then due to same reason with GA. GA is not suitable for all the problems, especially which are simple. Fitness value is calculated repeatedly and might be computationally expensive for some problems. Being stochastic no guarantees on the optimality or quality of solution. If not implemented correctly, GA may not converge to the optimal solution.

This technique works efficiently for single peaked objective function like cost function in linear regression. In complex problems, where we have multiple peaks and valleys, which cause them to fail as they have inherent property to stuck in local optima.

Conclusion: Random Search algorithm is best, where there is no distribution is in search space. They are simple and also takes much less time to converge.

PART-2

Implementation of Optimization Algorithm on Neural Network

Introduction: To implement the random optimization algorithm with feed forward neural networks and compare the performance with back-propagation from assignment -1.

Pima Indians Diabetes DataSet:

This Set has total Instances = 768 with 65% positive instances and 35% negative instances. From the assignment-1 we have found the best hyper parameters of Neural Network as below:

Reference: experiment Table of Datset-1 for NN,

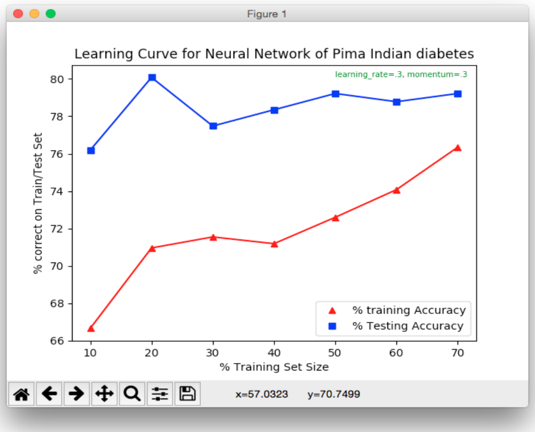
5 hidden neurons with 1 hidden layer, no of epoch = 40, learning rate = 0.3, momentum = 0.3

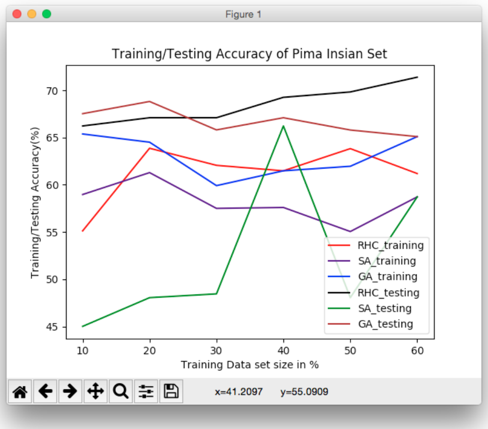
Correctly Classified Training Accuracy: 76.35%

Correctly Classified Testing Accuracy: 79.22%

Randomized Algorithmic Analysis:

Following sections evaluates Neural Network performance with varying randomization Algorithm, Training Data size and hyper-parameters of respective algorithms. Dataset is segmented into parts (60% for training and 40% for testing). ABAGAIL code is modified to work for PIMA INDIAN data set. Training, Testing accuracy and time is calculated and plot using python script with varying training data size.



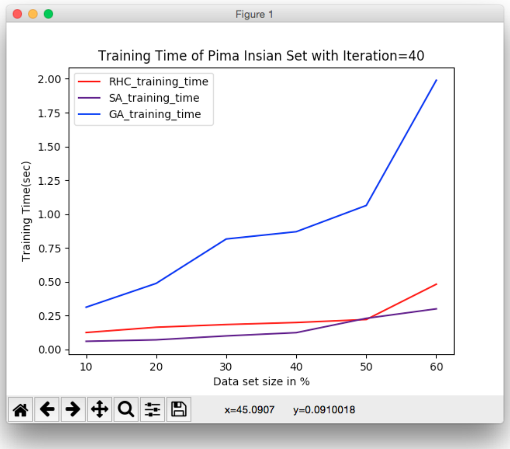


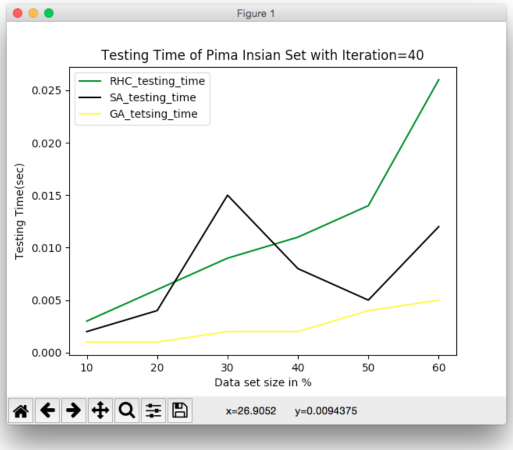
RHC Algorithm:

