CS 7641 Assignment 2 Randomized Optimization

Introduction

For this assignment, we tested different optimization algorithms covered in the class lectures and investigated how each algorithm performs on problems with different complexities. The optimization algorithms will also be applied to the heart disease prediction dataset from assignment 1 to test how well each optimization algorithm performs at finding the optimal weight for ANN.

Four optimization algorithms we covered in the report are shown below:

Random Hill Climbing(RHC) - This algorithm randomly picks starting points and maximizes fitness by climbing to the next neighboring optima. This algorithm can be easily stuck in a local optima between iterations. However, the added randomized starting point is aimed to increase the probability of climbing to a global optima.

Simulated Annealing(SA) - Similar to RHC, atTemperature component is added to SA to determine whether or not it should move to the next higher neighboring point. This algorithm allows the opportunity of exploring instead of exploiting all the time. At low temperature, this algorithm acts exactly like RHC. However, when the temperature is high, it exhibits the behavior of random walk.

Genetic Algorithm(GA) - A better fit population is used to produce offsprings to replace the least fit population. The goal is that every iteration will produce a better population which is very similar to evolution.

Mutual Information Maximizing Input Clustering (MIMIC) - An initial sample is drawn from an uniform distribution. A density estimator is being used to generate more samples as well as being improved by the generated samples. This algorithm is suitable for problems with complicated structures and is the only algorithm out of the four that remembers the learning history.

Part 1 Optimization Algorithm Exploration

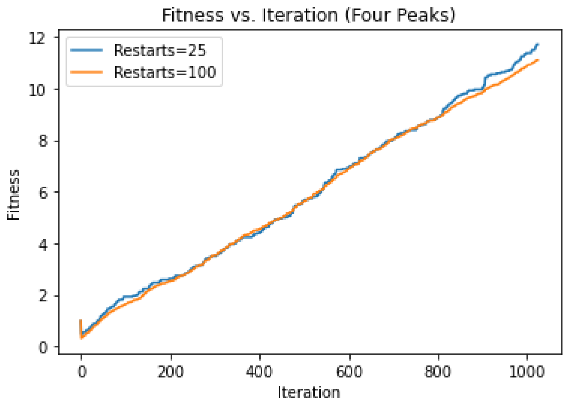
For the first part of the assignments, four optimization algorithms will be applied to 3 different problems to demonstrate their ability at finding the optimized solution (global optimal). The goal of the assignment is to demonstrate the strength and weakness of each algorithm. We also found that there is not one best optimization algorithm. Depending on the complexity and the structure of the problem, an easier algorithm might outperform a more sophisticated algorithm.

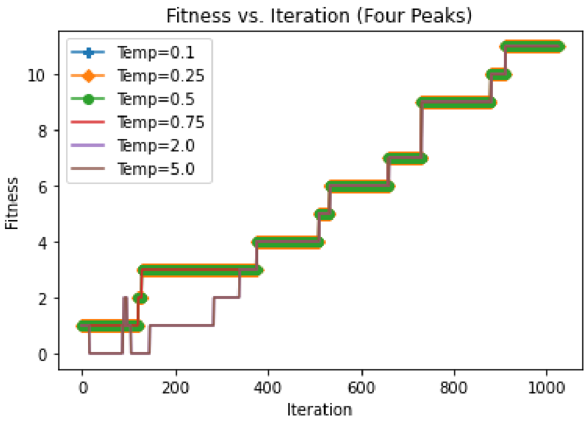
Problem 1 Four Peaks (Highlighting GA)

Four Peaks Fitness function is the calculating sum of the number of leading 1 and tailing 0 and length of the problem. Length of problem is only added to the sum calculation if the number of tailing 0 and leading 1 meets the threshold. Therefore this problem has 4 optimals and is slightly more complex than one max problem which is just simply the count of sums. Simply having 1s or 0's will reach local optima but a needed combinations of 1 and 0's are needed to reach the global optima.

