GEORGIA INSTITUTE OF TECHNOLOGY

Title: Machine Learning Homework2 – Randomized Optimization

Author(s):
Weifeng Lyu

Instructor:
Prof. Charles Isbell,
Prof. Michael Littman

TABLE OF CONTENTS

1. Introduction
1.1 Problem 2
1.2 Computer Configuration 2
2. Dataset
2.1 Season Stats 2
3. Project Architecture
3.1 Python Implementation 3
3.2 Project Demonstration and Sample Result .4
3.3 Randomized Optimization Problems. 6
4. Conclusions, Discussion and Improvement 10

1. Introduction

1.1 Problem

This project implements four randomized optimization algorithms, they are Randomized Hill Climbing, Simulated Annealing, Genetic Algorithms, and Mutual Information Maximizing Input Clustering.

(a). Randomized Hill Climbing
Randomized Hill Climbing is a mathematical optimization technique which belongs to the family of local search. The algorithm will:
1.Sample p points randomly in the neighborhood of the currently best solution.
2.Determine the best solution of the n sampled points. If it is better than the current solution, make it the new current solution and continue the search; otherwise, terminate returning the current solution.

(b). Simulated Annealing
Simulated Annealing is an effective and general form of optimization. It is useful in finding global optima in the presence of large numbers of local optima. The algorithm will:
1.Hill-climbing except instead of picking the best move, it picks a random move.
2.If the selected move improves the solution, then it is always accepted. Otherwise, the algorithm makes the move anyway with some probability less than 1.
3.The probability decreases exponentially with the "badness" of the move, which is the amount delta(E) by which the solution is worsened

(c). Genetic Algorithms
Genetic Algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. The algorithm will:
1.Starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation.
2.If parents have better fitness, their offspring will be better than parents and have a better chance at surviving.
3.This process keeps on iterating and at the end, a generation with the fittest individuals will be found.

(d). Mutual Information Maximizing Input Clustering
MIMIC, a framework in which we analyze the global structure of the optimization landscape. MIMIC begins by generating a random population of candidates chosen uniformly from the input space. From this population the median fitness is extracted and is denoted θ0. The algorithm will:
1.Update the parameters of the density estimator of pθi (x) from a sample
2.Generate more samples from the distribution pθi (x), then set θi+1 equal to the Nth percentile of the data.
3.Retain only the points less than θi+1, repeat (1)

1.2 Computer Configuration

Manufacturer: Dell, Model: Inspiron 7559 Signature Edition, Processor: Intel® Core™ i7-6700HQ CPU @ 2.60GHz, Installed Memory (RAM): 8.00GB (7.88 usable), System Type: 64-bit Operating System, x64-based processor

2. Dataset

2.1. Seasons Stats

This dataset is one of the datasets being used in Supervised Learning, which is standard long-period history small volume of data, 15000+ rows, acquired from Kaggle https://www.kaggle.com/drgilermo/nba-players-stats. The data predicts which position the player plays in basketball game based on the stats they play in historical data. However, due to the expensive running time, the dataset is randomly shuffled and only acquired 1/3 of its original dataset for this experiment. The number of total dataset rows is 5000 rows, then divided into three sets: 3000 rows for training, 1500 rows for testing, 500 for validations. In addition, I pre-process this dataset with cleaning some rows and columns with null value, and unselected those weakly-related attributes based on the principle that numeric data like ages, years, total rebounds, points, three points attempts are unselected, but percentage of three point shoot, free throw shot, average rebounds per game are selected because percentage and average data are more representative than historical numeric data. After unselecting those attributes, also cleaned the final output so that the column for that position will be 1 if the basketball athlete plays that position, if not then the column for that position will be 0.

3. Project Architecture

3.1. Python Implementation

This project implements in Pycharm IDE with Python integrated with ABAGAIL and Jython 2.7, also with other toolkits such as Numpy, Pandas and machine learning framework sci-kit learn. The experiment will be implemented in all Python environment plus Jython 2.7. Based on the libraries provided, we will implement the algorithm to train the model, all the midway data will be logged into txt file so those data are processed by Python Matplotlib to plot the figure. The workflow is designed for analyzing the accuracy, time complexity, mean squared error based on different randomized optimization methods, these methods are Randomized Hill Climbing, Simulated Annealing, Genetic Algorithms, Mutual Information Maximizing Input Clustering. For each of the algorithms, this experiment iterates different variables, except one for back propagation, the performance matrix will be logged into files and plotted for further analysis. This report demonstrates many graphs about analyzing accuracy, time performance and mean squared error in comparison of iteration-series. These four randomized optimization algorithms have many parameters to tune so that the best optimization result can be observed in reasonable amount of time. To obtain the views but maintain the readability of the report, the implementation chooses the following input parameters:
INPUT_LAYER = 24 (There are 24 input attributes for training, testing and validating)
HIDDEN_LAYER1, HIDDEN_LAYER2, HIDDEN_LAYER3 = 10 or 38
OUTPUT_LAYER = 5 (There are 24 output positions for verifications)
TRAINING_ITERATIONS = 100001
TRIALS = 11
ERROR_SAMPLE_INTERVAL = 100
HALT_COUNT_MAX = 5000
HALT_COUNT_THRESHOLD = .000001
The graphs will follow the same conventions to generate plots, parameters as x index, then the second parameters as legend, y axis such as accuracy, time to provide a good overview of randomized optimizations performance.

