CS7641 Machine Learning Assignment 2

Randomized Optimization

Student Name: Qianwen Shi (qshi39) Date: Mar 1st, 2020

Introduction

The purpose of this assignment if to explore four random search algorithms, including randomized hill climbing, simulated annealing, genetic algorithm and MIMIC. This assignment consists of two parts, part 1 is solving three optimization problems using four random search algorithms and part 2 is solving neural network weight optimization problem using the first three random search algorithms.

Randomized Hill Climbing (RHC)

Randomized hill climbing is a local search mathematical optimization technique which does hill climbing iteratively by starting at a random initial state and move towards better fitness neighboring points until there is no more improvement[1]. This algorithm can be very effective and uses little memory. The challenges of this algorithm include converge on local maxima instead of finding global maximum, ridges and alleys problem, or plateau problem.

Simulated Annealing (SA)

The simulated annealing algorithm is an optimization method that mimics the slow cooling of metals to approximate the global optimum of a given function in a large search space[2]. The key parameters are temperature and decrease rate, the former parameter determines the probability of accepting a worse solution and the latter decides how fast the temperature decreases over running period. The advantage of this algorithm is that it can escape the local optimum but can be slow for some problems.

Genetic Algorithm (GA)

Genetic algorithm belongs to evolutionary algorithms, which mimics natural selection process to find optimization solutions. It is based on the biological mutation and crossover of two parent chromosomes to generate a new offspring in the population[3]. Best individuals are selected using a fitness function whose result indicates the quality of the solution[3]. GA can be a robust algorithm but is also a slow technique due to its complexity.

MIMIC

Mutual-Information-Maximizing Input Clustering (MIMIC) uses estimation of probability densities to find the optimal solution. MIMIC first samples from regions of the input that most likely contain the minimum for cost function, then uses an effective density estimator to capture a wide variety of structure of the input space[4].

Part 1 Three Optimization Problems

1. Travelling Salesperson Problem (TSP)

Travelling salesperson problem is an interesting and classic NP-hard problem. In this experiment, the goal is set to find the **longest** travel distance for the salesperson to visit all the cities (each city only once). This problem can be widely applied in transportation and travelling routes in the real world. I used python to generate 30 unique cities coordinates and then used mlrose_hiive to perform the experiment following the guide in the mlrose manual[5]. The fitness of each algorithm represents the total distance traveled therefore the

higher the fitness the better the result. In mlrose, the fitness curve stops when no more improvement found hence on the graph the length of fitness curves is different.

Different key parameters for GA, SA and MIMIC were tested and the results of the fitness curve vs number of iterations are presented in Fig 1 – 4. For GA, based on Fig 1 the best result was found when population size set to 200 and mutation probability set to 0.5. At same population size, the fitness seemed to decrease as mutation probability increased. For SA, the initial temperature was tested with decay rate kept at 0.55 (Fig 3), then different decay rates were tested with initial temperature set to 10 (Fig 4). For TSP optimization, the effect of changing initial temperature or decay rate is not obvious. As for MIMIC, different keep_pct (proportion of samples to keep at each iteration) values are tested with population size set to 200. Figure 4 shows that pct set to 0.5 performs best while pct set to 0.1 performs the worst for TSP.