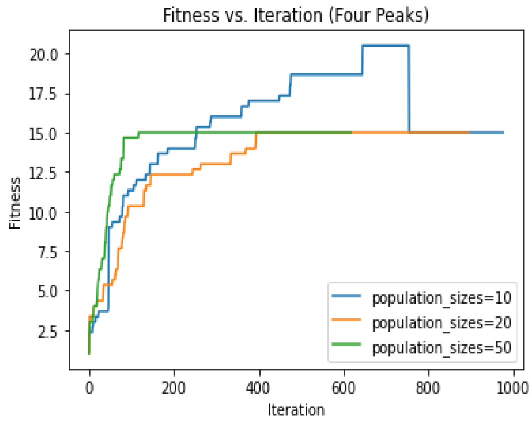
We first evaluated and tuned the following parameters for each optimization algorithm.

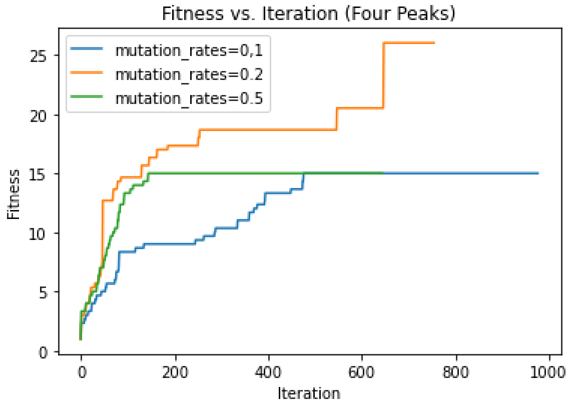
- RHC- Restarts
- SA- Temperature
- GA- Population Size & Mutation Rate
- MIMIC- Population Size & Keep Percentage





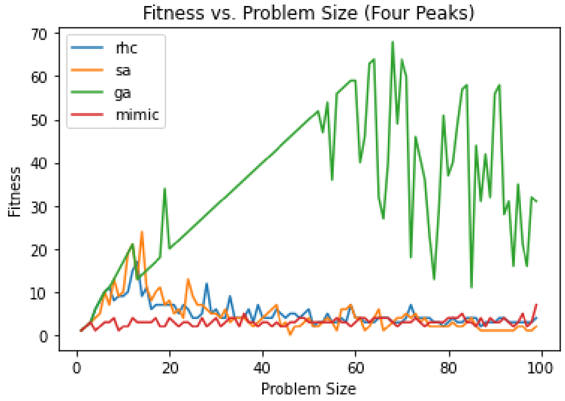
As shown above, the temperature did not impact the final fitness values found by RHC.But one interesting thing found was that when the temperature was the highest (similar to random walk), the algorithm produced different fitnesses than other temperature settings. Some of the fitnesses at earlier steps were better and some worse due to the random walk. This shows the differences between RHC and SA and how added exploration can sometimes produce better results than just exploiting.

