Randomized Optimization

Yichao Zhang
yzhang3414
CS7641 - Machine Learning

Mar 1, 2020

0. Introduction

In this report, there are mainly two parts of studies.

In the first chapter, we tested the performance of 4 optimizers: randomized hill climbing (RHC), simulated annealing (SA), genetic algorithm (GA), and MIMIC (MIMIC),
on 3 discrete value optimization problems: Travelling Salesman Problem (TSP), Continuous Peaks Problem (CPP), and Knapsack Problem (KSP).

In the second chapter, we applied RHC, SA and GA on a simple neural network with a single hidden layer.

The code can be downloaded from:
`https://github.com/yichigo/Randomized-Optimization`

1. Three Optimization Problems

Travelling Salesperson Problem (TSP), Should Highlight GA, But SA Won the Tests

TSP is a NP-hard problem. It provides a list of cities, and wants you to find a shortest path that go through each city and go back to the starting city.

In this problem, we generate 20 cities with 2D coordinate, where each value is a pair of floats among (0, 100).

We tuned some hyper-parameters of each optimizer. For SA, each decay schedule performed similar. For GA, we tested different mutation probabilities and populations, and the best performance is under population = 100 and mutation probability = 0.1. For MIMIC, we tested keep proportion between 0.1 and 0.5, and 0.1 performed better.

Fitness Curve

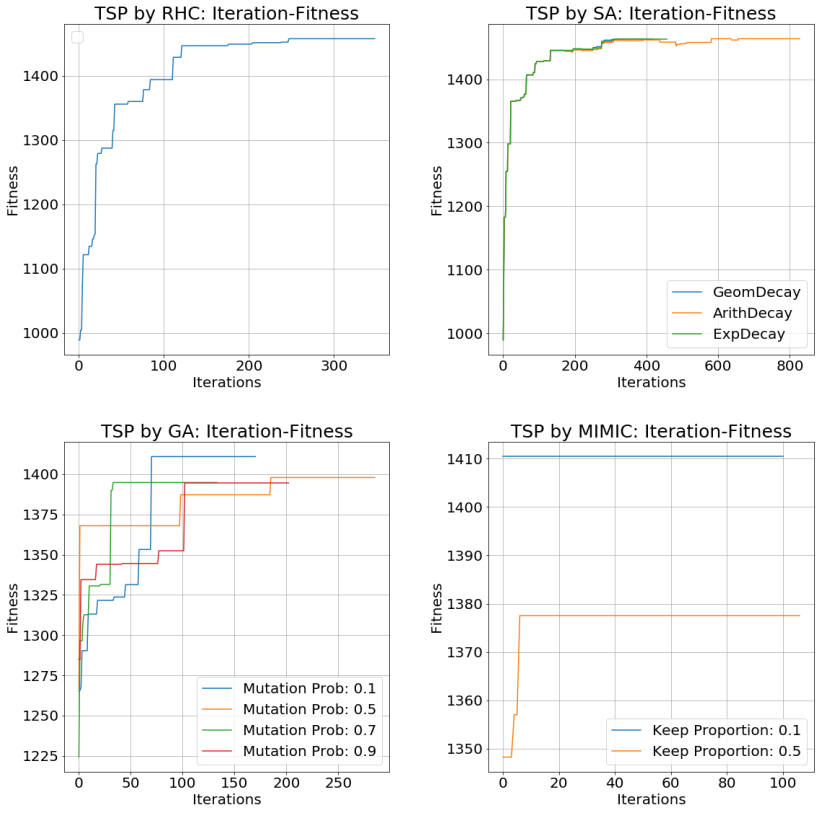

Figure 1: Fitness Curve of 3 optimizers: RHC, SA, and GA

From the fitness value in the fitness curve (Figure.1) of 4 optimizers, we can find that MIMIC cost the least number of iterations (< 10) to maximize the

fitness function. RHC, SA and GA all need about 100 iterations. However, RHC, SA and GA have higher fitness than MIMIC.

Compare Optimized Loss and Running Time

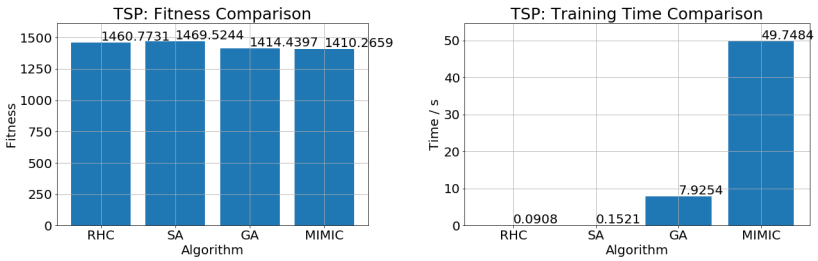

Figure 2: Compare the minimized loss function value and average training time

Figure.2 shows that both RHC and SA performed better and ran much faster than GA and MIMIC. The best optimizer SA maximized the fitness as high as 1469.5244 and spend for only 0.1521 second.