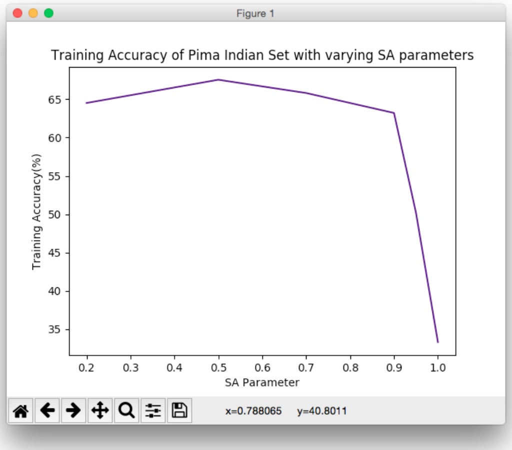
From the Training/Testing accuracy plot, it is clear that RHC algorithm worked best to find the optimized weight. The accuracy does reach it's optima with around 20% dataset and consistent afterwards. RHC also showed less standard deviation and consistently coming around 67% of training accuracy and around 70% for testing accuracy. This is comparative to back propagation's gradient descent algorithm from assignment 1. RHC employs similar principle to gradient descent which tests neighboring points to incrementally climbed towards local optima or settle a peak. Training time of RHC is also lowest.

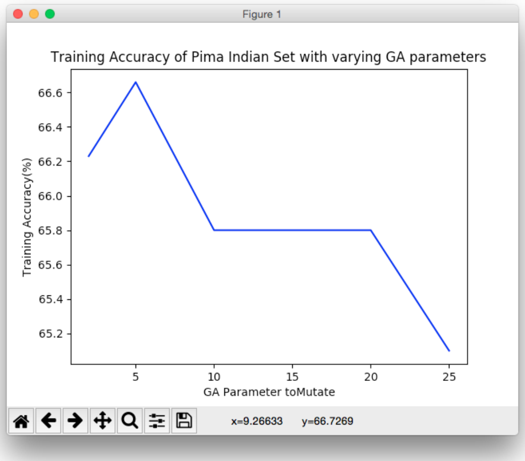
SA Algorithm:

The code is modified to find the ideal cooling rate for optimization. The accuracy varies between 45-65%. 45% lower accuracy is due to SA didn't converge. Further analysis is done to find the better hyper parameters of SA. The optimized temperature between 1E9 and 1E15 and cooling rate between 0.5 and 0.8 performs quite well. The lower temperature and high cooling rate indicates that some exploration is important at start; however high exploitation is important to select optimal neural network weights. The Graph shows very high cooling rate suddenly dropped the accuracy rate. Very high exploitation force SA to settle down to local optima.









GA Algorithm:

GA accuracy range is between 67% – 70% accuracy little lower than RHC. The further analysis is done for finding the optimized hyper parameters. In genetic algorithms, mutation provides exploration while cross over leads the population to converge on good solutions (exploitation). Beyond a light understanding of how mate and mutation affect the exploration and exploitation, GA is known to be black box for finding optimal solutions. From the curve, it is evident that toMutate = 5-10 is better performed for initial population 200.

Conclusion:

All four algorithms were looked closely and trying to find areas where they can excel. In more complex problems (NP hard) like Knapsack and Travelling salesman, MIMIC and GA performs better. On the other hand less complex or simple problems like Countone, RHC and SA perform better. RHC employs similar principle to gradient descent which tests neighboring points to incrementally climbed towards local optima or settle a peak. It is also observed that MIMIC and GA being computationally intensive and take much longer to converge.

For the second part, for the given simple dataset RHC performed better. RHC performs better in neural networks due to continuous weight. We tried to tune the hyper parameters of algorithm to find the exact optimization. RHC showed maximum of 70% accuracy while GA and SA both improved with hyper parameters, but reached tlll 67% accuracy. Accuracy is hovering around 67-70%, which we had got from back propagation as well. From given dataset's analysis in assignmen-1, it was established that given data had noise. This is further proved by this exercise too which explain lower accuracy rate.

Optimization Problem	**Best Optimization Algorithm**
KnapSack	MIMIC, GA
Traveling Salesman	GA
CountOnes	RHC, SA
ANN	RHC

References:

https://github.com/pushkar/ABAGAIL/wiki

https://gist.github.com/mosdragon/53edf8e69fde531db69e

UDACITY Lectures