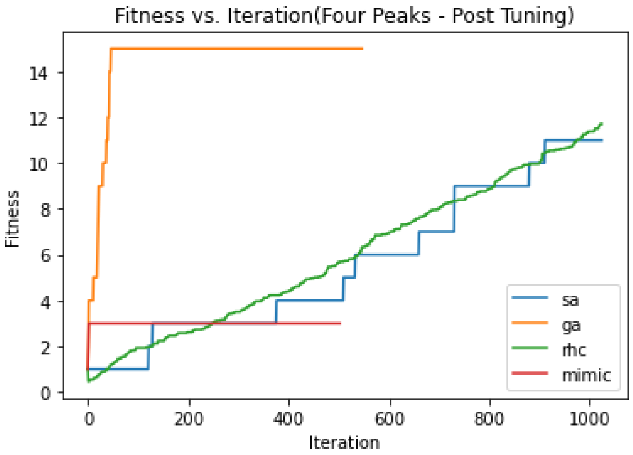




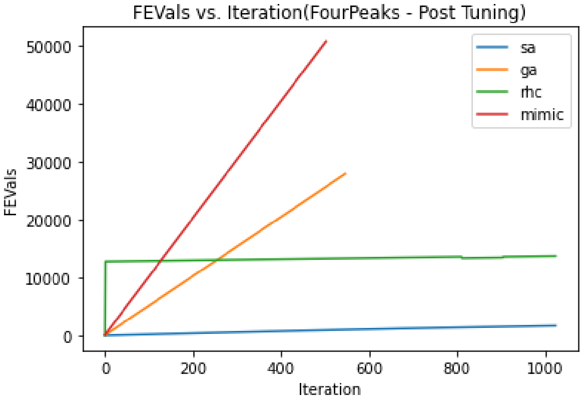
GA algorithms were tested with varying population size and mutation rates. The average of fitness for each parameter at each iteration was calculated and plotted on graphs above. GA was able to converge on every variable setting. However, higher populations were able to converge much faster because the training was more aggressive by allowing more population to be replaced. Something I did not expect to see was that the smallest population(10) actually performed better at an earlier stage. But sadly, then it was overfitted and caused the fitness to decrease as more iterations were being run .

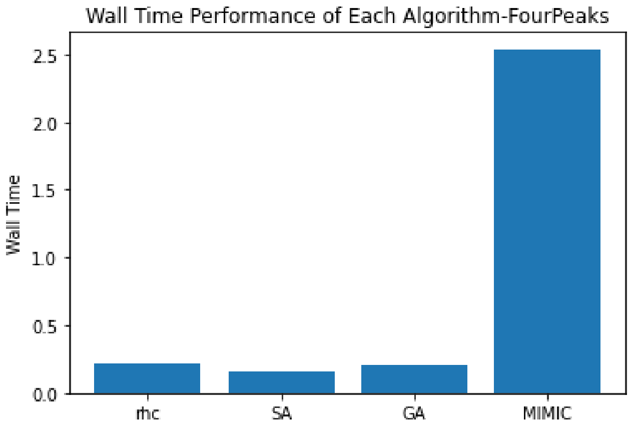
Using the fine tuned parameters, we also investigated how the problem size can impact the performance of each algorithm. As the size increases(>20), GA algorithm started to outperform other 3 algorithms significantly despite the signs of instability in evaluating the global optima which can be improved by fine tuning the algorithm further.





We also compared the performance of these 4 algorithms on four peaks with problem size of 20 on the right side. As expected, MIMIC was the first one to converge and stuck at the local fitness of 4. This shows one characteristic of MIMIC- quick converging which is helpful with reducing the cost especially when the cost function is complex to perform. GA was able to converge early as well with the best performance (WINNER!). Both RHC & SA both slowly improved but did not converge even after 1000 iterations due to presentence of local optimals.

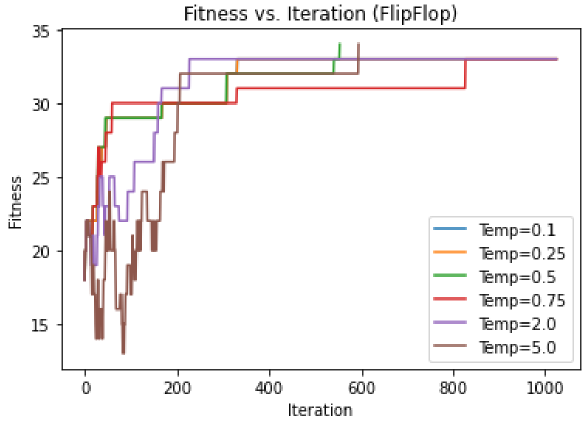


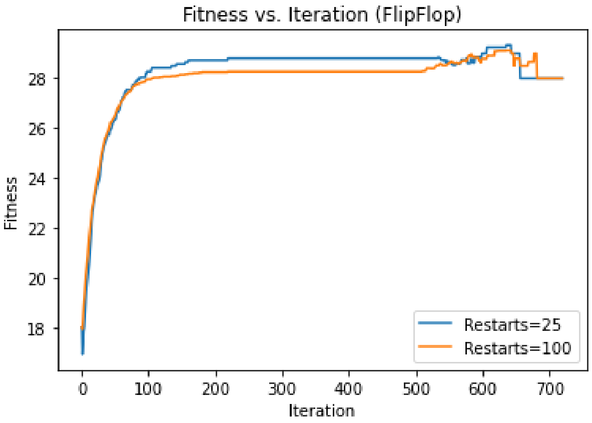


Both FEvals and Wall Time performance were also compared for these 4 algorithms . It shows that the MIMIC algorithm requires significantly higher number FEVals as iteration increases and takes the longest to finish despite being the first one to converge. RHC has higher FEVals increase and takes longer to run compared to SA.