GA is well known for the best performance on TSP problem, it is not only best on the optimized fitness but also fast. However, in this testing, we found that both RHC and SA are better than GA in TSP problem.

At last, we found that there are some bugs in TSP and GA the mlrose package, it has been discussed in `https://piazza.com/class/k51r1vdohil5g3?cid=342` and `https://github.com/gkhayes/mlrose/issues/34`

Hopefully it can be fixed as soon as possible, then we can test the TSP problems again.

Continuous Peaks Problem (CPP), Highlight SA

CPP is a problem with many local optima. It provides a vector of N binary numbers, and let us to maximize the continuous bits.

In this problem, we generate 20 numbers in a vector, where each value is 0 or 1.

We tuned some hyper-parameters of each optimizer. For SA, GeomDecay schedule is fastest to converge. For GA, we tested different mutation probabilities and populations, and the best performance is under population = 100 and mutation probability = 0.1. For MIMIC, we tested keep proportion between 0.1 and 0.5, and 0.1 performed better.

Fitness Curve

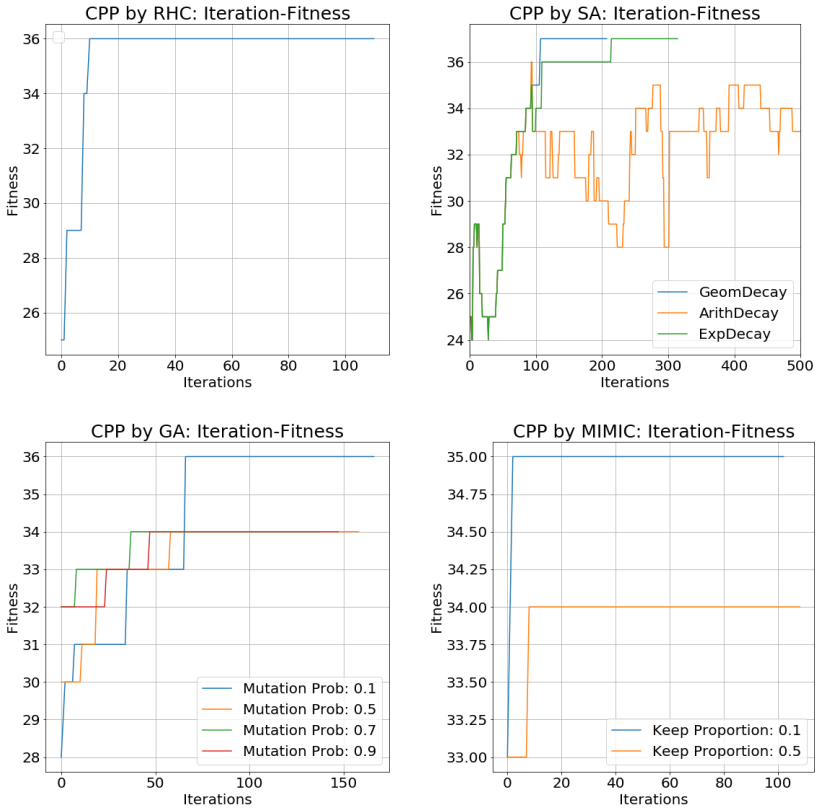

Figure 3: Fitness Curve of 3 optimizers: RHC, SA, and GA

From the fitness value in the fitness curve (Figure.3) of 4 optimizers, we can find that RHC and MIMIC cost less number of iterations (< 10) than SA

and GA. However, SA has the highest fitness.

Compare Optimized Loss and Running Time

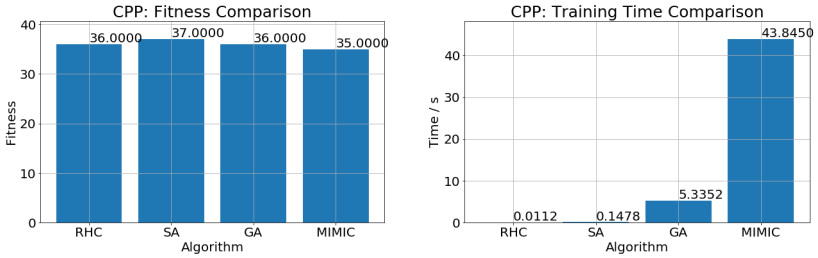

Figure 4: Compare the minimized loss function value and average training time

Figure.4 shows that both RHC and SA performed better and ran much faster than GA and MIMIC. The best optimizer SA maximized the fitness as high as 37 and spend for only 0.1478 second.