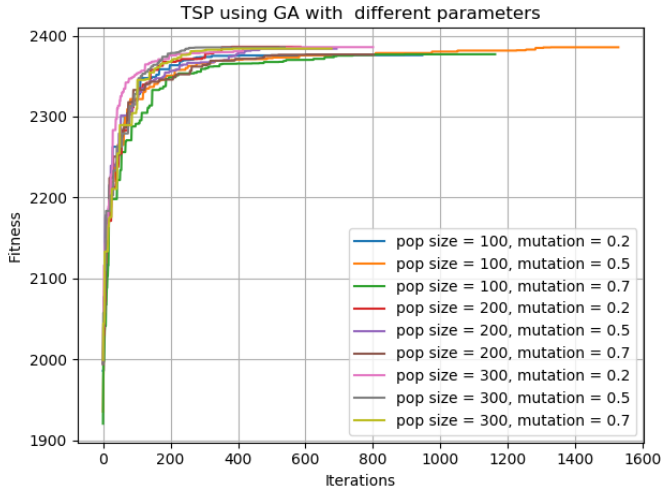

Fig 1

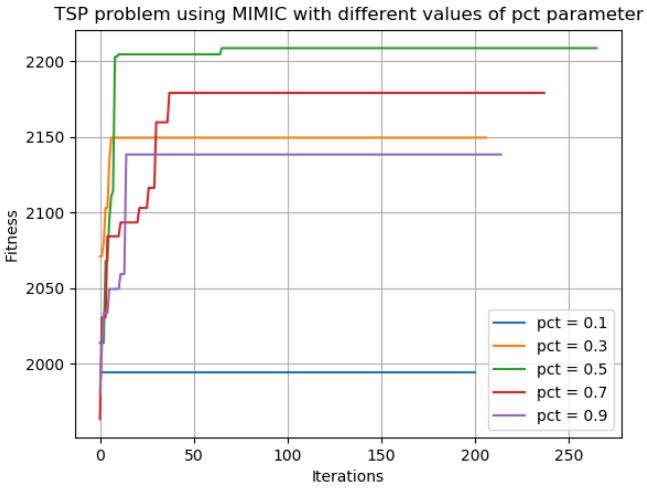

Fig 2

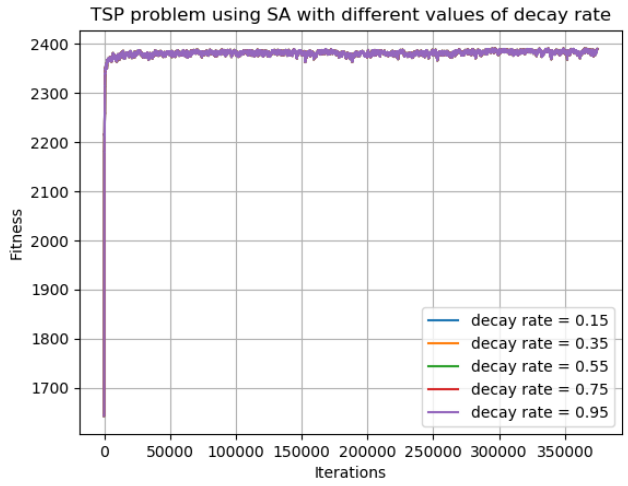

Fig 3

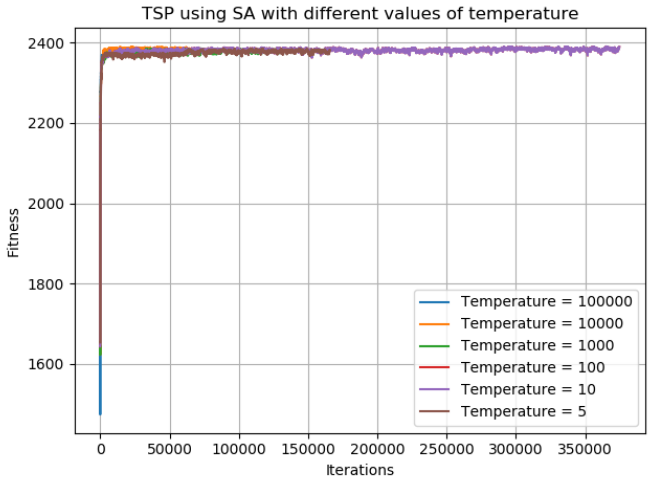

Fig 4

Using the best solution parameters, the fitness of four random search algorithms was compared in Fig 5. It is clear that GA outperformed other three algorithms as it reached highest fitness after 400 iterations. Although MIMIC quickly converged within 50 iterations, but its fitness remained the lowest. RHC and SA fitness gradually climbed up as iterations increased and converged after1200 iterations at a similar fitness GA achieved. Fig 6 compared the running time of these four algorithms for TSP and MIMIC took the longest time, followed by GA, while RHC and SA were much faster only needed less than a second.

