Name: Sol Won

Machine Learning – Randomized Optimization

Description of the problems:

The goal of this assignment is to implement four local random search algorithms and apply to solve 3 fitness functions. Moreover, apply implemented algorithms to Neural Network and the outputs will be compared with the results that is calculated in Assignment 1.

Randomized hill climbing (RHC), simulated annealing (SA), genetic algorithm (GA), and MIMIC algorithms are implemented to solve Knapsack, ContinuousPeaks, and 4 Peaks fitness functions.

Description of algorithms:

Hill-climbing search checks neighboring nodes in current state and move if there are nodes with the higher valued state. If there is no more nodes to climb, algorithm claims that the state is peak. Randomized hill climbing (RHC) repeats varius hill-climbing algorithms and choose the optimal solution. Mlrose library is used to reproduce RHC algorithm.

Simulated Annealing (SA) algorithm is inspired by metal annealing. Hill-climbing method that is discussed previously has a weakpoint where it might get stucked to local maximum because when there is no more of higher valued state, algorithm will claim that the point is peak. But what if an algorithm state comes down from the peak and move to an another peak? It will have higher possibilities that it will reach the peak. First, algorithm chooses a state and look at the neighboring states. If the neighboring state is better, it moves but if not, it calculates whether to move or not with the temperature. Mlrose library is used to reproduce SA algorithm.

Genetic Algorithm (GA) is inspired by the nature's genetic phenomenon. In reality, there is a providence to keep the good gene. Also, there is a chance of mutations to intervene and it also belongs to one of gene algorithms. In this algorithm, state works just like gene and states are scored by fitness functions. Only the states with higher fitness score is able to breed. Crossover and mutations are also part of this algorithm which makes this algorithm better. Mlrose library is used to reproduce GA algorithm.

MIMIC is an algorithm looking for optima by estimating probability densities. This algorithm communicates the cost funtion that is calculated from one iteration to another. It keeps this process until optima is decided. Mlrose library is used to reproduce MIMIC algorithm.

Part 1 - 4 Peak problem:

This is a problem that has 2 local optimum and 2 global optimum at the edge. Fitness function evaluate the score by the following equation "*Fitness(x, T) = max(tail(0,x),head(1,x)) + R(x,T)".* First, all the local search algorithms (RHC, SA, GA, and MIMIC) is implemented with the default parameters. Then, hyperparameter tuning is done to output the better fitness score. Tuning parameter is different for all the algorithms.

First, RHC fitness score is calculated by inserting a default parameters. Max_attempts is set to 10 with iteration infinite. This gave the fitness score of 2.0 which is very low.

Name: Sol Won

First parameter to tune was the "Max_attempts" parameter. This parameter is the number of attempts to try and find better neighbor at each stage. Max attempt of 1, 10, 100, 1000, 10000 is used. As shown in Figure 1, max attempt of 1000 and 10000 had the highest fitness score. There was no difference between 1000 and 10000. So, max attempt of 1000 is used to reduce the time constraint. Figure 2 shows the score based on max iterations. Parameter of 1, 10, 100, 1000, 10000 is used and iteration of 10000 had the highest score. Figure 3 is a graph of fitness score based on the problem length. Parameter used were 20, 40, 60, 80, 100 and length of 100 had the highest fitness score. To keep the result consistent, random seed of 1 is used for the whole experiment. For references, number of iterations to reach fitness score of 100 for different seeds is shown in Figure 4. Seeds 1 through 10 is graphed. There was some gap betweeen the highest point (iterations) and lowest point but seed 1 seems to have average number of iterations.

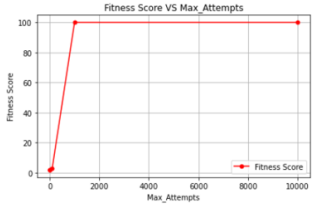

Figure 1: Fitness & Max_Attempts

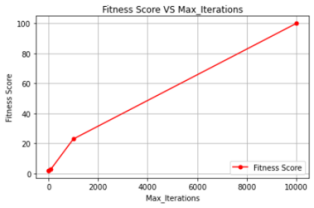

Figure 2: Fitness & Max_Iterations

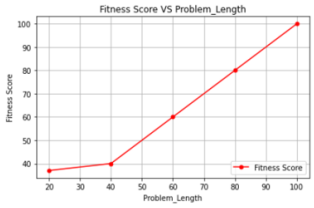

Figure 3: Fitness & Problem_Length

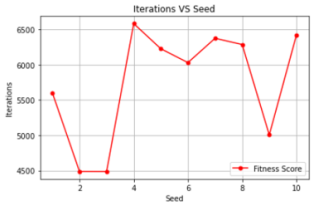

Figure 4: Iterations & Seed

Second, SA is also tuned by different parameters. First parameter that is tuned is the Max attempt. Figure 5 is a graph of fitness score with parameter 1, 10, 100, 1000, 10000. Max fitness score is hit from max attempt 10. Figure 6 is a graph with different max iteration parameters. Same parameter is used and the fitness score grows as the iteration grows. Figure 7 is the fitness score based on the problem lengths. It is quite interesting because the fitness score drops when problem length goes up from 80 to 100. Figure 8 is a graph of iteration that is taken to hit the max fitness score with different seeds. Highest and lowest interation was not that far from chosen seed which is 1. Moreover, table on the left shows the fitness score and iterations withdifferent schedules. GeomDecay and ExpDecay had better result compare to ArithDecay for this case.