Problem 2 FlipFlop (Highlighting SA)

FlipFlop problem is a simple count of consecutive pairs where each pair has alternative numbers. We were expecting that the simpler algorithm (RHA, SA) would perform better in this problem.

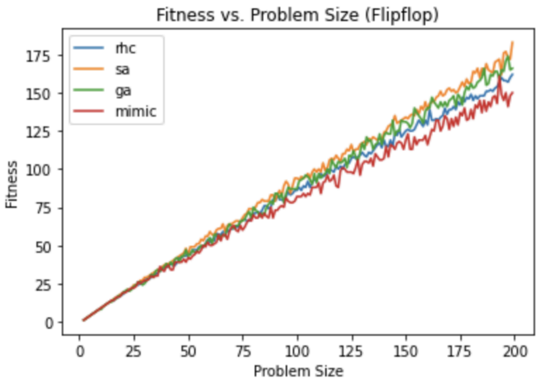


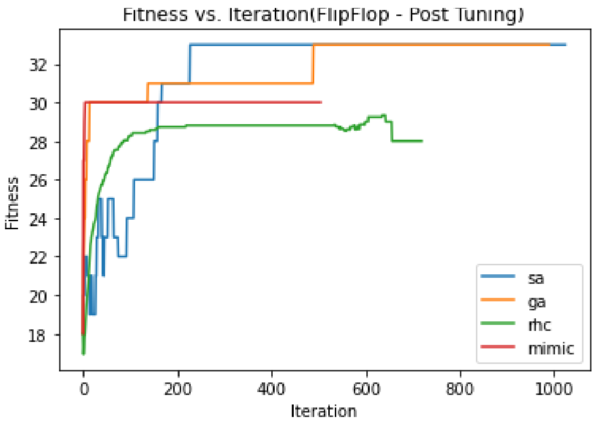


Using different temperatures did not change how fast the algorithm was able to converge. Lower temperature did allow faster convergence because random walks occur and take longer for the result to stabilize when temperature is high.

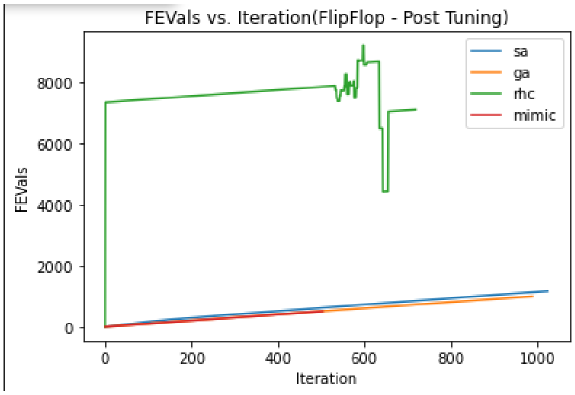
The tuned algorithms were then applied to different problem sizes to investigate how problem size can impact the performance of each algorithm. The performance of each algorithm did not get impacted by the problem size due to the simplicity of the problem.

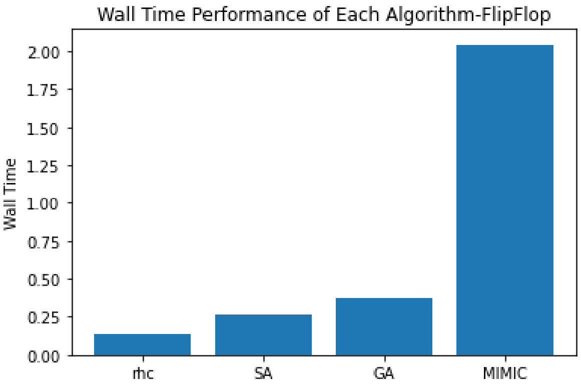
Both SA and GA were able to reach the global optimals. MIMIC was still the first one to converge and stuck at 30. One weakness GA showed was that it requires significantly more iterations to improve when the fitness is closer to the global optimal.