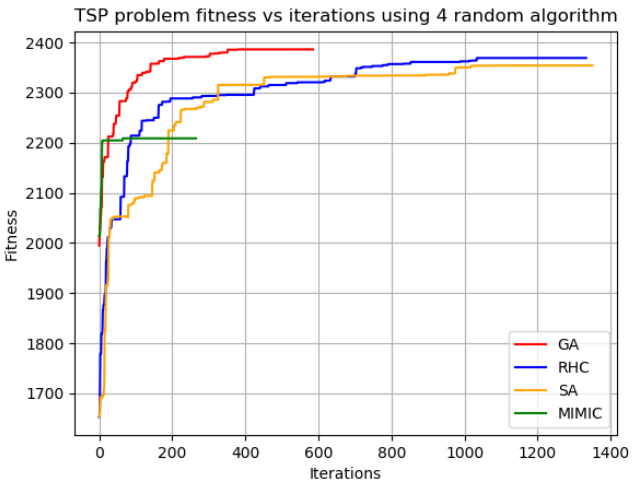

Fig 5

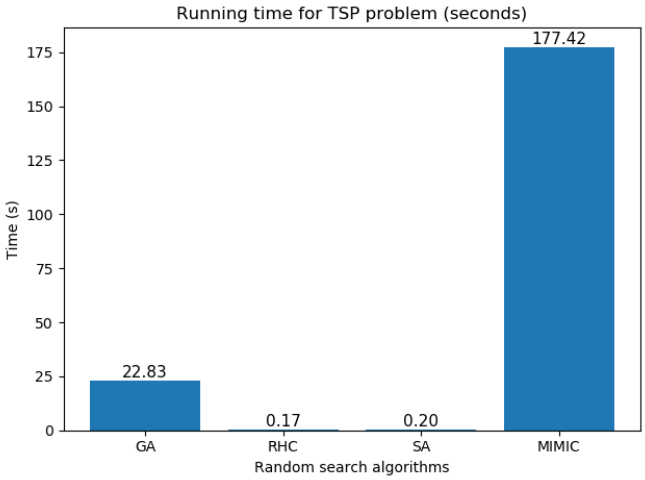

Fig 6

2. Continuous Peaks Problem (CPP)

The second investigated optimization problem is the continuous peaks problem, which is derived from four peaks problem. The goal is to find the global maxima (highest peak) with respect to local optima (other peaks) The experiment was performed using the mlrose fitness function ContinuousPeaks()[6].

The fitness of GA with different population size and mutation probability was presented in Fig 7. All cases converged after 50 iterations with population size 200 and mutation rate 0.5 performed best. Fig 8 presented the fitness change over iterations as decay rate increased (initial temperature set to 10). Fig 9 showed the fitness variation with different initial temperature when decay rate set to 0.55. It is a bit hard to select best parameters as the fitness fluctuated. The parameters used for algorithms