CS 7641 Machine Learning Assignment 2

Siyuan Li
sli761@gatech.edu

***Abstract*—In this report, I focused on 2 problem sets. The first one is that I utilized 3 methods of randomized optimization to find the best weights for the neural network on Italian Mortgage Loan Default problem. The second one is that I applied 4 randomized optimization method to 3 discrete optimization problems and compared the performance of different methods, regarding training time, fitness score, etc.**

0 OPTIMIZATION METHOD OVERVIEW

In this section, I will first illustrate the idea of each of 4 random optimization algorithms, so that we can have a better idea what we are dealing with.

1) Random Hill Climbing

Random Hill Climbing is a local search algorithm used to find the best solution for a given problem by starting from a random solution and iteratively improving it by exploring the neighboring solutions. The algorithm evaluates the objective function for each candidate solution in the neighborhood and selects the one that maximizes the objective function until a stopping criterion is met. Although it may get stuck in local optima, Random Hill Climbing is often used as a benchmark algorithm and can provide insight into the problem structure.

2) Simulated Annealing

Random simulated annealing is an optimization algorithm that generates new solutions by random perturbation of the current solution. The new solution is evaluated, and if it is better than the current one, it is accepted. If it is worse, it may still be accepted with a probability that depends on a temperature parameter. The algorithm explores the solution space more widely at higher temperatures and converges towards the global optimum as the temperature decreases.

3) Genetic Algorithm

The Random Genetic Algorithm is an optimization method inspired by natural selection. It generates a group of potential solutions, evaluates their

fitness, and evolves them over time through selection, crossover, and mutation. The GA is effective in exploring a large search space, but its performance depends on various factors that need to be optimized.

4) MIMIC

MIMIC stands for Mutual Information Maximizing Input Clustering, and it is a probabilistic optimization algorithm used to solve combinatorial optimization problems. MIMIC generates candidate solutions and estimates their probability of being optimal. It builds a probability distribution model of the problem space and uses mutual information to determine which variables should be included. MIMIC is best suited for complex problems with dependencies between variables.

1 NEURAL NETWORK WITH RANDOM OPTIMIZATION

In this paper, I will use Italian Mortgage Loan Default data prior year 2004 dataset.

1) Dataset description

The dataset itself is obtained from my real work, which is to classify whether a loan applicant will default in the future. The whole dataset contains over 300,000 records from year 1990 to year 2020 of Italian mortgage loans. In this project, for simplicity, I only truncated the data before 2004, which contains 21,235 records. The dataset contains the following features:

- Explanatory Variables: OLTV, DTI, Employment Status, Loan Purpose, Payment Frequency, Interest Rate Type, the property's occupancy status, the loan's originator channel, payment type and whether the borrower is foreigner.
- Response Variable: Default (0: not default, 1: default)

Default	*Number of observations*
0	18518
1	2452

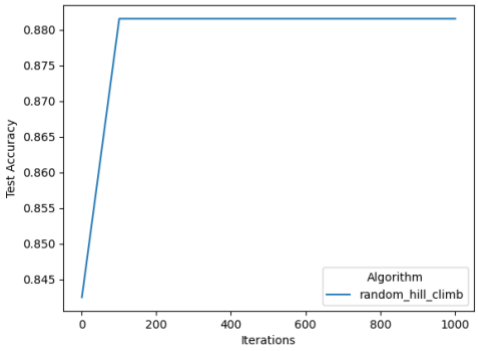
2) Methodology

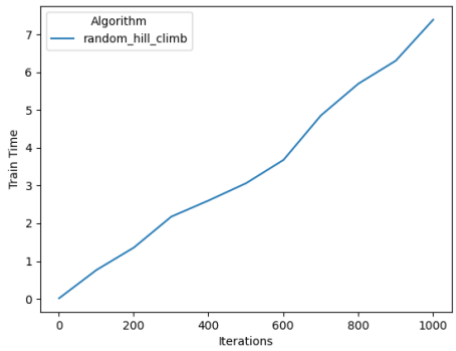
The dataset was split in to 60% as training set and 40% as test set. All the random optimization algorithms are run with 2 hidden layers, with each layer have 5 perceptron units. The reason I set the parameter like this is because I want to compare the performance with the one I tested in the first assignment.