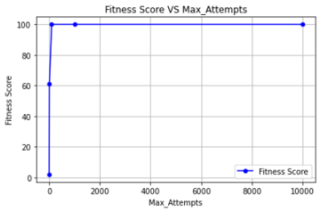

Figure 5: Fitness & Max_Attempts

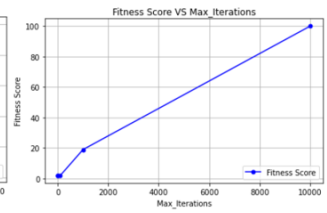

Figure 6: Fitness & Max_Iterations

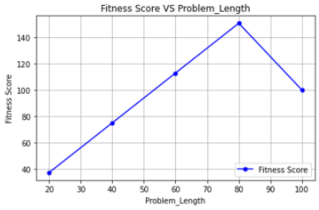

Figure 7: Fitness & Problem_Length

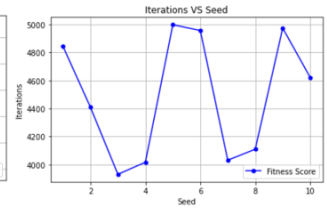

Figure 8: Iterations & Seed

Schedule	GeomDecay	ArithDecay	ExpDecay
Fitness Score	151	80	151
Iterations	4844	7627	4844

GA is tuned for this problem. First tuning was max atempt. Figure 9 shows that the fitness score with max atempt 10 already shows a good result but keep increasing as max atempt increases. Figure 10 shows that

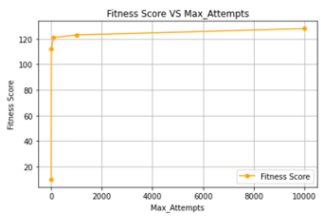

Figure 9: Fitness & Max_Attempts

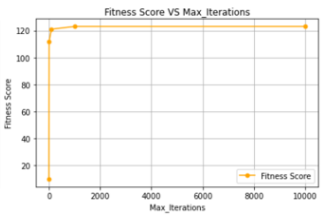

Figure 10: Fitness & Max_Iterations

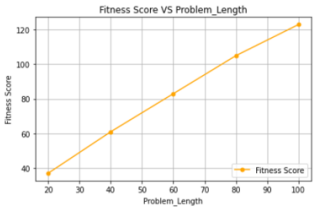

Figure 11: Fitness & Problem_Length

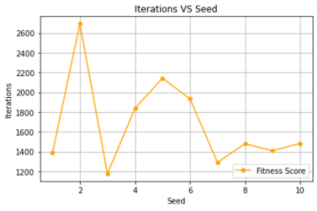

Figure 12: Iterations & Seed

1000 iteration in this case is good enough since there is no difference in fitness score between 1000 and 10000 iterations. Figure 11 shows that the fitness score increases as problem length increases. Problem length of 10 has the highest fitness score. Figure 12 is a graph of iteration that is taken to hit the max fitness score with different seeds. Unlike other algorithms (RHC and SA), the variation between the seed was high. Lastly, pop size was tuned. Pop size of 50, 100, 150, 200, and 250 was set as a parameter to see the difference. Pop size of 150 and 250 had the highest fitness score among 5 different pop sizes.

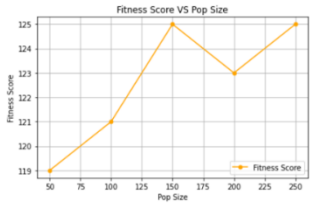

Figure 13: Fitness Score & Pop Size

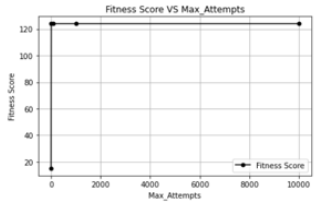

Figure 14: Fitness & Max_Attempts

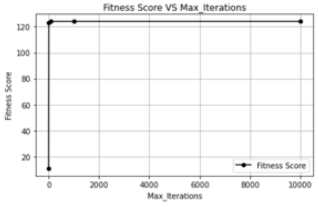

Figure 15: Fitness & Max_Iterations

MIMIC algorithm is also tuned. Figure 14 is a graph which shows the fitness score with different max attempt parameters. (1, 10, 100, 1000, and 10000) From max attempt 10 to 10000, the fitness score stayed the same. Figure 15 is a iteration graph. It also shows that the result is consistent from iteration 10 to 10000. Figure 16 shows the fitness score based on the problem length. There was a high gap between problem length 80 and 100. Figure 17 shows how many iterations were taken to reach the max fitness score. All the seeds had similar iterations between 3010 to 3045. Figure 18 shows the tuning of pop size. Pop size of 50, 100, 150, 200 and 250 is set as a parameter and pop size 200 (which is the default) had the best fitness score.