(a). Parameter Tuning
We implement neural network Back Propagation algorithm in the basic for contrasting, and there is no parameter tuning implemented for Randomized Hill Climbing. For Simulated Annealing, we choose the cooling rate between 0.15 to 0.95 in 0.2 steps for iteration. For Genetic Algorithm, we choose mate for 20 or 10, mutate for 20 and 10, using cross product to iterate and compute the graphs. The parameter tuning is objective to determine how these parameters impact for four algorithms and how they should select considering time and accuracy.

(b). Accuracy Analysis
We use the function, errorOnDataSet(network, ints, measure) to compute the accuracy for all algorithms. To improve the data authority, we execute the experiments for 11 trials to compute the average of accuracy. The final accuracy is calculated by the average of 11 trials. In the case that some optimal are acquired in fewer iterations, we fix this issue by padding the optimal accuracy from the last index in which the experiment has to the longest iterations in all trials. Therefore, all the trials have the same iterative length with only trivial impacts, we compute numpy.mean for those data and obtain the average of accuracy. The accuracy score will be evaluated how well this algorithm performs.

(c). Time Complexity Analysis
We use time counter to execute the calculation of time. We track the time at each end of step iterations so the total time to reach the optima is obtained. Same idea as Accuracy analysis, we pad the optimal elapsed time from the last index in which the experiment has to the longest iterations in all trials, then compute numpy.mean for obtaining the average of time consumed. The time it takes for each iteration will be evaluated how fast the algorithm performs.

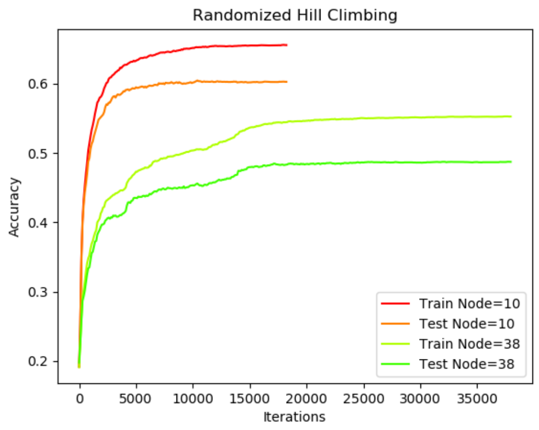
(d). Optimization Analysis

We will find three optimization problems to analysis, which are Flip-Flop, Continuous Peak, Traveling Salesman. Same idea as above, we will reiterate the experiments for 5 times for each optimal reach, then take the average of the optima data to improve the data authority.

3.2. Project Demonstration and Sample Result

(a). Randomized Hill Climbing

Randomized Hill Climbing is easy to apply, does not need many resources, usually fast.



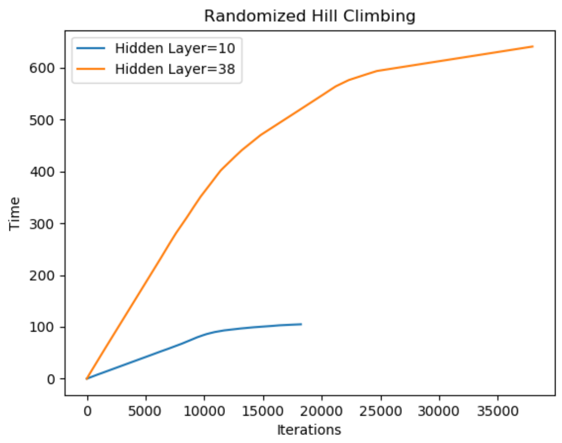
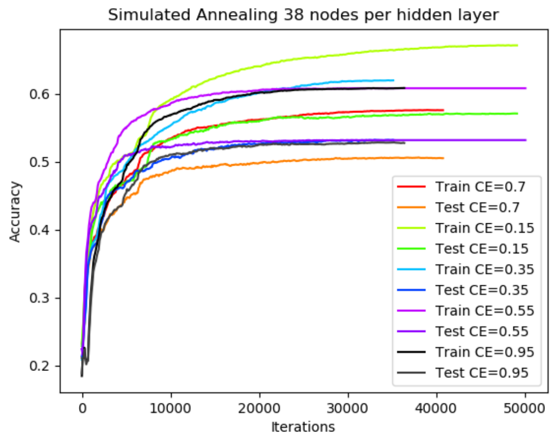