To tune the parameter, the max iteration for each algorithm is always set to [1, 101, 201, 301, 401, 501, 601, 701, 801, 901]

3) Randomized Hill Climbing

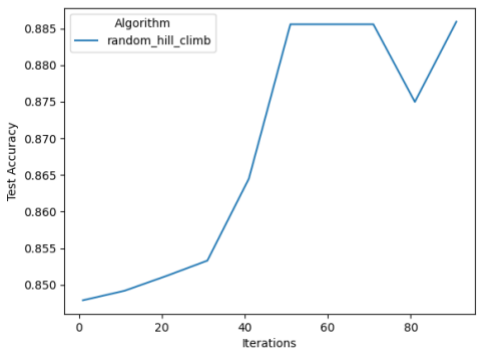
The accuracy vs iteration and training time charts are shown as below:

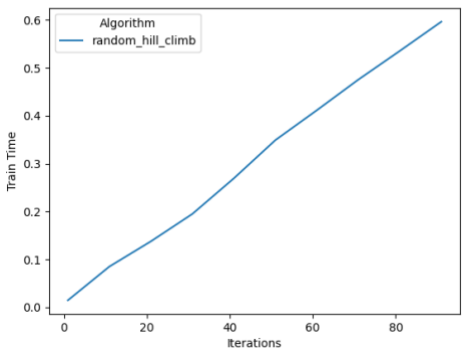




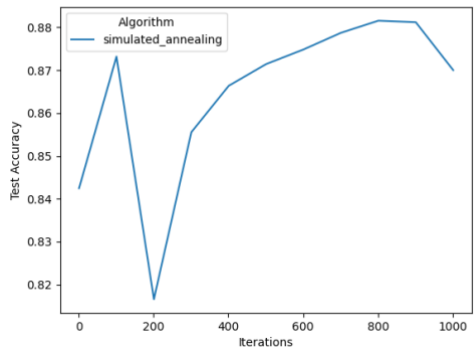
We can tell from the chart that the RHC converges with the increase of iteration. Also, for the training time, it increases in a linear-wise pace with iteration times.

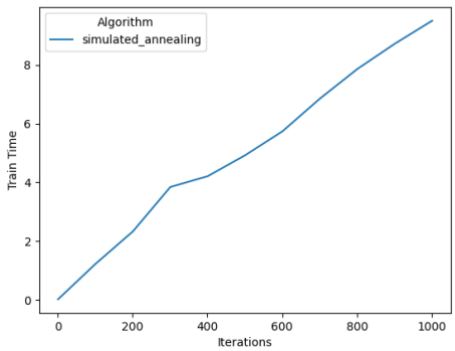
From the charts above, we cannot clearly observe the pattern between iteration and test accuracy as it has already converged to 89% at 101 iterations. To compensate for it, I re-ran a zoom-in chart:



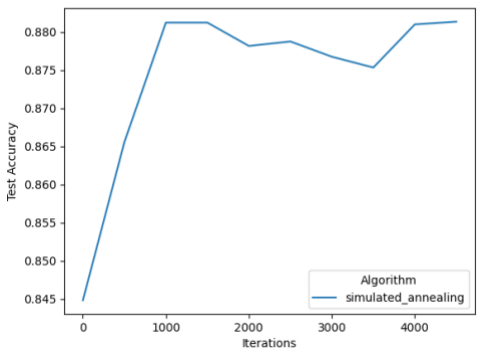


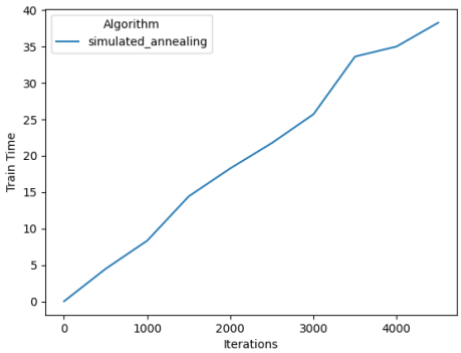
4) Simulated Annealing





For the simulated annealing algorithm, the results do not converge in the 1000 iterations. At 1000 iteration level, the accuracy is lower than RHC (87% vs 89%). To have a full view, I expand the iteration times.