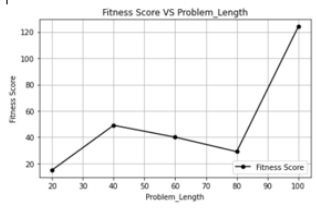

Figure 16: Fitness & Problem_Length

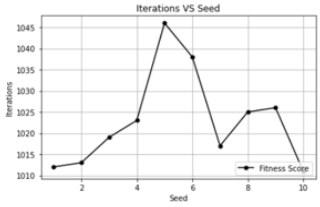

Figure 17: Iterations & Seed

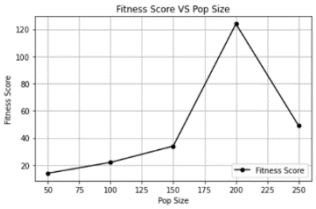

Figure 18: Fitness Score & Pop Size

From next problem, hyperparameter tuning section will be much shorter without explanation since same parameter was used to tune different algorithms.

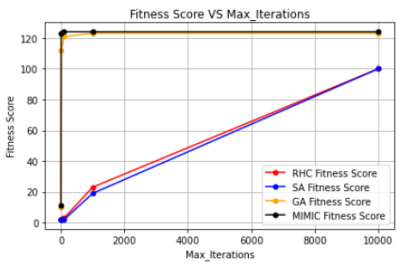

Figure 19: Fitness & Iterations

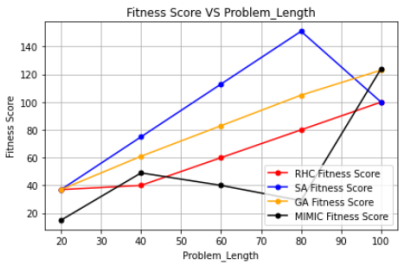

Figure 20: Fitness & Problem Length

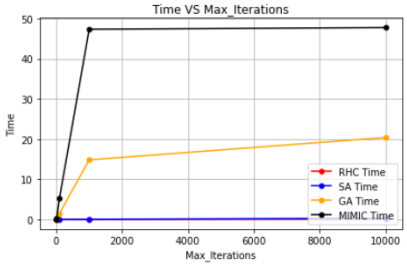

Figure 21: Time & Max_Iterations

Figure 19 is the fitness and iterations graph which shows all 4 algorithm results. When it reaches max fitness score, it continues for several hundred iterations and claims the peak. It shows GA and MIMIC algorithm has higher fitness score compare to RHC and SA. Also GA and MIMIC

used very little iterations to hit the high fitness score. Figure 20 is a graph of fitness score based on the length of the problem. At problem length 100, genetic algorithm and MIMIC had the highest fitness score which was 120. But by looking at the graph, GA was the most constant and there is high possibilities that the fitness score will go up as problem length goes up. Figure 21 shos the time it took to run the algorithm. MIMIC took longest as expected. RHC and SA was very fast in terms of runtime.

For 4-peak problem, I could conclude that GA is more likely to find global optima compare to other algorithms. MIMIC also had good performance but the time it took was so long and the difference of fitness score (compare to GA) was not enough to beat the timing constraint.

Part 1 - ContinuousPeaks problem:

This is a problem that is very similar to 4 peaks problem. Instead, there are many local optimums. Fitness function evaluate the score by the following equation "*Fitness(x, T) = max(max_run(0,x),max_run(1,x))+R(x,T)".* Parameter tuning is done to obtain the max fitness scores for each algorithms.

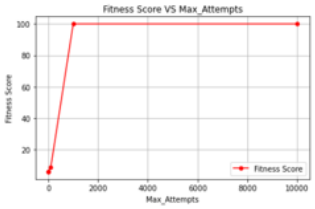

Figure 43: Fitness & Max_Attempts

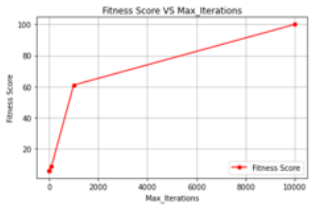

Figure 44: Fitness & Max_Iterations

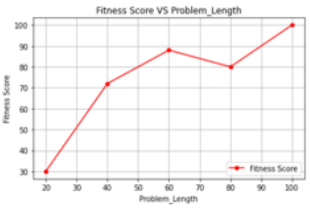

Figure 45: Fitness & Problem_Length

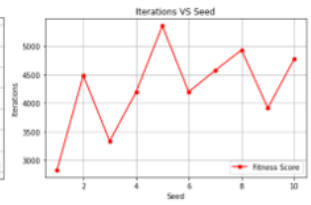

Figure 46: Iterations & Seed

For Continuous Peaks problem, only the result is elaborated since parameters are same with 4 peak problem. Figure 43 to 46 is a tuning result for RHC. By looking at Figure 43, max attempt is enough from 1000 attempts. Figure 44 shows that the fitness score keeps rising until10000 iteration. Figure 45 shows that the fitness score increases as problem length increases. Figure 46 shows that applying different seed doesn't really matter with the number of iterations take to achieve max fitness score.