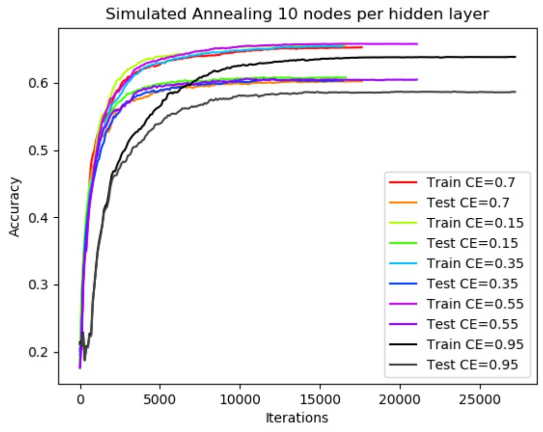
Figure1. Randomized Hill Climbing Accuracy and Time

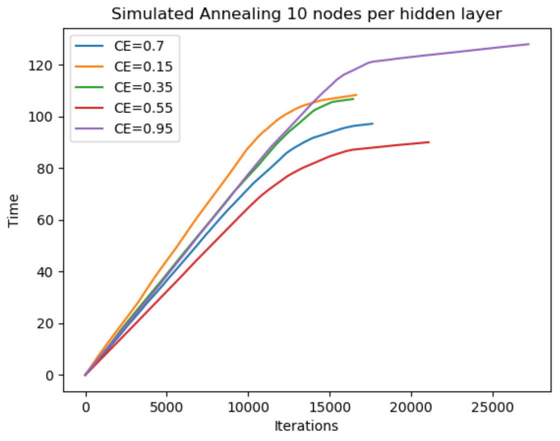
Randomized Hill Climbing at lower hidden layers perform better than greater hidden layers at both time and accuracy performance in this problem. Since too many weighted layers could involve overfitting issue in this problem, it is recommended to implement Randomized Hill Climbing with smaller hidden layers. In addition, the size of dataset will affect the accuracy, the size of training set is twice of test set, we can observe there is 5% difference between their accuracies. Global optima will take longer time to achieve in this scenario since Randomize Hill Climbing is easily satisfied with local optima, which will cause too many iterations to restart the process.

(b). Simulated Annealing

Simulated Annealing heuristic considers some neighboring state s* of the current state s, and probabilistically decides between moving the system to state s* or staying in-state s. These probabilities ultimately lead the system to move to states of lower energy. This step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted. Even though the simulation reaches the local optima, the probability of going downward is $p(x, xt, T) = 1\ if\ f(xt) \geq f(x), or\ e^{\frac{f(xt)-f(x)}{T}}\ otherwise$. Therefore, Simulated Annealing will not get stuck at local optima and will continue to look for global optima with saving the total iterations.







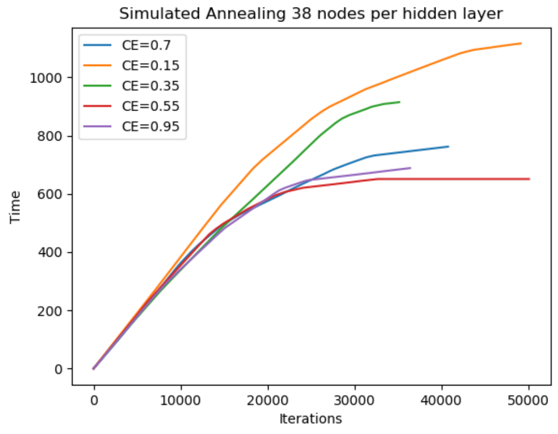
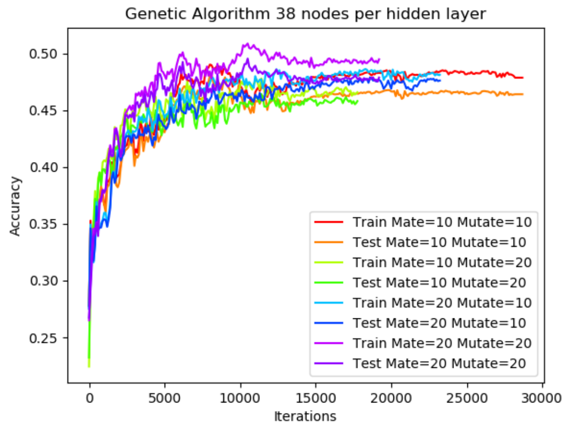


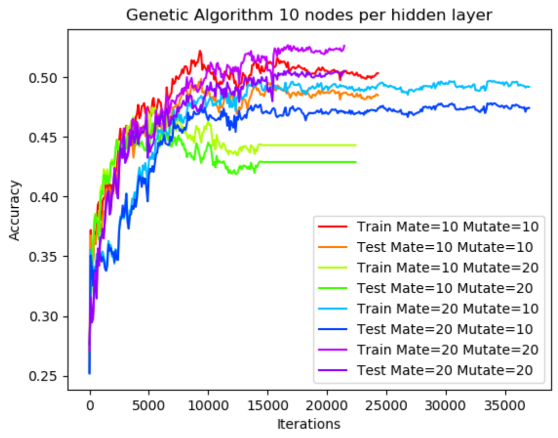
Figure2. Simulated Annealing Accuracy and Time