comparison were initial temperature set to 1 and decay rate 0.99 (default setting). Fig 10 showed keep_pct parameters effects on fitness when population size kept at 200. The keep_cpt set to 0.7 gave much higher fitness than other values.

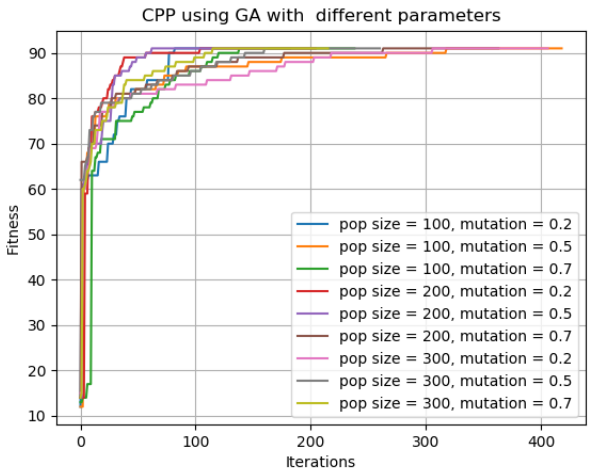

Fig 7

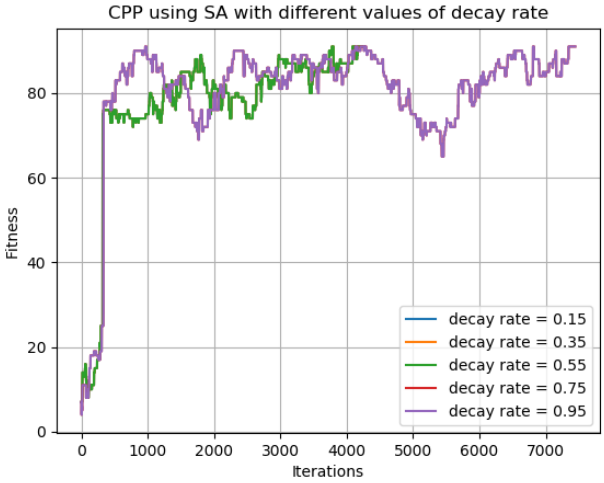

Fig 8

According to Fig 11, both GA and SA reached optimal fitness eventually (GA within 100 iterations while SA after 600 iterations). RHC was the worst for slow convergence and low fitness value. MIMIC performed slightly better than RHC as it converged to a higher fitness value within 10 iterations. Looking at running time for algorithms in Fig 12, MIMIC was the longest and RHC the shortest. Although both GA and SA reached converged to same fitness value, GA was 100 times slower than SA. Therefore for continuous peaks problem, SA was the best algorithm as it could provide good solution in short time.