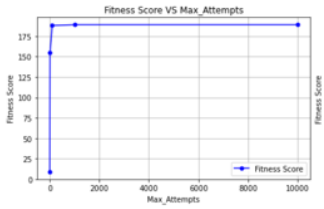

Figure 47: Fitness & Max_Attempts

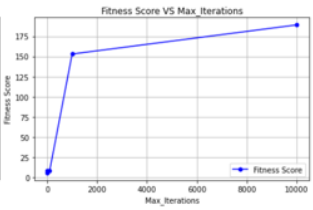

Figure 48: Fitness & Max_Iterations

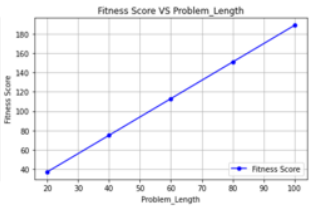

Figure 49: Fitness & Problem_Length

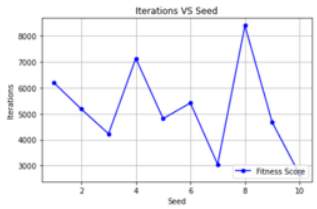

Figure 50: Iterations & Seed

Figure 47 to 50 is a tuning result for SA. By looking at Figure 47, max attempt is enough from 100 attempts. Figure 48 shows that fitness score increases as number of iteration increases. Score continuously grow until iteration 10000. Figure 49 shows that the fitness score increases as problem length increases. Figure 50 shows that applying different seed doesn't really matter with the number of iterations take to achieve max fitness score.

Schedule	GeomDecay	ArithDecay	ExpDecay
Fitness Score	189	189	189
Iterations	6193	6421	6922

Schedule is also tuned. For this problem, fitness score of all 3 schedule had less gap compare to other problems. But still,

GeomDecay had the lowest iterations to reach the max fitness score.

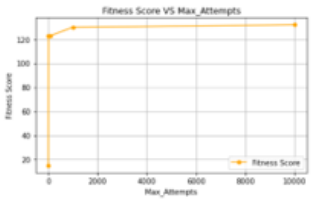

Figure 51: Fitness & Max_Attempts

Figure 52: Fitness & Max_Iterations

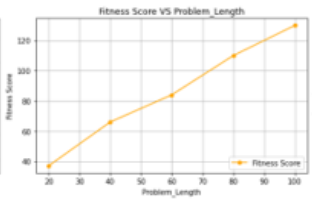

Figure 53: Fitness & Problem_Length

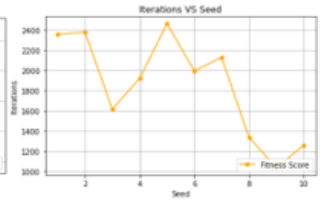

Figure 54: Iterations & Seed

Figure 51 to 55 is a tuning result for GA. By looking at Figure 51, max attempt constantly grows until 10000 attempts which means 10000 is the tuned parameter in this case. Figure 52 shows that 10000 iteration is enough to achieve max fitness score. Figure 53 shows that the fitness score increases as problem length increases. Figure 54 shows a huge gap of iterations between thee seeds. Figure 55 shows that pop size of 250 has the best fitness score compare to other pop sizes.

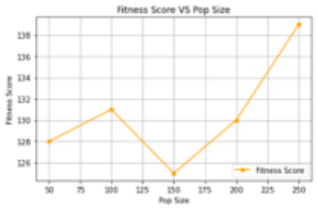

Figure 55: Fitness Score & Pop Size

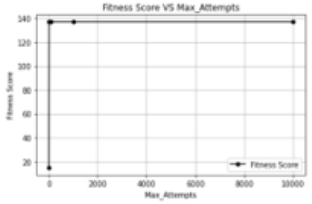

Figure 56: Fitness & Max_Attempts

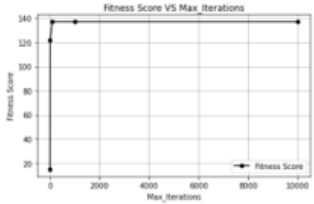

Figure 57: Fitness & Max_Iterations

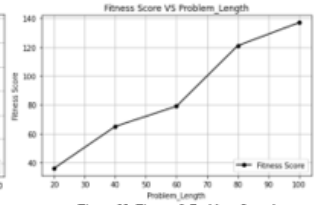

Figure 58: Fitness & Problem_Length

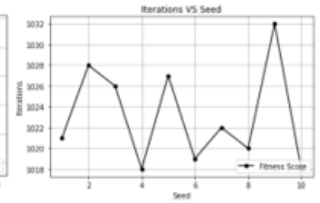

Figure 59: Iterations & Seed

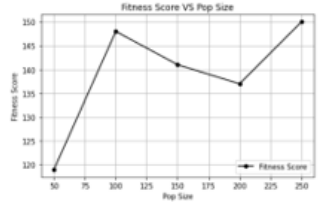

Figure 60: Fitness Score & Pop Size

Figure 56 to 60 is a tuning result for MIMIC. By looking at Figure 56, max attempt is achieved since 10 attempt. Figure 57 shows that 100 iteration is enough to achieve max fitness score. Figure 58 shows that the fitness score increases as problem length increases. Figure 59 shows that applying different seed doesn't really matter with the number of iterations take to achieve max fitness score. Figure 60 shows that pop size of 250 has the best fitness score compare to other pop sizes. Now comparison of the local search algorithms will be presented.