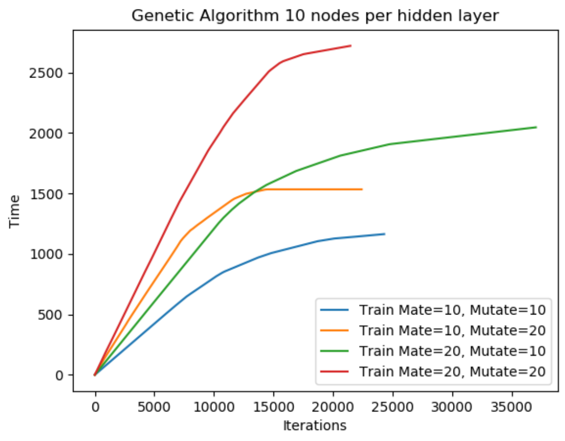
Simulated algorithm is based on exponential and geometric cooling schemes depending on their relative cooling rates. From the above figures, the cooling rate should be low enough for the probability distribution of the current state to be near thermodynamic equilibrium at all time. The relaxation time is another factor could be affected, which wait for the equilibrium to be restored after a change in temperature also depends on the candidate generator, in a very complicated way. Therefore, the ideal cooling rate cannot be determined beforehand, and should be empirically adjusted for each problem. From above figures, Simulated Annealing with lower cooling rate tends to perform well with lower iterations in this dataset at both time and accuracy performance. The increasing of nodes per hidden layer does not vary a lot at accuracy level but require expensive computational time. The accuracy tends to converge at around 15000 iterations, with higher cooling rate, it tends to converge faster.

(c). Genetic Algorithms

The algorithm creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation). Genetic Algorithm quickly finds higher fitness values in lower iterations but oscillates in higher iterations. This may be due to the large problem space where finding a small population that allows us to improve fitness consistently is unlikely. Also, despite the low iterations to convergence for Genetic Algorithm, there is more work per iteration in comparison to the hill climbing algorithms. Every iteration has the Genetic Algorithm evaluate and rank the fitness of its population and perform crossover and mutation. Crossover causes a more controlled and justified move in the location of the generated solutions. Mutation is a random explore, and crossover is more like exploiting. Each time a gene is considered, the mutation probability is used to determine whether to mutate. small values ensure that not too many mutations are considered at once, it will depend on the number of genes in each member of the population. Therefore, greater mates, higher mutation probability could be positive at increasing accuracy but will also require more time.







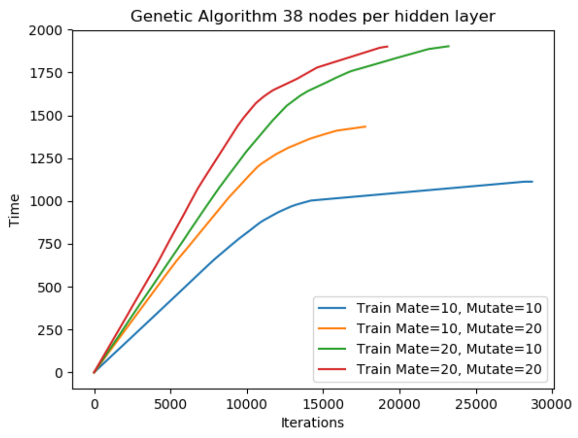
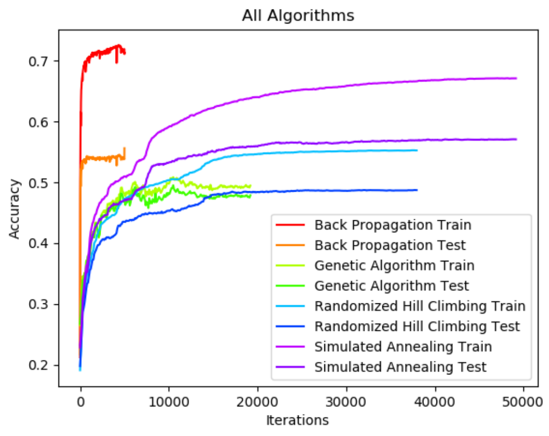


Figure3. Genetic Algorithm Accuracy and Time

From the above figures, it illustrates Genetic Algorithm with greater mate and mutate probability will perform better accuracy but require more computational time. If mate is greater than mutate probability, it will lead to the fact that the algorithm takes many iterations to converge but the time for converging is equally fast with same number of mates. In this experiment, it is recommended to balance its time complexity and accuracy regarding the parameter choice.

(d). Algorithm Comparison

In this experiment, we will discuss the comparison among Back Propagation (what we used in Assignment 1), Genetic Algorithm, Randomized Hill Climbing, Simulated Annealing to compare how they perform regarding time complexity and accuracy using train set (more data) and test set (less data = half of train set). All the algorithms are selected with the best accuracy that it can achieve for each parameter settings.