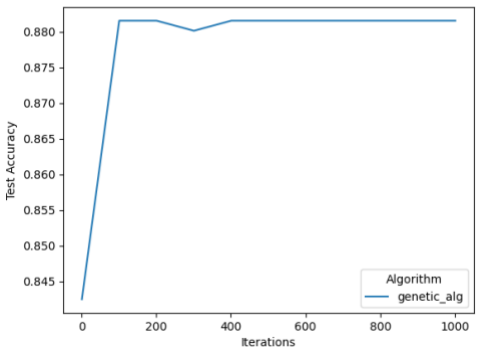


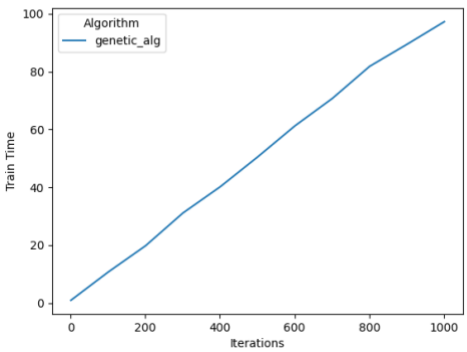


We can see after the iteration goes to 5,000, the result converges to 88.5%, better than lower iterations but still not as good as RHC.

I was trying to figure out how we can try different cooling exponent in mlrose package but failed to find the corresponding parameter. It should be very interesting to see how the performance changed with respect to cooling exponent change.

5) Genetic Algorithm

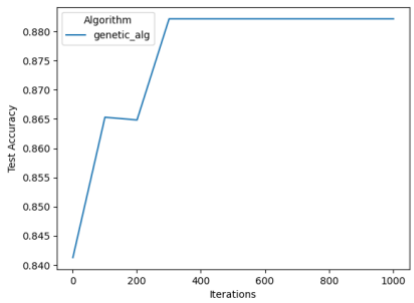


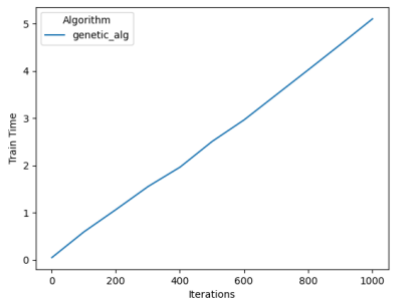


For genetic algorithms, we can see the result converges quickly when iteration increased to 100. Please note, the population setting for above charts is 200 with mutation probability as 0.1. We can also note that although the predict accuracy is similar to previous 2 method, the training time for GA is significantly longer (100s vs 8s when iteration time is 1,000).

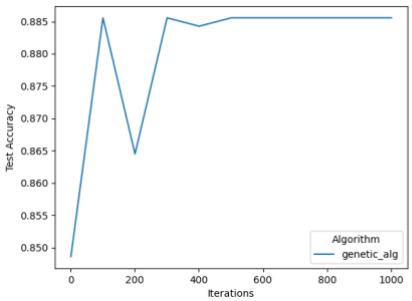
To get a better understanding regarding how population size will impact predict accuracy, I did the below experiments.

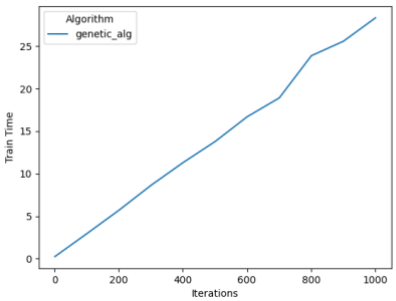
Population = 10:



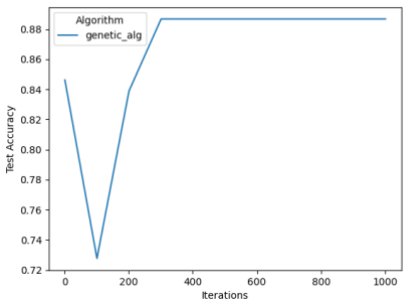


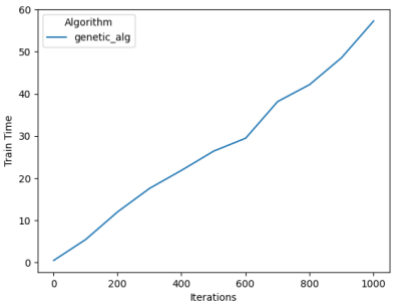
Population = 50:





Population = 100:





We can see from above that although the test dataset accuracy is always similar, the training time increased significantly with the increasing in population size: 5s for population size of 10, 25s for 50, 50s for 100 and 100s for 200.

