Problem Creation and Data Sets

Problem 1

Definition:

Data was sourced from Kaggle's Breast-cancer dataset. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe the characteristics of the cell nuclei present in the image. The optimization task or The Problem I have designed for this data is to find a model that maximizes the accuracy of predicting the Diagnosis. The advantages of Genetic Algorithm Optimization Problem is best highlighted in the classification problems where a given sample belongs to one of two classes. Since the data has this structure I think it will rely help me highlight its advantages.

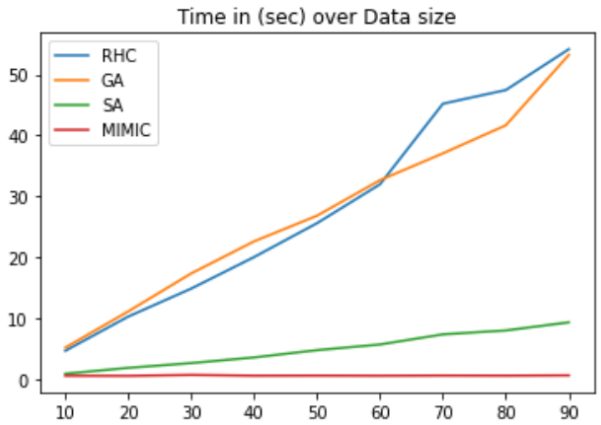
https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

Results:

The results for the algorithms were not entirely as expected. As we know genetic algorithm should support speedups and robustness to noise. Although it is was clearly achieved that the Genetic Algorithm most accurately finds the best state and feature for this optimization problem it id further found that it took one of the longest times to do so.

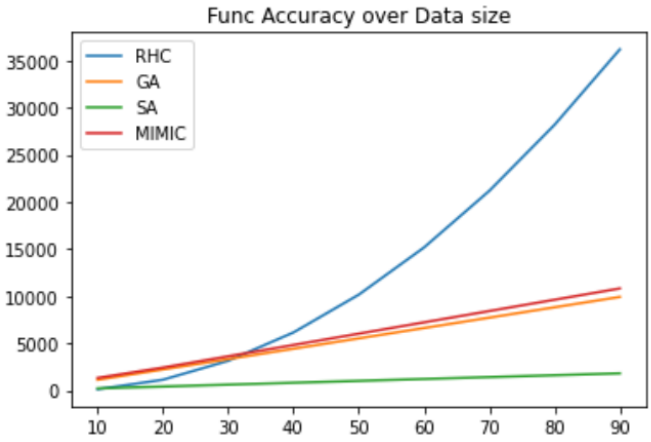
```
Best features
.............
best state RHC:
[1 1]
best feature RHC:
-0.8
best state GA:
[0 1]
best feature GA:
-0.8
best state SA:
[1 0]
best feature SA:
-0.8
best state MIMIC:
[0 0 1 1 0 1 1 1 0 0 0 1 0 1 1 0 1
1 1 0 1 1 1 0 1 0
 0 0 0 0 1 0 0 1 1 1 0 0 1 1 1 1 1
1 1 0 1 1 0 1 0 0
 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0]
best feature MIMIC:
8.0
```

As the results form the functions show MIMIC found the best feature to be 10 and best state to be 0. Which is aligns well with the data suggesting that the MIMIC function is accurate throughout the data size.



When looking at the time that each model took, It is clear that random hill climbing and Genetic Algorithm took the longest time. This makes sense because the random Hill climbing algorithm is a repetitive algorithm and so the search algorithm can have an

exponential time complexity depending on the data. Similarly, the Time complexity of the genetic Algorithm is also dependent on many factors including the size and shape of the data.



Lastly, when looking at Accuracies for these algorithms most of them have a continuous improvement as the data size increases, however, Randomized hill climbing seems to have an exponential improvement in accuracy.