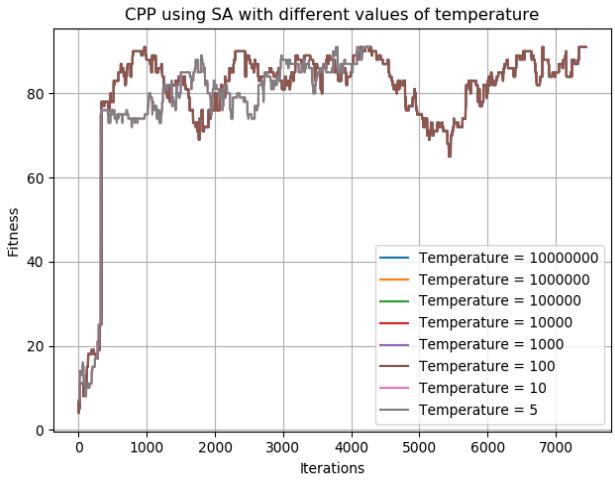

Fig 9

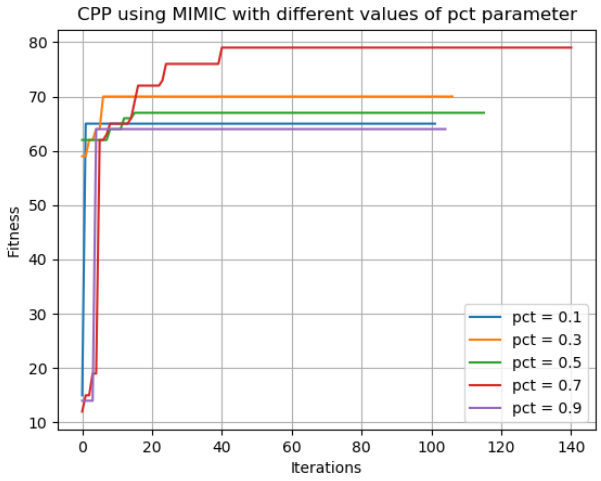

Fig 10

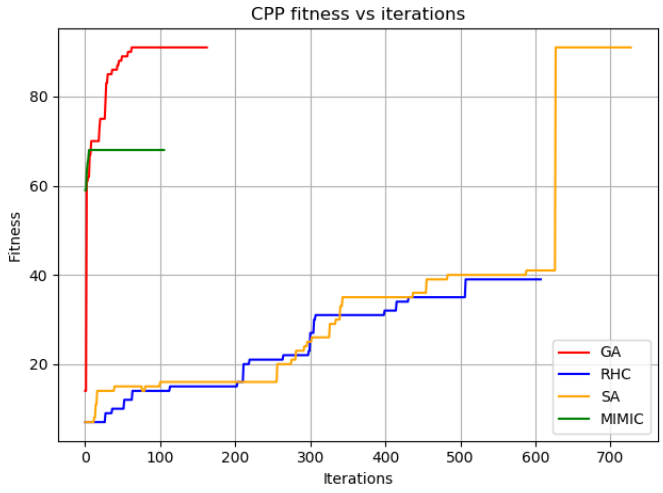

Fig 11

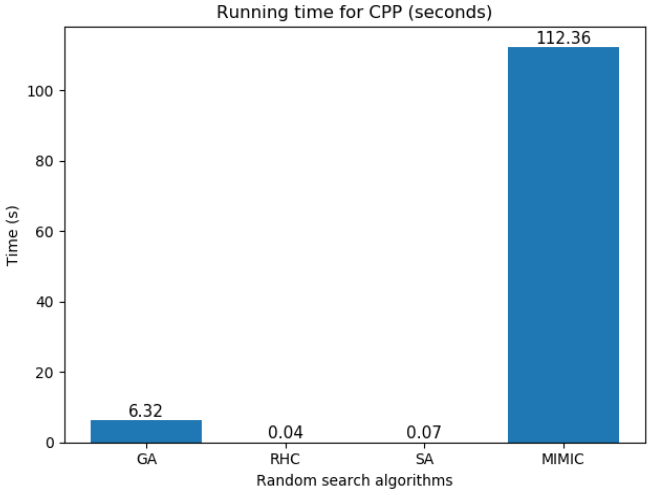

Fig 12

3. Knapsack Problem

Knapsack problem is the third optimization problem I am interested in. The problem is that given several items with different values and weights, determine the best solution of what items to put in the bag (with limited total weights allowance) to reach maximum total values. I randomly generate 40 items with weights between (1, 30) and values between (1, 500) and the bag weight allowance set to 60% of the total weights of all items. In this problem, the fitness was the total value of the items put in the bag so we want to maximize it. Then I used the mlrose_hiive[7] to perform the experiment with results summarized in Fig 13 – 16.

The fitness of GA with different population size and mutation probability was presented in Fig 13. All cases converged after 50 iterations with population size 300 and mutation rate 0.7 performed best. Fig 14 indicated the fitness change over iterations as initial temperature changed (decay rate set to 0.95). Initial temperature at 100000 reached to the highest fitness after around 150 iterations. Based on Fig 15, when keep_pct set to 0.3, the fitness quickly converged to highest value around 10 iterations. Then I investigated different population size effects on fitness with keep_pct set to 0.3 (Fig 16). Population size of 400 and keep_pct of 0.3 were used for algorithm comparison.