6) Algorithm comparison

Algorithm name	Accuracy	Training time
RHC	0.886037202731339	4.257929086685185
SA	0.88007220783298	6.503951311111145
GA	0.885409308531512	379.3393681049345
Benchmark	89.97%.	390s

It's interesting to know that with randomized optimization on the weights, we cannot actually beat the benchmark, which is set as learning rate as 0.01 and Alpha as 0.0001

2 OPTIMIZATION PROBLEMS

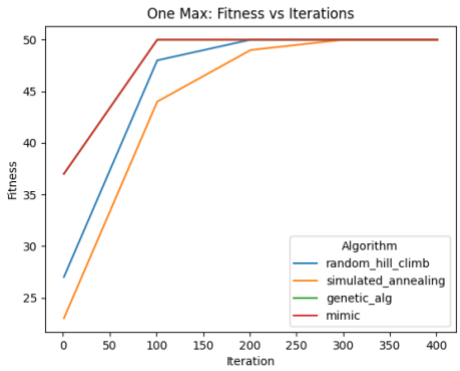
In this section, I will discuss the application and performance of the four random optimization method on the 3 problem sets, which are One Max, Flip Flop and Four Peaks

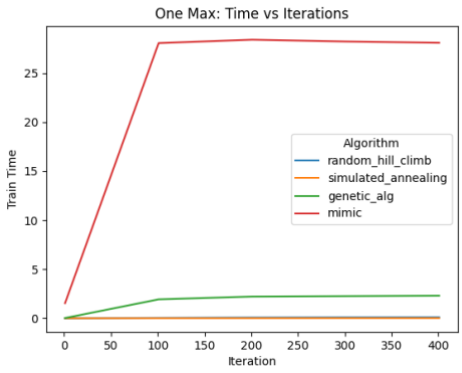
For each problem, the algorithm was repeatedly applied with [1, 101, 201, 301, 401, 501] as their max iterations. To keep the result comparable, we always fix the random state to 10. For MIMIC and GA, I fixed population size to 200 as this is what we see best from NN example.

1) One Max

The One Max problem is a well-known optimization task. In this task, we need to find a binary string of fixed length that contains the highest possible number of ones. For instance, if the length of the binary string is 10, the ideal solution would be a string consisting of ten ones, represented as "1111111111".

In this report, I set the length of the question to 50. The performance of each algorithm is as below:





From the charts above, we can see that when iteration goes to large enough, all the algorithms have a very similar performance regarding the fitness score (all around 50). However, when the iteration small, MIMIC has a significant higher fitness score. Also, the level of fitness score of simulated annealing increased most rapidly with the increase in iteration.

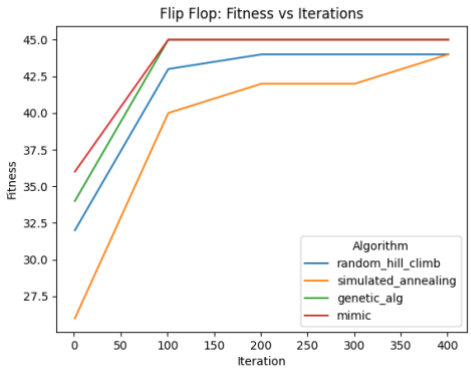
As for the training time, MIMIC's training time is several times higher than other algorithms. Given the similar performance, we can refuse MIMIC to be the best algorithm in this problem. In the meantime, the training time for simulated annealing and random hill climbing is almost constant, regardless the increase in iteration.