Despite how simple the question is, RHC still required more FEVals at each iteration as shown below. But due to algorithm simplicity it is still able to finish the fastest. MIMIC, despite being the first one to converge, still took the longest to finish. Flip Flop is the simplest structured problem out of the three problems, so it makes sense that the MIMIC algorithm was an overkill for this problem.

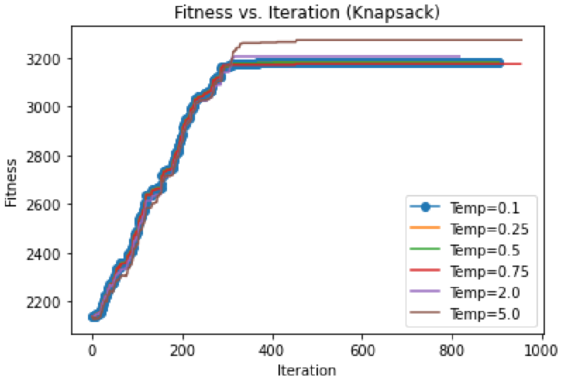


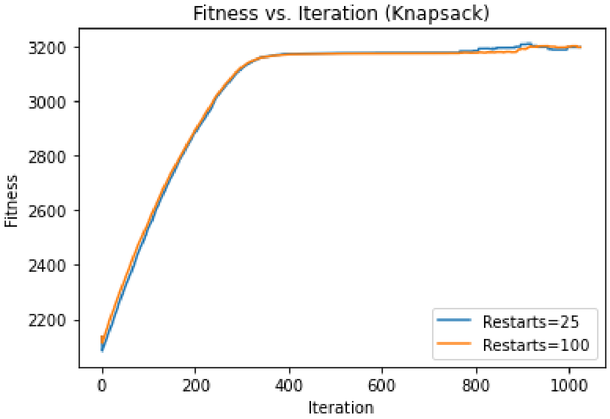


Problem 3 KnapSack (Highlighting MIMIC)

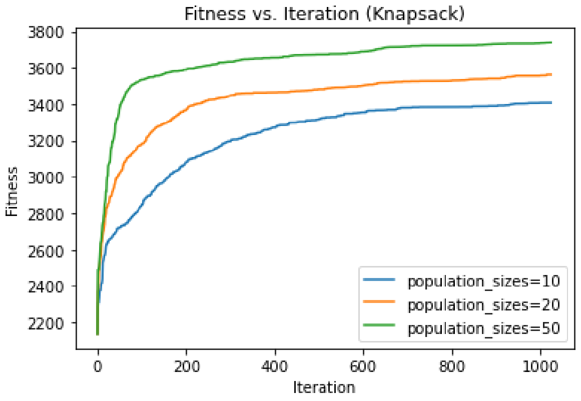
I consider Knapsack to be the most complex structured problem among the three. It requires a balance of multiplying values and state vectors, with the additional condition that the sum of the weight and state vector multiplications can not exceed the maximum capacity (W).

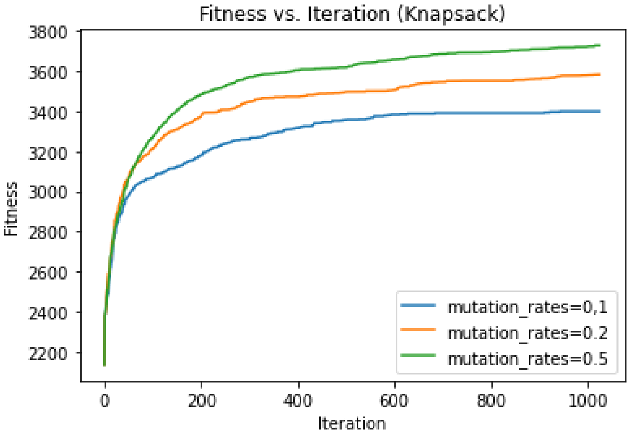
The temperature that performed best was the highest temperature (5.0) which makes sense since the cost function is harder to calculate and has more local optimals which makes RHC harder to reach to global optima. With random walk, it actually was able to reach optima faster by randomly moving directions.