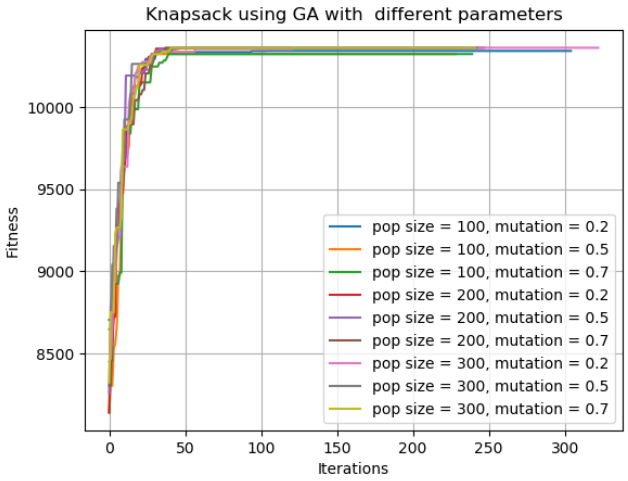

Fig 13

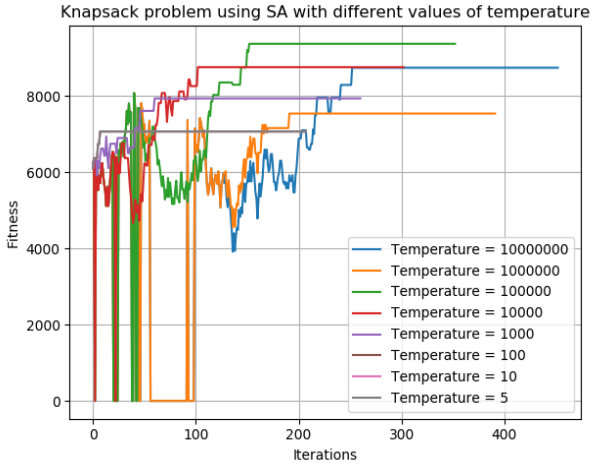

Fig 14

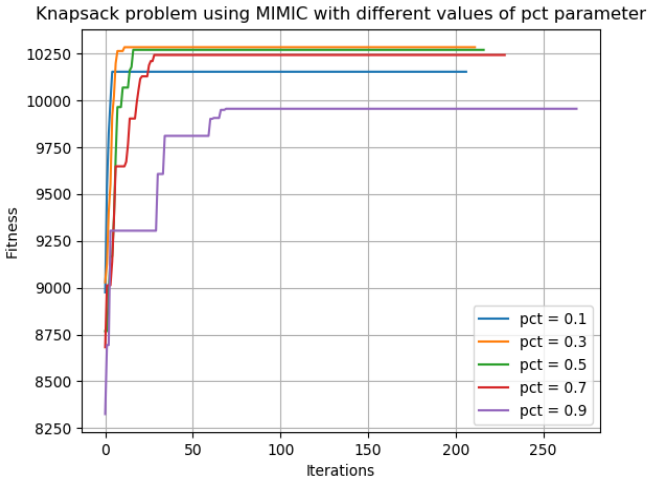

Fig 15

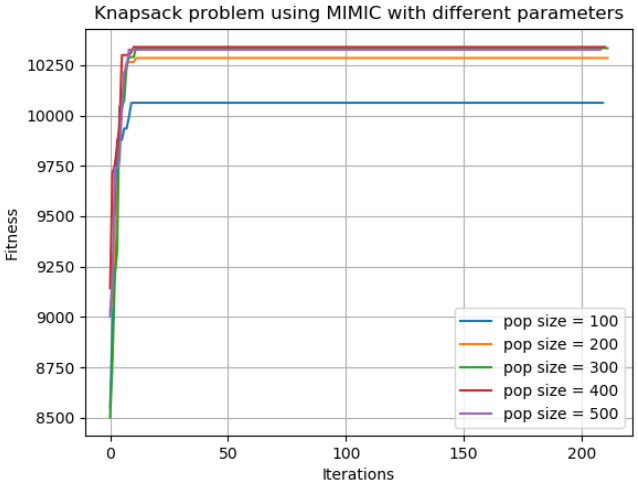

Fig 16

Using the above best parameters, the fitness among different algorithms was compared and summarized in Fig 17. Both GA and MIMIC reached highest fitness at small iterations number with MIMIC converged much earlier. RHC performed the worst as the fitness value when converged was quite low. SA performance was better than RHC, but the converged fitness was still not comparable to that of GA and MIMIC In terms of the running time (Fig 18) MIMIC took much longer compared to GA, RHC and SA. In terms of reaching optimum at low iterations and running time, MIMIC still outperformed the other three algorithms.