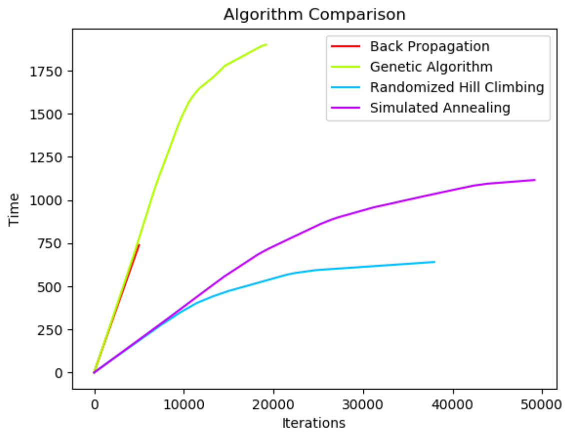


Figure4. All Algorithm Accuracy and Time Comparison

From what we can see in the figure, we can conclude the following facts:

1. All randomized optimization algorithms perform worse than neural network backpropagation respective to accuracy, and also takes greater amount of time to converge, especially Simulated Annealing. Since the network weights are continuous values rather than discrete, this makes the problem space infinity and narrow the basins attraction that randomized optimization algorithms is likely to find. Back Propagation also adjusts the weights in the right direction based on that, where randomized optimization tries completely random changes and ignores that information.
2. The test set has great margin of accuracy with train set under Back Propagation Network. All optimization algorithms seem to overfit easily in low iteration compared to backpropagation. From here we can observe that randomized optimization does not greatly depend on the size of dataset. One of the explanations could be that the margins between the training and test accuracy for each algorithm in the proceeding plots tells that Back Propagation is gradient based and seeks iterative improvements based on previous weights. These weights are based

on training data, so Back Propagation adheres closely to those training data then resulting in overfitting. By contrast, Randomized Optimization provides stochastic nature that they do not retain the data on training data per iteration and thus able to avoid overfitting with training set.

3.3. Randomized Optimization Problems

(a). Flip-Flop

Flip-Flop is simple structure algorithm that has low gained in the intensive computations of probability distributed generation, with increasing size, the processes will take to find the highest fitness regardless of time and iterations. Flip-Flop Optimization assigns a higher value with bit strings that alternate between 0 and 1. The greatest uninterrupted sequence of 0s and 1s are set to fitness value, the global optima after iterations is N-1. Here we iterate N from 20 to 200, taking the average after 5 trials.

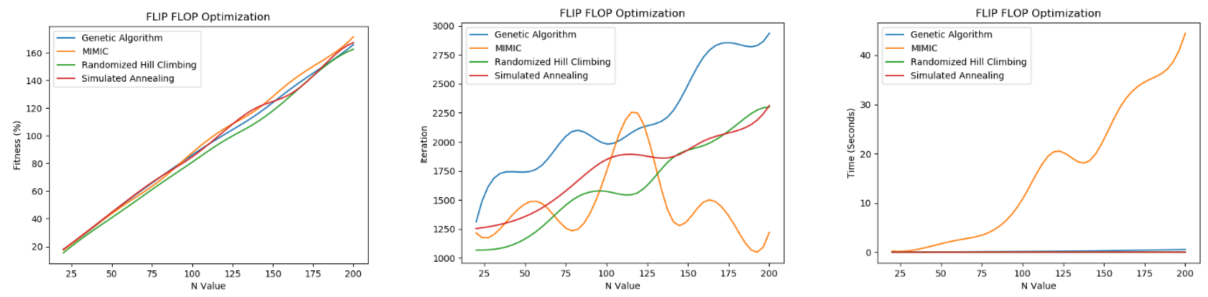


Figure5. Flip Flop Optimization

We set four algorithm parameters to (mate=0.2, mutate=0.2 in Genetic Algorithm), (CE=0.1 in Simulated Annealing), (M=0.1 in MIMIC). From the figure we can see the facts:

1. MIMIC Algorithm costs more time to converge than other algorithms in this problem with highest fitness achieved, it reflects that MIMIC takes more time to construct a dependency tree for every iteration.
2. MIMIC tends to explore the problem space earlier with fewer iteration, allow to avoid local minima more effectively than other algorithms. Other algorithms will keep searching for the optima for more iterations.
3. All algorithms can achieve same levels of fitness after reaching the optima, which is linearly subject to N value.