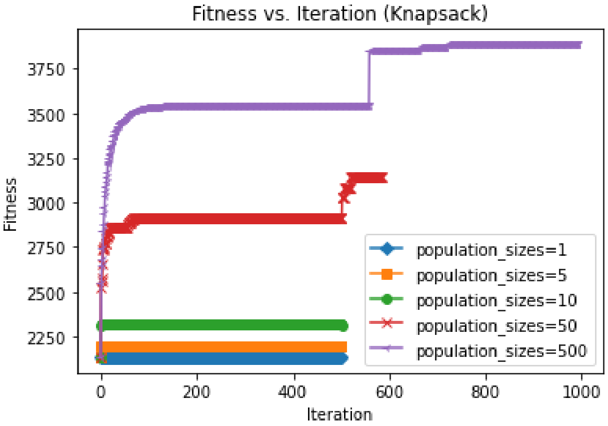


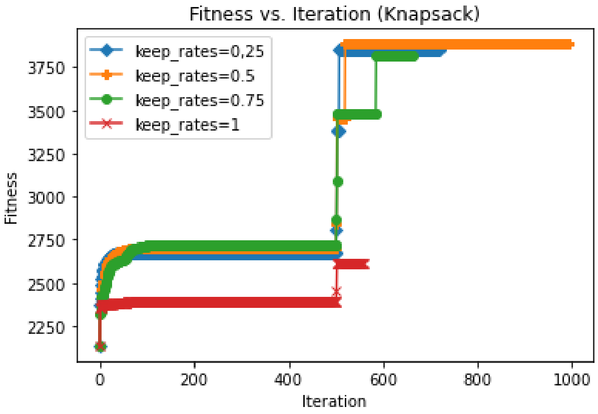
Since Knapsack is considered a more complicated problem I was expecting that GA and MIMIC to show their strength here. As shown on plots below, altering parameters for GA does show huge improvements in fitness which we did not see in previous 2 problems. GA was performing the best with the highest population and mutation rate for Knapsnap.



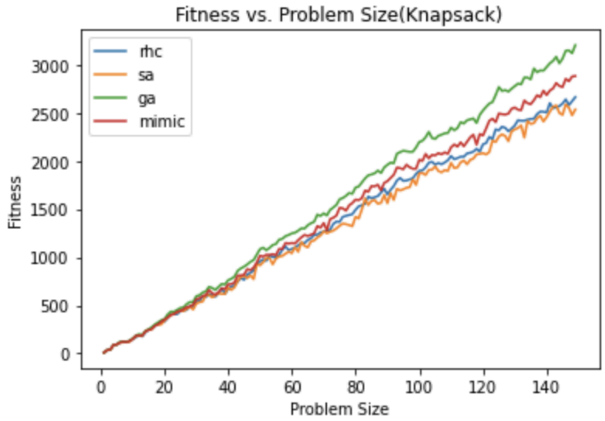


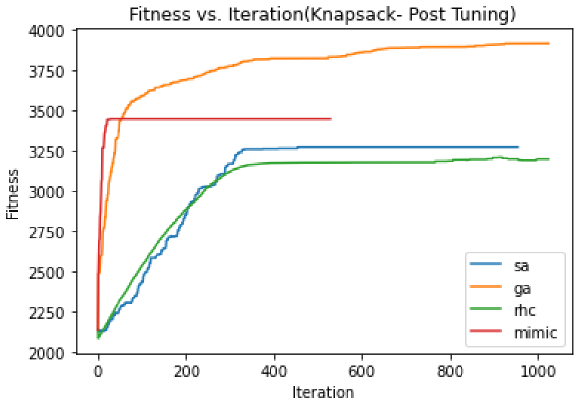
Similar to GA, MIMIC also performed the best when the population size was the highest. The impact of changing the keep rate is not obvious but the performance is the worst when the learning rate is 1.