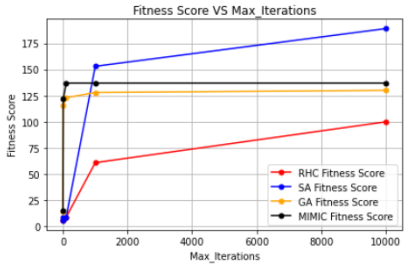

Figure 61: Fitness & Iterations

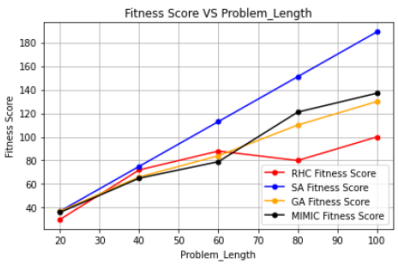

Figure 62: Fitness & Problem Length

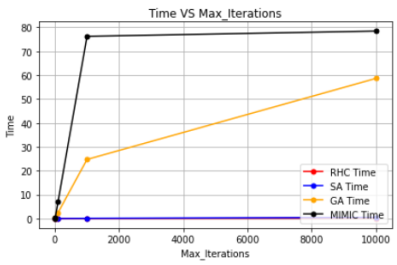

Figure 63: Time & Max_Iterations

Figure 61 is the fitness and iterations graph which shows all 4 algorithm results. It shows that SA algorithm is on top of all the other algorithms and still have potential to grow if there are more iterations. RHC showed the similar graph with upslope but the score was very low compare to SA. Figure 62 shows the fitness with different problem lengths. MIMIC, RHC, and GA seem to

have similar scores ith up slope lines. SA fitness score continuously grew as problem length increased. The increase is pretty much of a constant rate. Figure 63 shows the time constraint. MIMIC and GA took longer time compare to RHC and SA. Also, since the fitness score is also lower than SA, this problem is advantage of SA.

Part 1 - Knapsack problem:

Knapsack problem is to fill the sack within the list of weights. Total weights should be less or equal to the sack limit. While stacking up the sack, the combination of value has to be as large as possible. Local search algorithms is used to see which algorithm will give the best fitness. Parameter tuning is done to obtain the max fitness scores for each algorithms.

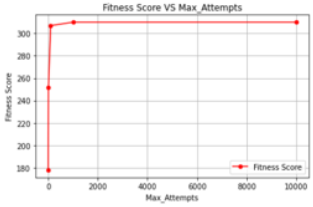

Figure 22: Fitness & Max_Attempts

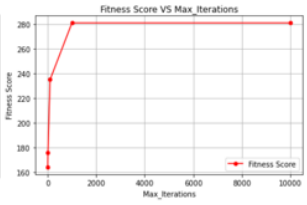

Figure 23: Fitness & Max_Iterations

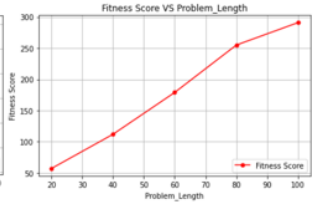

Figure 24: Fitness & Problem_Length

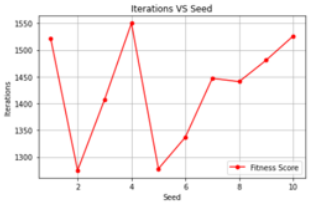

Figure 25: Iterations & Seed

For Knapsack problem, only the result is elaborated since parameters are same with 4 peak problem. Figure 22 to 25 is a tuning result for RHC. By looking at Figure 22, max attempt is enough from 1000 attempts. Figure 23 shows that 1000 iteration is enough to achieve max fitness score. Figure 24 shows that the fitness score increases as problem length increases. Figure 25 shows that applying different seed doesn't really matter with the number of iterations take to achieve max fitness score.

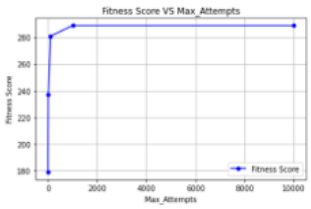

Figure 26: Fitness & Max_Attempts

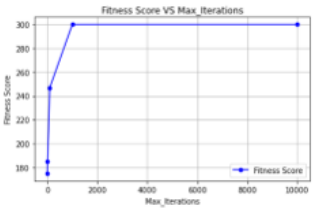

Figure 27: Fitness & Max_Iterations

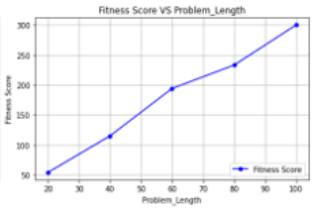

Figure 28: Fitness & Problem_Length

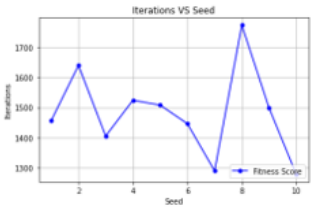

Figure 29: Iterations & Seed

Figure 26 to 29 is a tuning result for SA. By looking at Figure 26, max attempt is enough from 1000 attempts. Figure 27 shows that 1000 iteration is enough to achieve max fitness score. Figure 28 shows that the fitness score increases as problem length increases. Figure 25 shows that applying different seed doesn't really matter with the number of iterations take to achieve max fitness score.