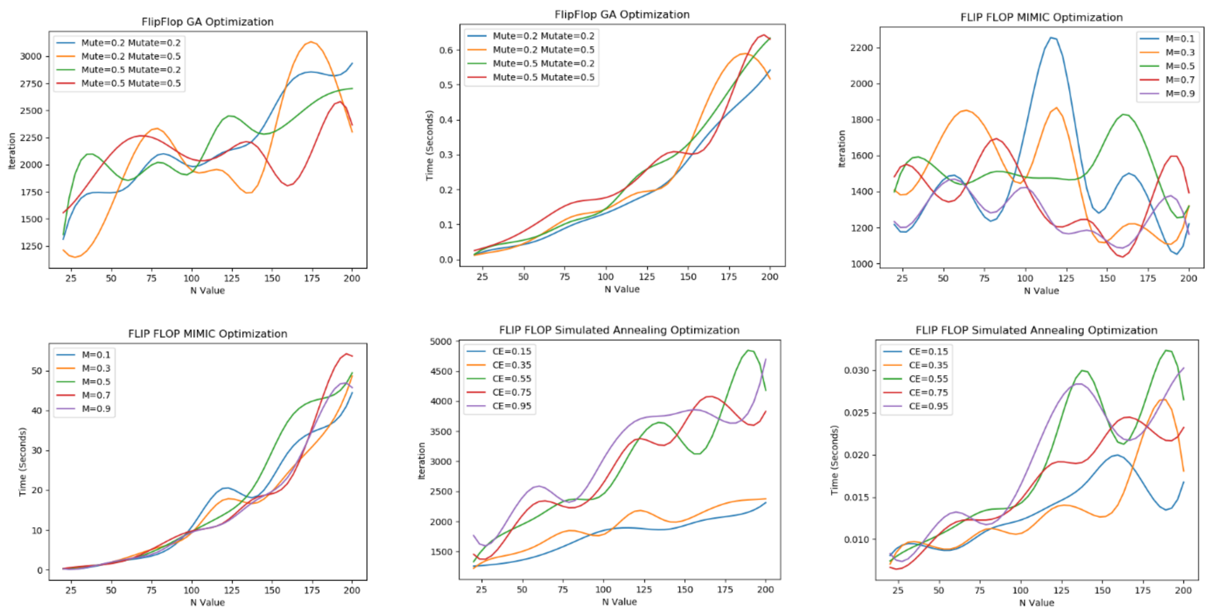


Figure6. Flip Flop Optimization

From above diagrams considering flip-flop problem is detailly implemented in four randomized optimizations, we can see not all the algorithm's parameter has significant impact on its performance so there are some graphs not posted here but in the project directory folders. From Flip-flop problem with lower cooling rate adjusted, simulated annealing tends to limit its iteration to find the optima and achieve the same fitness. As our neural network analysis illustrated, Simulated Annealing is given a chance to explore the data because of its temperature component. It gets stuck in local minima like RHC but is able to escape. Simulated Annealing shines in this type of problem with its tendency to explore the problem space in early iterations, has the best tradeoff between time cost and achieving high fitness value in this flip-flop problem. Although MIMIC and Simulated Annealing achieves same level of fitness in this selected problem, it is best to use Simulated Annealing because less computation time is cost.

(b). Continuous Peaks
Continuous Peaks Problem assigns higher fitness values with sequences of unbroken 0s and 1s in the bit string, then assigns a bonus once the max sequence of both 0s and 1s are greater than a preassigned value of T. The global optimum is 2N-T-1, when higher values of T is a difficult optimum to achieve, the algorithms will tend to stop with acquiring the local optima. The value of T for these experiments was set to 2N/5, which represents a narrow basin of attraction for finding the global optima.

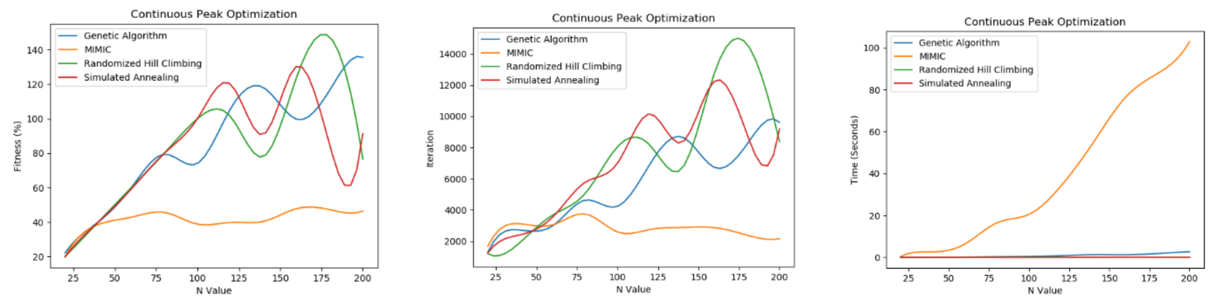


Figure7. Continuous Peak Optimization

Here we iterate N value from 25 to 200 with 25 increment, from the above figures we can see the following facts:
1. MIMIC achieves worst fitness in this problem with greatest time consumed with smallest iterations. As N increases, MIMIC is outperformed by other algorithms. At this point the problem size may be too large for MIMIC to effectively determine the feature dependencies that yield the best estimated probability distribution.
2. Surprisingly, randomized optimization outperforms other algorithms around N=175 but starts dropping after it, this might lead to the fact that in the continuous peak, Randomized Hill Climbing does not get stuck in the local optima.
3. Genetic Algorithm and Simulated Annealing are best at high N value. Genetic Algorithm is benefited from crossover to allow better fitness values when it’s searching for the peak. Simulated Annealing has certain probability of avoiding stuck at local optima.
In contrast of last problem of Flip-flop, here we can find not all the algorithms can obtain high fitness. If we look into time comparison, Randomized Hill Climbing and Simulated Annealing takes the most iterations with least time cost as expected, and iterations also drop around 160-170 N value. They are computationally lighter compared to Genetic Algorithm and MIMIC, only evaluating one neighbor per iteration, then updating the current hypothesis if the fitness function evaluation at the neighbor is greater in specific functions. Now we can dive into more detailed analysis for different algorithmic parameters in continuous peak problem.