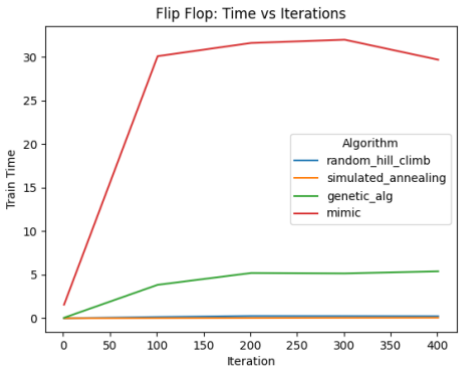
In this way, I would say random hill climbing is the best algorithm in the One Max problem.

2) Flip Flop

The Flip Flop problem is another classic optimization problem. In this problem, the objective is to find the binary string of a fixed length that maximizes the number of alternating ones and zeros in the string. For example, if the length of the binary string is 10, then the optimal solution would be a string such as "0101010101" or "1010101010", where the ones and zeros alternate. We can consider the Flip Flop problem as a variation of the One Max problem that adds an additional level of complexity to the optimization task.

Again, I set the length of the problem to 50. The performance of each algorithm is as below:



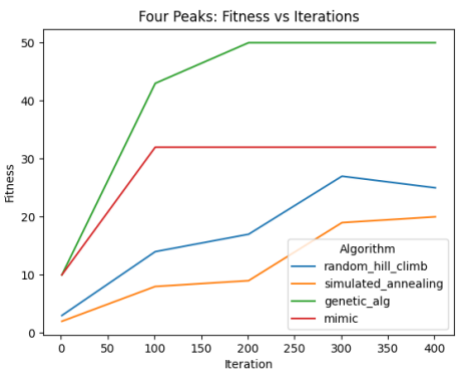


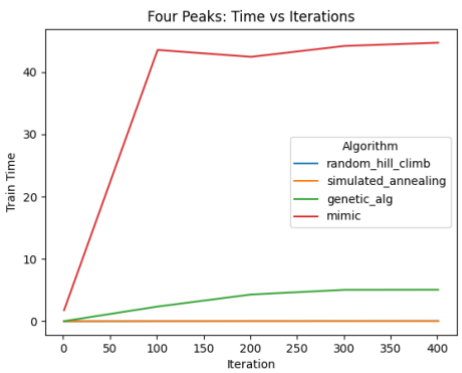
As Flip Flop problem is a variation of One Max, I was expecting that these 2 problems will have similar performance comparison matrix. However, it seems that for the flip flop problem, all the algorithms cannot get to the same level of fitness score as in the One Max, which makes sense as the complexity of Flip Flop is much higher than the One Max. Correspondingly, the training time for the same algorithm increased by a bit compared to One Max.

In the problem set, genetic annealing has the same fitness score with MIMIC, but with much lower running time. So, I would say SA is the best algorithm in this problem set.

3) Four Peaks

The Four Peaks problem is another well-known optimization problem. The target is to find the optimal solution for a binary string of a fixed length that has four peaks, i.e., four contiguous blocks of ones. The objective is to maximize the number of ones in the string while also ensuring that the string has at least one peak. The Four Peaks problem is more complex than the One





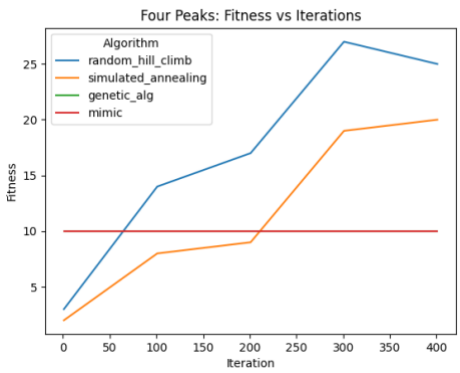
Max problem and the Flip Flop problem, so I would expect a lower fitness score and much longer training time. Still, the problem length is set to 200.