Schedule	GeomDecay	ArithDecay	ExpDecay
Fitness Score	318	318	318
Iterations	1457	8979	1460

Schedule is also tuned. GeomDecay and ExpDecay had similar result where it reached fitness score of 318 with less than 2000 iterations. GeomDecay has 3 iterations less compare to ExpDecay schedule.

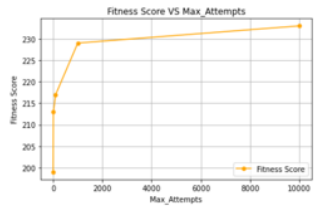

Figure 30: Fitness & Max_Attempts

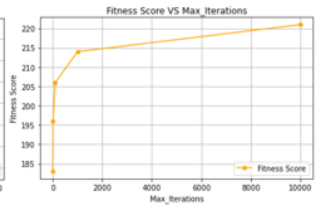

Figure 31: Fitness & Max_Iterations

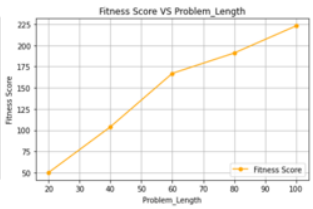

Figure 32: Fitness & Problem_Length

Figure 33: Iterations & Seed

Figure 30 to 34 is a tuning result for GA. By looking at Figure 30, max attempt constantly grows until 10000 attempts which means 10000 is the tuned parameter in this case. Figure 31 shows that 10000 iteration is enough to achieve max fitness score. Figure 28 shows that the fitness score increases as problem length increases. Figure 25 shows a huge gap of iterations between thee seeds. There are quite a lot of seeds with higher iterations to get the max fitness scores. Figure 34 shows that pop size of 250 has the best fitness score compare to other pop sizes.

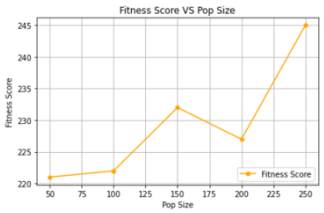

Figure 34: Fitness Score & Pop Size

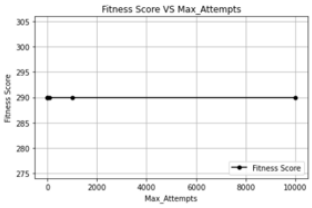

Figure 35: Fitness & Max_Attempts

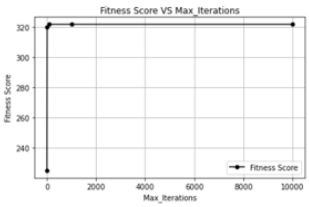

Figure 36: Fitness & Max_Iterations

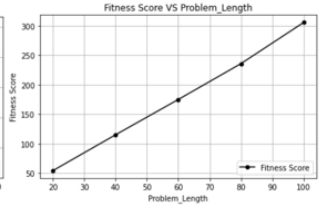

Figure 37: Fitness & Problem_Length

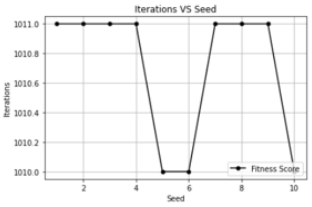

Figure 38: Iterations & Seed

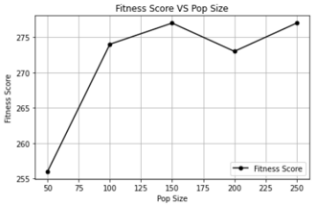

Figure 39: Fitness Score & Pop Size

Figure 35 to 39 is a tuning result for MIMIC. By looking at Figure 35, max attempt is achieved since 1 attempt. Figure 36 shows that 100 iteration is enough to achieve max fitness score. Figure 37 shows that the fitness score increases as problem length increases. Figure 38 shows that applying different seed doesn't really matter with the number of iterations take to achieve max fitness score. Figure 39 shows that pop size of 150 and 250 has the best fitness score compare to other pop sizes. Now comparison of the local search algorithms will be presented.