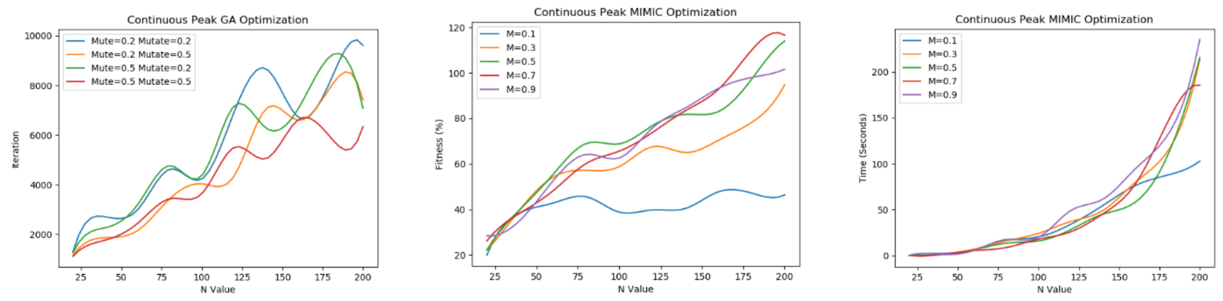


Figure8. Continuous Peak Optimization
From the experiments implemented, not all algorithmic parameters play significant impacts on performance, so we only select 3 figures to analyze how they affect in continuous peak problem, other figures are in project directories.
1. When mute and mutate are high in Genetic Algorithm, it tends to take less iteration to reach the optima.

2. When M in MIMIC algorithm is low, the fitness it can achieve is very bad, also with less time taken. Therefore, to look for the best optima, M value is an important factor to track to guarantee the fitness performance. As our neural network analysis illustrated, the best algorithm is debatable, it can be either Randomized Hill Climbing, Simulated Annealing or Genetic Algorithm. From Time analysis perspective, Randomized Hill Climbing and Simulated Anneal perform equally well, but Genetic Algorithm performs best with larger values of N and given more time for tweaking of hyperparameters. Randomized Hill Climbing appears to be the most well-rounded algorithm for this problem with the simplest, fastest, highest fitness, just need to track the N value in case of it starts dropping drastically.

(c). Traveling Salesman

Traveling Salesman Problem aims to connect a set points, or ‘cities’, with the shortest path possible and return to the origin city without visiting any other point more than once. For any set of points N there are N! possible routes, making this a problem domain that quickly becomes very large with increasing N. It is most easily expressed as a graph describing the locations of a set of nodes.

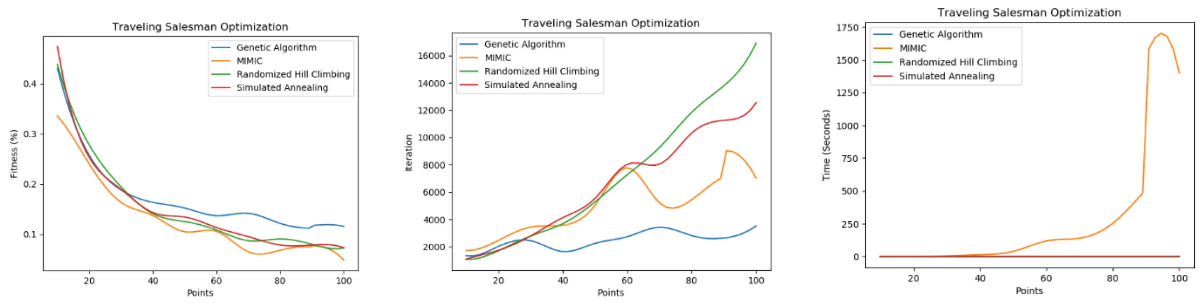


Figure9. Traveling Salesman Optimization

For this project, the number of points is between 10 and 100 with 10 increment, and the (x, y) location of these points are randomly generated after which all algorithms are run on the same set of points. From plots generated, we can draw the following facts:

1. Fitness is decreasing respective to the number of increasing points for all four algorithms in this problem. Therefore, the problem should be small enough for the algorithms like SA and RHC to stumble upon high fitness values easily.