Problem 2

Definition:

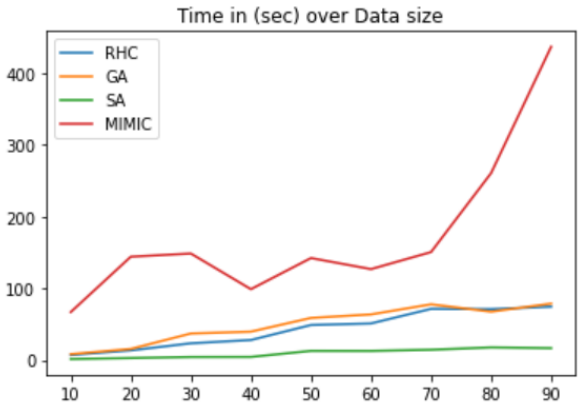
The second Classification task I picked was to predict Cover_Type. I used the UCI Cover type data set with over 50 features about the soil type From these I picked out Important features to design a classification model that best predicts clover type.

https://archive.ics.uci.edu/dataset/31/covertype

Results:

Below are the results for the second optimization problem. In this problem all of the models lead to the accurate best feature except MIMIC.

```
Best features
.............
best state RHC:
[1 0]
best feature RHC:
-0.5
best state GA:
[0 1]
best feature GA:
-0.5
best state SA:
[0 0]
best feature SA:
-0.5
best state MIMIC:
[1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0
1 0 0 0 0 1 0 0 1
 1 0 1 0 0 0 1 1 1 1 0 0 1 1 1 1 1
1 1 0 0 0 0 0 0 1
 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
best feature MIMIC:
14.0
```



For the Clove classification model, the data was very showing that staining Simulated Annealing look the shortest time. Genetic algorithm and randomized hill search fall somewhere in between white MIMIC took the longest. Which is quite accurate if the

data is looked at closely. These results show that staying an SVM function on this data will result in many local optima, and since the MIMIC function si prone to it, it changes its time complexity. Simulated Annealing has a good balance between exploration and exploitation, allowing it to quickly converge to near-optimal solutions.

Problem 3

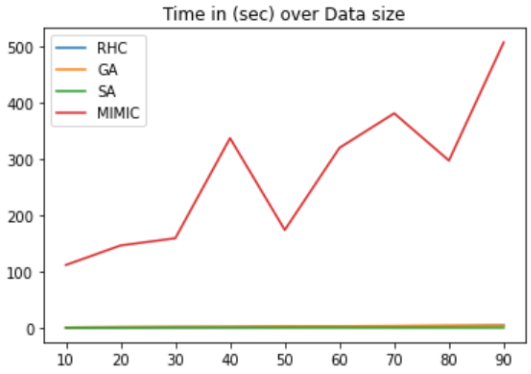
Definition:

Lastly, for the knapsack problem, I created my data by using Numpy's random functions. The data consists of data values and weights. the classification task is to find the optimal fit with a most value that is under a certain weight criteria.

Results:

```
Best features
.............
best state RHC:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1]
best feature RAC:
1388.0
best state GA:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1]
best feature GA:
1388.0
best state SA:
[1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0
 1 1 1 1 1 1 1 1 1 1 1 1 1]
best feature SA:
1336.0
best state MIMIC:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 0 0 0 1 1 0 0 0 0 1 0 1 0 1 1 1 1
 1 0 0 0 0 1 1 1 1 1 0 0 0 1 0 1]
best feature MIMIC:
16.0
```

Looking at the above results for each of the randomization functions on the Knapsack problem we can see that randomized hill search and Genetic Algorithm has the exact same results simulated Annealing's beast feature is a bit different and the MIMIC algorithm is drastically different. Here we can tell that MIMIC algorithm can find the correct optimization function. While the other algorithms got stuck on local maxima, the mimic algorithm was able to find the optimal solution most accurately.



A similar pattern was observed with time. Mimic took the to longest from all algorithms.

Conclusion:

From the above experiments and comparisons it is clear that MIMIC is the lowest of all algorithms. However, It is reliable in terms of accuracy in knapsack-type problems. When all models failed to find the best solution for the knapsack problem, MIMIC did. Similarly Simulated Annealing is mostly the fastest and best option for quickly finding the global optima's. Randomized hill climbing is also fast algorithm but not the best as it can get hard to converge and get stuck on local minima. Lastly Genetic Algorithm's speed and accuracy really just depends on the optimization problem, if the problem is not to complicated, it will quickly converge to the right solution but in complex problem this algorithm also struggles.

Resources:

https://github.com/gkhayes/mlrose/tree/master

https://archive.ics.uci.edu/dataset/31/covertype

https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

https://www.kaggle.com/code/nalkrolu/simulated-annealing-algorithm-for-tsp