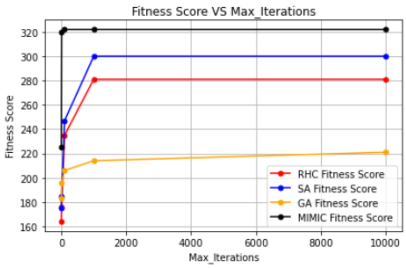

Figure 40: Fitness & Iterations

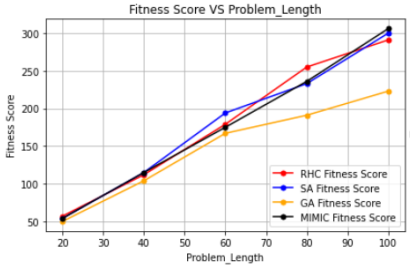

Figure 41: Fitness & Problem Length

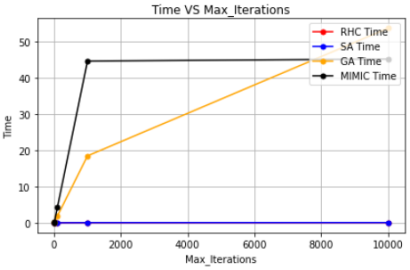

Figure 42: Time & Max_Iterations

Figure 40 is the fitness and iterations graph which shows all 4 algorithm results. It shows that MIMIC is on top of all the other algorithms. All other algorithms at least needed 1000 iterations to reach the max fitness score but MIMIC reached the highest fitness scores with just few iterations. Figure 41 shows the fitness with different problem lengths. MIMIC, RHC, and SA seem to have similar scores and up slope lines. GA fitness score increased slowly as problem length increased. Figure 42 shows the time constraint. MIMIC took longer time compare to RHC and SA but still, MIMIC had advantages. MIMIC was strong for this problem since it communicated a underlying structure of a problem with the iterations that happened before.

All the local search algorithms is compared with their best parameters. GA had advantages with the 4 peaks problem, SA on coutinous peak problem, and MIMIC on knapsack. It is possible to optimize better if more number of parameters exist. But the result after tuning was enough to differentiate and tell which local search algorithms excelled while solving varius problems.

Part 2 – Local Search Algorithms with Neural Network

Breast Cancer Wisconsin (Diagnostic) Dataset is used for this analysis. There are total of 30 attributes of cell nucleus and they are computed from a digitizd image of a fine needle aspirate of a breast mass. Classification problem is to predict whether the tissue is benign or malignant by looking at those 30 attributes. There are 357 benign cases and 212 malignant cases.

	Predict [0]	Predict [1]
True [0]	137	6
True [1]	10	75

Benchmark is brought from assignment 1 neural network part. Following is the confusion matrix and classification report after hyperparameter tuning. There are 6 false negative and 10 false positives. This output is acceptable since there are just 16 items that is wrong out of 228 records. Table on the left is a classification table. Accuracy is 0.93 with the test dataset which is acceptable.

	precision	recall	F1-score	Support
0	0.93	0.96	0.94	143
1	0.93	0.88	0.90	85
Accuracy			0.93	228
Macro avg	0.93	0.92	0.92	228
Weighted avg	0.93	0.93	0.93	228

Now, local search algorithms will be used to find good weights for a neural network. First, each search algoritms are used with default parameters. Then, parameter tuning will happen and accuracy will be compared.

First, random hill climbing is applied with default parameter. Following table is the confusion matrix after the training and predicting. There are 14 false negatives and 13 false positives. Prediction is considered as good but compare to our benchmark, it's not.

	Predict [0]	Predict [1]
True [0]	129	14
True [1]	13	72

Table on the left is a classification report for RHC neural network. Accuracy score is 0.88 with F1-score of 0.91 and 0.84. Compare to the result of our benchmark, accuracy got lower by 0.5. The difference is understandable since there are random factors in local search algorithms.

	precision	recall	F1-score	Support
0	0.91	0.90	0.91	143
1	0.84	0.85	0.84	85
Accuracy			0.88	228
Macro avg	0.87	0.87	0.87	228
Weighted avg	0.88	0.88	0.88	228

However, my thought was that the parameter tuning might help accuracy scores to get higher. First, max iteration was tuned. Parameters used are 10, 100, 1000, and 10000. Blue line is the training accuracy and the red line is test accuracy. Figure 64 shows that there is improvement as number of iterations increase. After 10000 iterations, accuracy score increased from 0.88 to 0.91. Figure 65 shows that there was no improvement in accuracy score even max attempts is adjusted. After the parameter tuning, there are 13 false negatives and 8 false positives. F1-score is impoved a lot as well. F1-score changed to 0.93 and 0.88. In conclusion, parameter tuning made the accuracy score to go up (In this case, it is really close to the benchmark) but still, it is less compare to the benchmark accuracy and f1 scores.

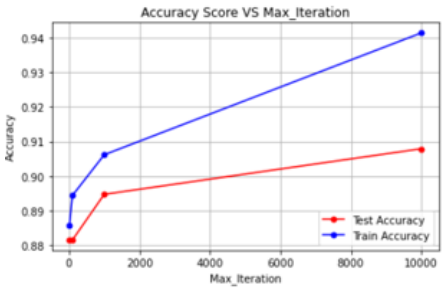

Figure 64: Accuracy vs Iteration

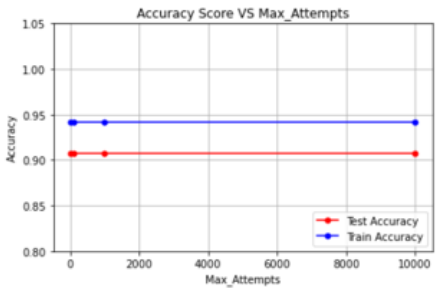

Figure 65: Accuracy vs Attempts

Second local search algorithm is simulated annealing. Following table is the confusion matrix.