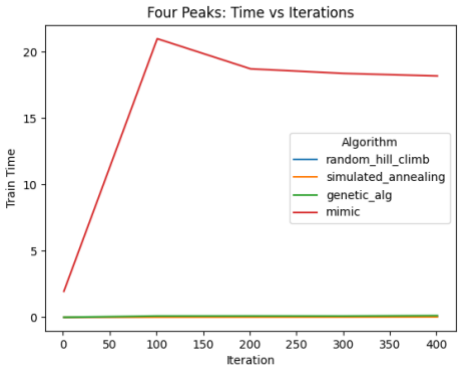
Similar to previous problems, MIMIC is always the algorithm that takes longest time. However, in this problem set, it is no longer with the highest fitness score. Genetic programming, although used much short training time, becomes the algorithm with best performance. It is also very interesting to learn that in this problem set, RHC and SA's performance are much worse than GA and MIMIC. We can infer from this that RHC and SA are more suitable for simple optimization problems, given their faster running time. However, GA and MIMIC are more suitable to solve complex problems and can always gave us a very good fitness score.

Learning from the example of neural network, I learnt that the population size will have a great impact on GA's performance. As currently in every problem, I always set population size of GA and MIMIC to 200, I decide to explore a little bit.

For simplicity, I only focused on Four Peaks problem as this is the most complex optimization problem in my settings.

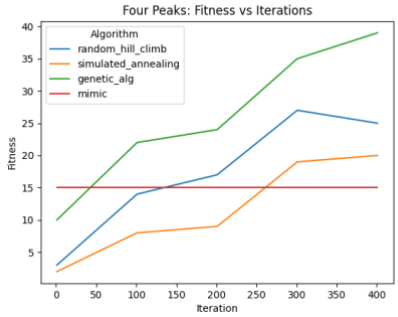
When population size is 10:

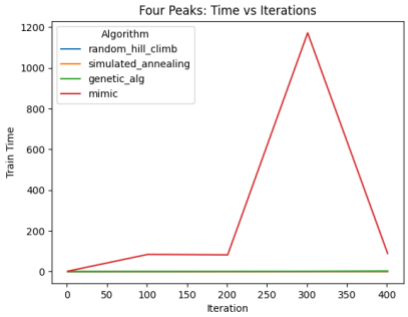




With halved running time, MIMIC cannot improve fitness score over increased iterations.

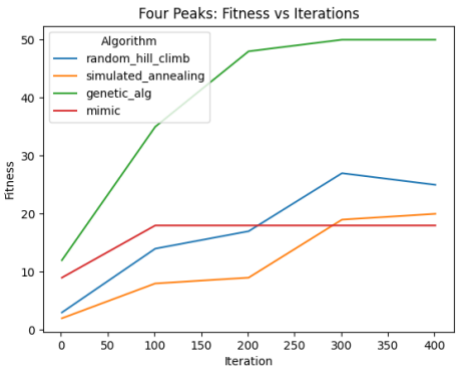
When population size is 50:

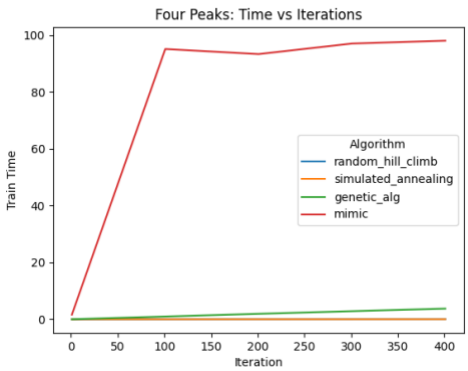




It is interesting to notice that MIMIC running time decreased a lot when the iteration goes to 400. I think this is purely out of chance. If we repeat this experiment several times using random states, we should observe another pattern.

When population size is 100:





Although the running time increased exponentially, the fitness score does not increase much for MIMIC. Instead, genetic algorithm's performance is brilliant in this case, given its perfect fitness score and barely increased running time.

4) Comparison

Combining the observation from neural network and optimization problem sets, we can find that different algorithm can become the best fit for different settings, with the change in problem complexity. However, none of the random optimization method can outperform our neural network using back propagation with decent gradient. The comparison is below:

Name	Computation Time	Num of Parameter	Characteristics
RHC	Short	0	Can efficiently solve simple problems
SA	Short	2	Similar to RHC
GA	Medium	3	Can efficiently solve complex problems, with relative short running time
MIMIC	Long	2	Can solve complex problems. But for simple questions, it will not be efficient