Knapsack Problem (KSP), Highlight MIMIC

KSP is a interesting problem that it provides a list of items, and each item has its weight and value. Our goal is to find a combination of items that the total weight is under a limit but the total value is as large as possible.

In this problem, we generate 20 items with random weights among (0,20) and random values among (0,100) coordinate. We set the max weight percentage as 0.6.

We tuned some hyper-parameters of each optimizer. For SA, each decay schedule performed similar. For GA, we tested different mutation probabilities and populations, and the best performance is under population = 100 and mutation probability = 0.1. For MIMIC, we tested keep proportion between 0.1 and 0.5, and 0.5 performed better.

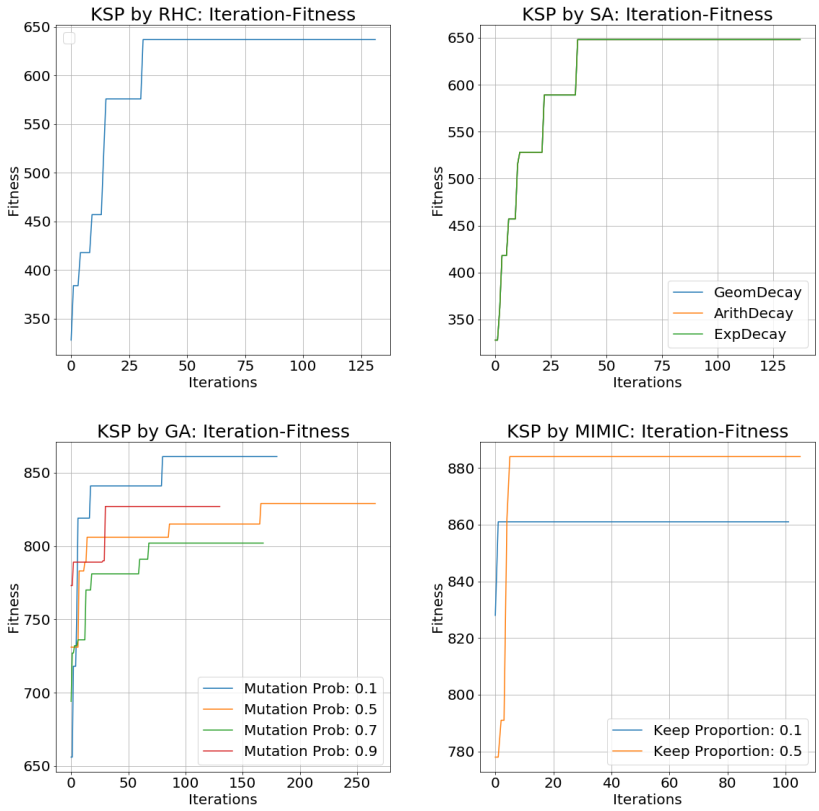

Figure 5: Fitness Curve of 3 optimizers: RHC, SA, and GA

Fitness Curve

From the fitness value in the fitness curve (Figure.5) of 4 optimizers, we can find that MIMIC cost less than 10 of iterations to reach the best fitness (> 880). GA also got a high fitness (> 850), while the fitness of RHC and SA did not hit 650.

Compare Optimized Loss and Running Time

Figure.6 shows that MIMIC got the best fitness (884) among 4 optimizers, and GA also reached a higher fitness than RHC and SA. However, MIMIC spend much longer time (21 seconds) than others.

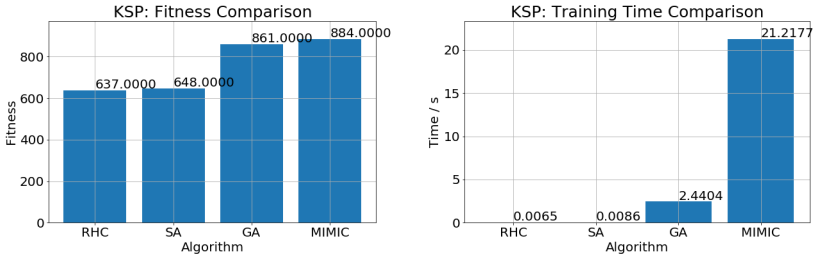

Figure 6: Compare the minimized loss function value and average training time

2. Neural Network Optimization

Heart disease can be predicted from some simple medical measurements. It helps people save a lot of time to treat the disease rather than waiting for the busy doctors and suffering the disease without any treatment.
We use the data set heart.csv from: `https://www.kaggle.com/ronitf/heart-disease-uci`
There are 13 features: age, sex, cp (chest pain type), trestbps (resting blood pressure)..... and 1 target with value 0 or 1. We randomly split the dataset with training size 80% and testing size 20%.
We use a single hidden layer neural network by MLPClassifier from scikit-learn to train a predictive model. The size of the hidden layer has been optimized as 40, with identity activation function following by the hidden layer.