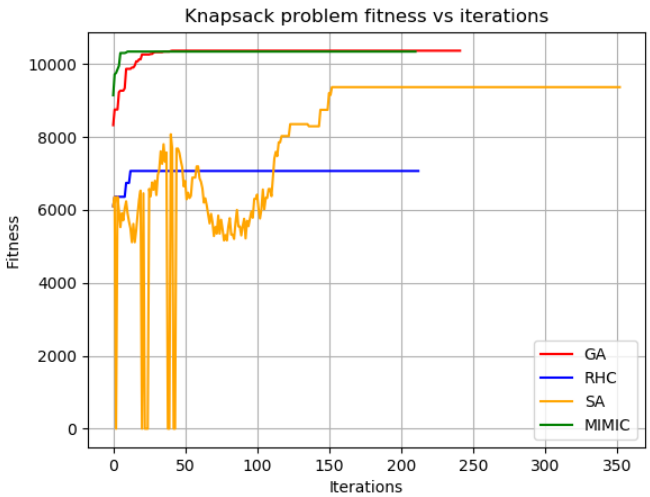

Fig 17

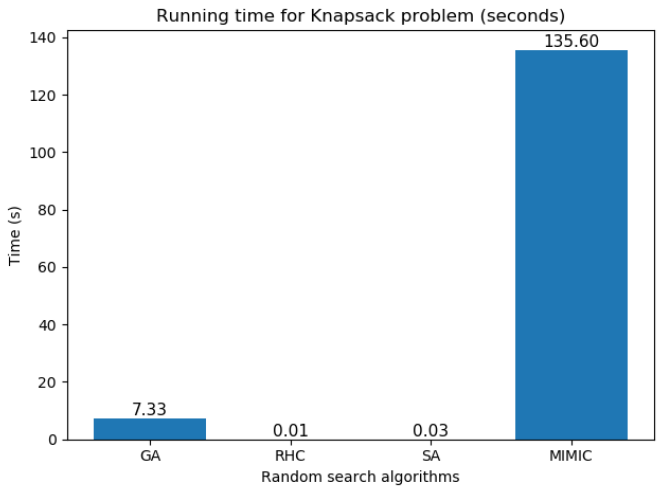

Fig 18

Part 2 Neural Network (NN) Weight Optimization

For this part of the assignment, the objective is to use above 3 random search algorithms (RHC, SA and GA) to optimize the neural network weight, which means fitting the weight parameter to minimize loss function for a given training dataset. The dataset I used is the wine quality dataset with 12 attributes and 1599 instances. The 12th attribute "wine quality" is preprocessed to 0 (quality below 6) or 1. The data split is 30% for testing and 70% for training. The experiment was performed using mlrose_hiive following the guide of weight optimization problem[8]. Since I didn't normalize data for assignment 1, I used the gradient descent as the baseline in this part. After testing different settings, to reach lower loss and higher accuracy score the gradient descent neural network was set as: one hidden layer with 50 nodes, activation function set to 'tanh' and learning rate set to 0.001. This is the same settings I used in Assignment 1 and accuracy score results are similar.

When applying the above setting to test three random search algorithms, the accuracy scores were pretty low, usually below 50%. Hence, I have tested different parameters and improve the accuracy scores with following neural network settings: 3 hidden layers, each had 10 nodes; activation function set to 'relu'; and learning rate set to 0.15.

The NN weight optimization fitness curve vs iterations of these 4 algorithms was summarized in Fig 19. In this experiment, the loos function was the one we are trying to minimize, so the fitness here represented loss (the closer it was to zero the better). The baseline gradient descent and GA both quickly converged within 20 iterations, but the fitness of baseline was closer to zero. SA and RHC eventually reached the same fitness as GA but need more iterations (RHC after 200, SA after 250). Therefore, in terms of fitness, gradient descent slightly outperformed other algorithms.

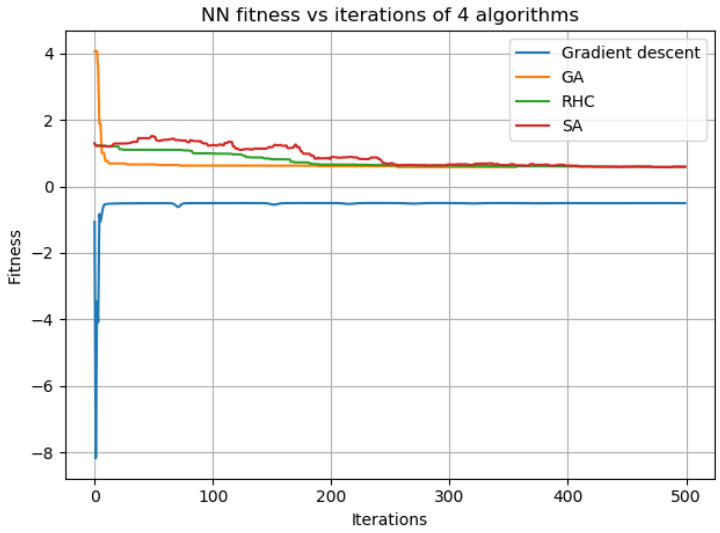

Fig 19

The accuracy score was used to check the model performance and the results for training data and testing data among different algorithms were illustrated in Fig 20 and 21. The gradient descent algorithm had the highest accuracy score, GA and SA had very similar accuracy scores. The training and testing accuracy scores for RHC were around 10% lower compared to other three algorithms hence the worst performance for this dataset. However, we also need to consider the running time for algorithms to see which one is better.