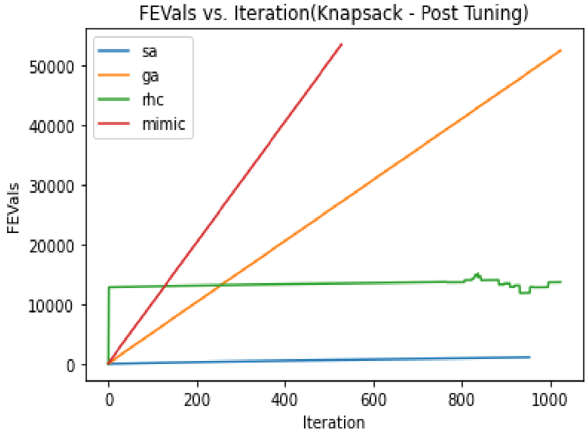


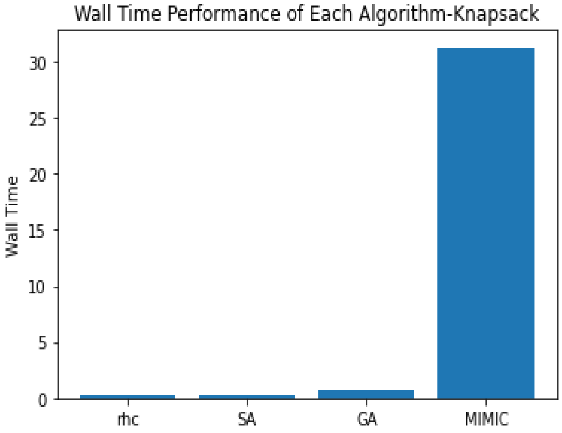
As problem sizes increase, GA and SA are outperforming RHC & SA. I was not able to highlight the performance of the MIMIC algorithm on the last problems because the structures of problems were not complicated and the ability to learn from previous experiences did not enhance the performance. But, MIMIC came in handy for Knapsack because of its ability to remember structures. It is still the first one to converge in less than 200 iterations which is very valuable when the cost functions are expensive to simulate. And the final fitness is significantly better than RHC & SA which performed poorly due to the complexity of problem.





Although the iterations required for MIMIC to converge was low. The high FEVals & complexity of evaluation at each are required for MIMIC which caused the time performance to tank.



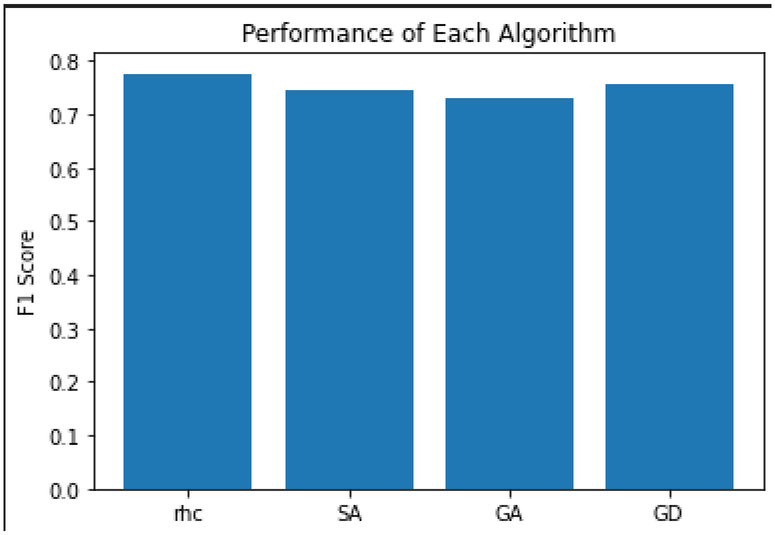


Part 2 ANN Weight Optimizations

We also investigated how the optimization algorithms can be applied to ANN to optimize the weights of the ANN algorithm Assignment 1 heart disease dataset. The heart disease dataset from UCI, includes health data that are readily available in every patient's file and whether or not the patient has underlying heart diseases is used for this evaluation. The dataset is simple and fairly balanced and includes 303 records, each with 13 features (e.g. age,sex,cholesterol level).