For SA optimizer, we tried 3 decay schedules, where GeomDecay and ExpDecay converge much faster than ArithDecay.

For GA optimizer, we tried different mutation probabilities, and the default mutation rate 0.1 is the fastest one to converge, by 50 iterations.

Fitness Curve

From the loss function value in the curve (Figure.7) of 3 optimizers, we can find that GA cost the least number of iterations (< 300) to minimize the loss function. RHC needs about 3000 iterations, and SA needs the most, about 3500 iterations.

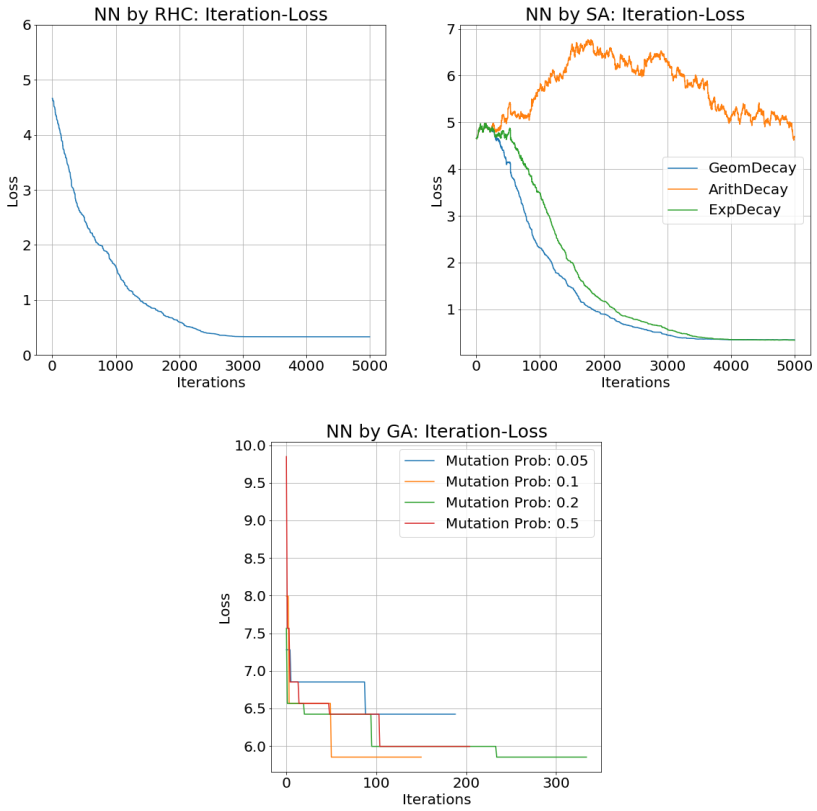

Figure 7: Fitness Curve of 3 optimizers: RHC, SA, and GA

Compare Optimized Loss and Running Time

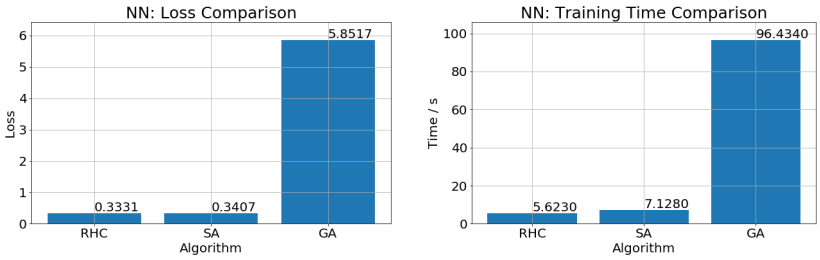

Figure 8: Compare the minimized loss function value and average training time

Figure.8 shows that both RHC and SA perform much better and run much faster than GA. The best optimizer RHC minimized the loss as low as 0.3331 and ran for only 5.6 second.

3. Conclusion

Based on our results, RHC and SA are much faster than GA and MIMIC, and MIMIC is even much slower than GA.

RHC works on many simple problems, however, RHC is not suitable for problems with many local optima.

SA, GA and MIMIC have ability to jump out of local optima. And SA is the fastest choice. GA is the best for some problems like TSP. (Although GA is not as good in our test due to the problem of mlrose package). MIMIC could have the best fitness in some complex problems, however, it is very slow.