	Predict [0]	Predict [1]
True [0]	133	10
True [1]	20	65

There are 10 false negatives and 20 false positives. Prediction is not in perfect state but good enough to be considered. Compare to RHC, SA has less false negatives and higher false positives. Table on the right side shows the accuracy score of 0.87 and f1 score of 0.90 and 0.81. Compared to the benchmark, accuracy is less by 0.6 and this also applies to the F-1 scores. Figure 66 shows the accuracy of train and test. Both lines goes up as iteration goes up. It first starts up with little bit of underfitting but as number of iteration goes up, it seems to have overfitting issues. After the tuning, accuracy became nearly 0.91. It is lower compare to our benchmark but there is an improvement. Figure 67 shows that accuracy doesn't really matter with the max attempt parameter. In SA, different schedules are used as well. GeomDecay had accuracy of 0.91, ArithDecay with 0.90, and ExpDecay ith 0.91. After the tuning, accuracy score and F1 score increased. It was close to our benchmark but it was still lower compare to the benchmark values.

	precision	recall	F1-score	Support
0	0.87	0.93	0.90	143
1	0.87	0.76	0.81	85
Accuracy			0.87	228
Macro avg	0.87	0.85	0.86	228
Weighted avg	0.87	0.87	0.87	228

Figure 66: Accuracy vs Iteration

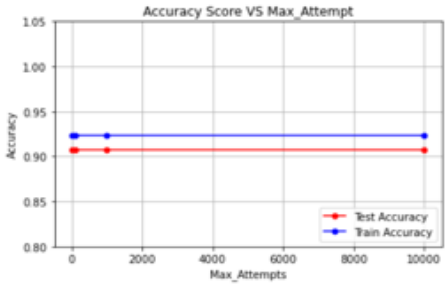

Figure 67: Accuracy vs Attempts

Last local search algorithm is the Genetic algorithm. Following is the confusion matrix. There are 4 false negatives and 24 false positives. Did very good job on predicting the false negatives but not really good in predicting false positives. Table on the left side is a classification report. With the default parameter, accuracy score is 0.88 with F1-score of 0.91 and 0.81. Parameter tuning is done with 4 parameters. (max iterations, max attempt, pop size and mutation probability) Below is a graphs of tuning.

	Predict [0]	Predict [1]
True [0]	139	4
True [1]	24	61

	precision	recall	F1-score	Support
0	0.85	0.97	0.91	143
1	0.94	0.72	0.81	85
Accuracy			0.88	228
Macro avg	0.90	0.84	0.86	228
Weighted avg	0.88	0.88	0.87	228

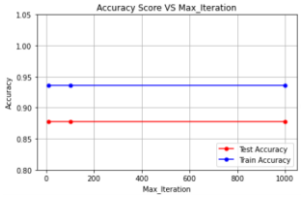

Figure 68: Accuracy vs Iteration

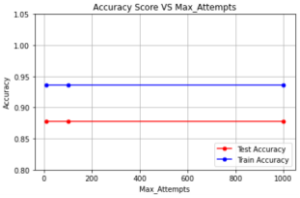

Figure 69: Accuracy vs Attempts

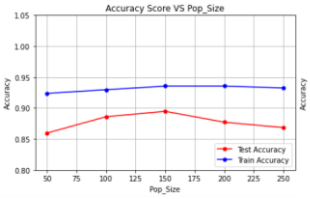

Figure 70: Accuracy vs Iteration

Figure 71: Accuracy vs Attempts

There are few interesting points to look at. For figure 68 and 69, it is just a straight line where accuracy score is steady. Which means, max iterations and max attepmts parameters don't play a big role here. Figure 70 shows the accuracy based on the pop size. The graph is quite interesting

since the accuracy line goes up and down. Figure 71 shows the different accuracies based on mutation probability. Parameter was 0, 0.1, and 1. As a result, mutation probability of 1 gave the best accuracy which is 0.92.

In conclusion, there are 3 local search algorithms used. Randomized hill climbing, simmulated annealing and genetic algorithms are used to compare the accuracy and other results with our benchmark. Since all 3 local search algorithms include some sort of randomized factor, even the values are really close, it could not beat the benchmark values.

References:

Dataset Citations:

Wolberg, William H. “Breast Cancer Wisconsin (Diagnostic) Data Set.” *UCI Machine Learning Repository*, 1995, archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29.

Library Citations (for the code):

Heyes, Genevieve. “Fitness Functions¶.” *Fitness Functions - Mlrose 1.3.0 Documentation*, 2019, mlrose.readthedocs.io/en/stable/source/fitness.html.