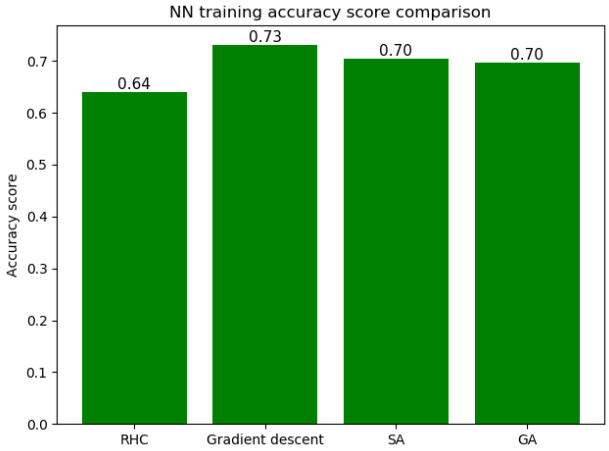

Fig 20

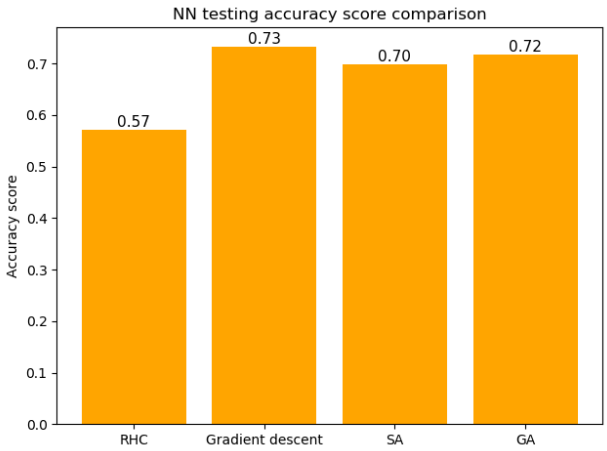

Fig 21

Fig 23 summarized the running time for 1000 iterations. Not surprisingly, GA was the slowest (680 s) due to its complexity. Despite GA had good accuracy score, the running time was too long, so GA was not an efficient model for this experiment. RHC and SA had similar run time as SA had better accuracy score so SA outperformed RHC. The gradient descent only took 1.6 seconds and resulted in high accuracy therefore gradient descent was the best algorithm for neural network weight optimization in this case. Among GA, SA and RHC, RHC and SA were better algorithms for wine quality when taking into consideration of both accuracy and running time. Since the weights in neural network are continuous not discrete, and from part 1 it was found that SA worked best for continuous problems, so it worked better than RHC and GA for weight optimization.

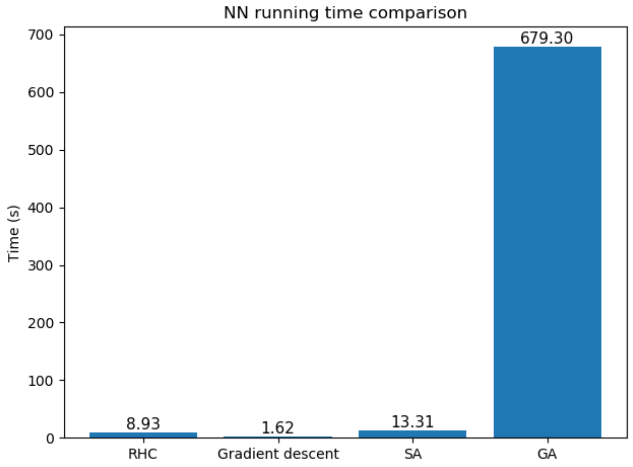

Fig 22

Summary

In this assignment, four random optimization algorithms (RHC, SA, GA and MIMIC) were explored using 3 different optimization problems. Each algorithm had its advantages and disadvantages over different optimization problems. The strength and important parameters of four algorithms were summarized in Table 1[9]. As for neural network weight optimization problem using wine quality data, SA was the best algorithm as it fits continuous problems better.

Algorithms	Parameters (mlrose_hiive)	Running Time	Problem Best Solved	Strengths
RHC	None	Shortest	Wine Quality	No need of parameters tuning, easy and fast
SA	Initial Temperature Temperature Decay Rate	Relatively Short	Continuous Peaks Wine Quality	Good at approximating global optima
GA	Population Size Mutation Probability	Longest	Traveling Salesperson	Good for complicated problems
MIMIC	Population Size Keep_pct	Relatively Long	Knapsack	Good for complicated problems and approximating global optima

Table 1. Summary of random optimization algorithms

References:

1. https://en.wikipedia.org/wiki/Hill_climbing
2. https://www.sciencedirect.com/topics/materials-science/simulated-annealing
3. https://www.sciencedirect.com/topics/immunology-and-microbiology/evolutionary-algorithm
4. J. S. De Bonet, C. Isbell, P. A. Viola, MIMIC: Finding Optima by Estimating Probability Densities, Computer Science, NIPS1996
5. https://mlrose.readthedocs.io/en/stable/source/tutorial2.html
6. https://mlrose.readthedocs.io/en/stable/source/fitness.html?highlight=knaps#mlrose.fitness.Knapsack
7. https://mlrose.readthedocs.io/en/stable/source/fitness.html#mlrose.fitness.ContinuousPeaks
8. https://mlrose.readthedocs.io/en/stable/source/tutorial3.html
9. https://medium.com/@duoduoyunnini/introduction-implementation-and-comparison-of-four-randomized-optimization-algorithms-fc4d96f9feea#_ftn1