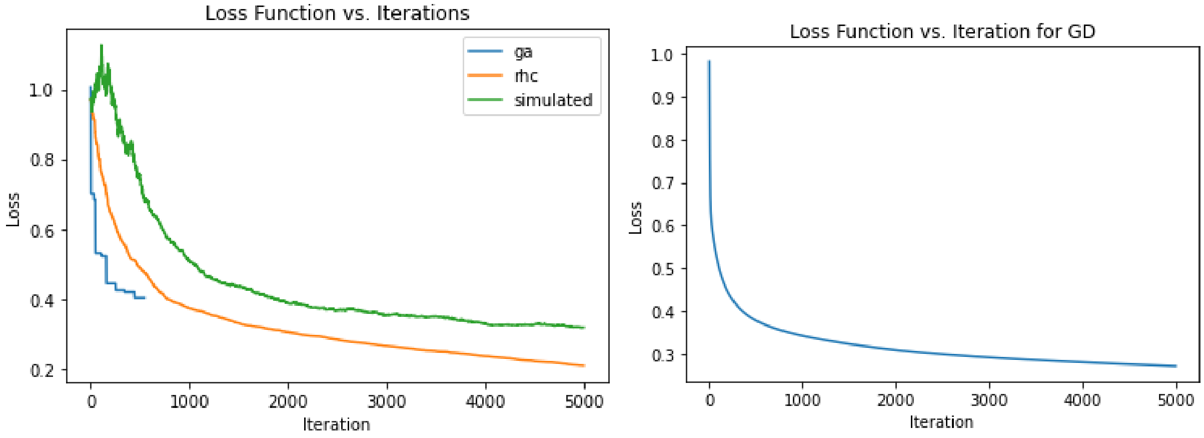
In order to compare the performance of these algorithms, the tuned values from assignment 1 were applied on Gradient Descent Algorithm (GD). GD is an optimization algorithm that minimizes errors between predicted and actual values. Three other algorithms (SA, RHC and GA) were also applied to the ANN Runner to evaluate the performance of each algorithm at finding optimal weight. The performance here is evaluated using F1 Score produced by each model. F1 Score takes both accuracy and recall into consideration which is able to give us a more holistic view of the model performance compared to looking at the accuracy rate only.

Bar chart showing F1 Score of each algorithm is displayed below:

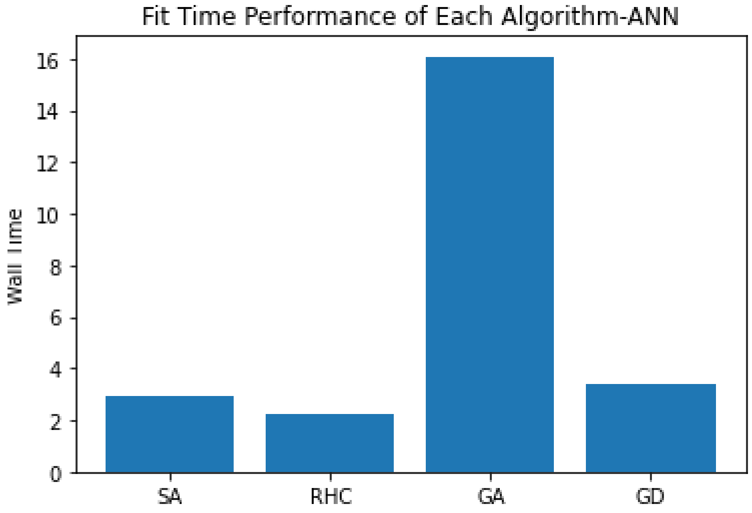


Due to the simplicity and linearity of the dataset, RHC was able to produce the highest F1 Score, followed by GD,SA and GA. Additionally tuning on default parameter values (pop_size , mutation rate) can improve the F1 Score further.

Using the loss curve provided by the Mlrose package, we also plotted the loss functions for all 4 algorithms below and saw that both RHC & SA were able plateaus after ~2000 iterations. However, the SA algorithm due to the random walk was seeing some loss fluctuations on the earlier iterations. The GA algorithm never flatten out on the loss curve which means it never converged. The GD algorithm was also able to quickly converge because of the linearity of the dataset and it was not likely to be trapped in the local optimals.



Lastly, the Wall Time for each algorithm was compared and showed that GA requires the longest time to compute because of its complexity in each iteration. Given the straightforward nature of the dataset, RHC remained as the winner in achieving high precision and accuracy with the best computation performance.


Fit Time Performance of Each Algorithm-ANN
Wall Time
0
2
4
6
8
10
12
14
16
SA
RHC
GA
GD