2. Genetic Algorithm takes least iteration and time, while other algorithms increase its iterations as value of points increases. It seems to be the best algorithm to apply because crossover plays a significant role in the success of this algorithm. In addition, the ABAGAIL library includes a genetic algorithm crossover function specifically designed for the Traveling Salesman Problem. Therefore, the computation is much simpler and faster. A child path is created by selecting any starting point and the next point in the path is selected by seeing how the parents are connected and choosing the shortest distant one. Furthermore, the mutation function ensures that other paths are explored that might not be present in the current population. For these reasons, Genetic Algorithm iteratively find better fitness compared to other algorithms.

3. MIMIC Algorithm costs more time to converge than other algorithms, also generate lowest fitness. It easily gets stuck at local optima. It seems to be the worst algorithm in this problem.

4. Simulated Annealing and Random Hill Climbing perform equally well but worse than Genetic Algorithm, but when the number of points increase, Genetic Algorithm wins them.

All algorithms except Genetic Algorithm takes increasing iterations to converge based on the number of points in the problem domain. Genetic Algorithm takes the lowest number of iterations to converge among almost all problem sizes. This suggests that Genetic Algorithm is best, effective enough, through crossover and mutation. Regarding to Simulated Annealing and Randomized Hill Climbing could be evaluated in cases whether time is important. MIMIC performed worst so it is not suitable to apply in this Traveling Salesman Scenario.

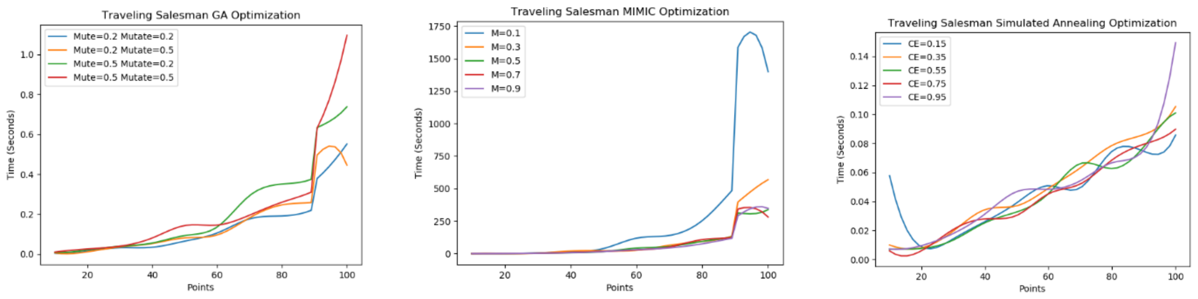


Figure10. Traveling Salesman Optimization

Traveling Salesman problem is the most time-consuming computation respective to other algorithms running in this project, so time is the most considerable factor to evaluate. There are some other graphs related to iteration and fitness, but those algorithmic parameters do not weight too much impact on this problem, so we will not discuss here, the plots are under project directory for reference. From above figures, we can find that when points are increased to specific size, the computation grows similar-exponentially. From the left figure, it is recommended to reduce the mute and mutate for less computational time. For the middle figure regarding MIMIC algorithm, when M is small, it takes thousands of seconds to complete one optimal, which is unacceptable. We also notice that in low values of cooling rate for Simulated Annealing, it requires greater time to find the optimal, which illustrates the fact that it is better to adjust the cooling rate for graphs with fewer node points if Simulated Annealing is in used. Parameter choice for randomized optimization algorithms is quite important for Traveling Salesman problem because it varies a lot respected to time performance.

5. Conclusions, Discussion and Improvement

In this assignment, there is some data-cleaning and preprocessing required to be done before putting those data into algorithms. The output values are normalized to be 1 at the last few columns.

Randomized Hill Climbing, Simulated Annealing, and Genetic Algorithm are all interesting randomized optimization algorithms. They can be utilized for very complex and structured functions, as well as be utilized for their efficiency in optimization problems. We also see that optimization algorithms do not perform well in large problem domains with continuous values and back propagation demonstrates a more efficient algorithm to determine the network weight. If the problem domain size by using fewer network node is reduced, we will achieve higher accuracy with less overfitting. Implementing best-fit randomized optimization algorithms into different datasets is a machine learning technique requiring lots of skills, also as known, associated with many crossed study fields.

In my opinion it is likely the best to utilize these types of algorithms where there is no concrete solution to functions and randomization would be an advantage, and in combination with other learning techniques. Back Propagation has just proved to be the most efficient for more complex tasks in general; there are circumstances where other local searches are better. Randomized Hill Climbing on a neural network to find a fair solution quickly, but it wouldn't be feasible to find a near optimal solution. Genetic Algorithm may help you to find an optimal Neural Network design but normally they have so many drawbacks such as algorithm parameters' tuning, computation complexity, which is not that feasible for real-world applications. Simulated Annealing is a randomized approach to find the global solution, while Back Propagation is a gradient-based search and thus has an inclination to be trapped in the local minima.

In conclusion, this assignment gives me a deep understanding of how Randomized Optimization works and an idea of how it is applied to problems such as Flip-flop, Continuous Peak and Traveling Salesman Problem, also improve my skills of manipulating large set of data in Python.