Assignment 2:
Randomized Optimization

Vipul Koti
vkoti7@gatech.edu

1 INTRODUCTION

This report is presented in two sections, The Part 1 focuses on the three problem optimization domains, where we try to maximize the Fitness function. We choose these problems to highlight the superiority of one Random Optimization Algorithm over others. We are comparing four random optimization algorithms, two Stochastic Optimization algorithms (Random Hill Climb & Simulated Annealing) and two population algorithms (Genetic algorithm & MIMIC). The Part 2 covers the problem, to find good weights for the Neural Network we created in Assignment 1 (Dataset: Heart Failure Prediction Dataset). We compared the performance of the three Random optimization algorithms (Random Hill Climb, Simulated Annealing & Genetic algorithm) with the backprop (Gradient Descent) we used in Assignment 1. To do these experiments, we used mlrose hiive[1] python library.

2 PART 1: RANDOM OPTIMIZATION ANALYSIS ON 3 PROBLEMS

2.0.1 *Common Methodology Followed & Algorithms Hyperparameters analyzed*

For all the three problems and the four Algorithms. We first run the runners (Downloaded mlrose hiive code to understand inputs and outputs for Runners) with different random seed and choose best, These Runners give us the Data frame with the metric's (Time, FEvals, Fitness) for each combination of the hyperparameters list's we provide as input to the runner. We used this Metric Data Frame to analyze function evaluations and to plot hyperparameter tuning Fitness plots. Also, we choose the best hyperparameter's combination from the Data Frame for the algorithm based on the fitness score, which we use later to study the fitness score, function evaluation and time taken by the best tuned algorithm's on different sizes of the problem.

Paramaters	Values
max_iters	500(SA &RHC) and 200(GA & MIMIC)
max_attempts	100(For All)
random_state	33(For All)
restarts	1 to 100(For RHC)
pop_size	50,100,200(GA & MIMIC)
keep_pct	0.1, 0.4, 0.8(MIMIC)
temperature	0.05, 0.1, 0.5, 1, 10, 20, 25(SA)
mutation_prob	0.1, 0.25, 0.5 (GA)

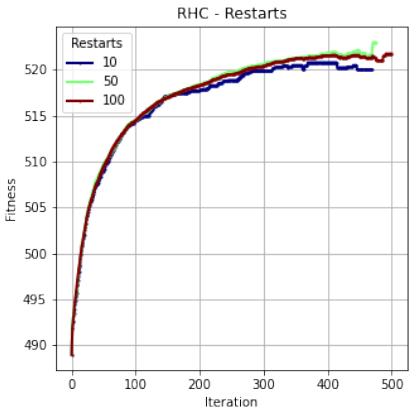
As listed in the table, these are the parameters we analyzed for part 1. We choose maximum iteration as 500 for SA RHC in comparison to 200 for GA Mimic, as SA RHC are fast to converge and do comparatively less function evaluation per iteration. So to have a fair comparison, we used those values. Also, **we use max_attempts as 100, as we noticed the algorithms stuck in local optima for default value 10.**

Note - Though, We ran the algorithms for the larger parameters. But we plotted only for the parameters listed in the table. So we will cover larger parameter's result in our analysis.

2.0.2 *Problem 1: N Queens Problem*

N Queen problem is an optimization problem classified as NP-Complete Problems. It is the problem where we place N queens on an N×N chess board and try to find a place for the queens, where no queen attack each other. The position of each queen is denoted by a state vector.For Fitness Function, As suggested in mlrose document[2], we use custom fitness function to make it maximization problem. Our Goal is to find the state vector for which there are no pairs of attacking queens. **As non attacks on each queen in a subspace can be optimized independently, We choose this problem to highlight the advantages of our Selected Genetic Algorithm.** For the Runners, we choose no. of queens as 33, which means 561 possible attacks in total and for problem size plots we vary the no. of queens between 10 and 60, which is 55 to 1711 possible attacks.

Random Hill Climb(RHC)-

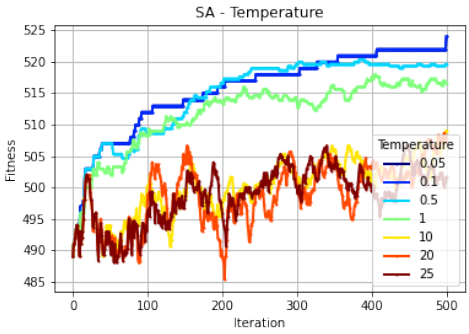


RHC, the stochastic hill climbing algorithm, converges to fitness score 517+ for 16+ current_restart value for the problem of 33 queens. It takes, minimum, 5,800+ Function evaluations to converge. **The RHC algorithm basically loop through random state vectors (Queen Positions) to find the better state vector with the better fitness. It gets stuck at a local optima of 524 fitness score.** From the plot and DF Metrics, Fitness score for restarts (10, 50 100) are converging around 520 plus score. Mean Function evaluations for restart 10 is around 2,317, 50 is 9,777 and 100 is 18,866.

The best fitness score 524 is for restart= 50 and FEvals=6,186 and iteration= 272, As we use max_attempts=100 and the fitness score converges after 272 iteration (max iteration=500). RHC uses very little memory, as it doesn't store the previous state's tree.

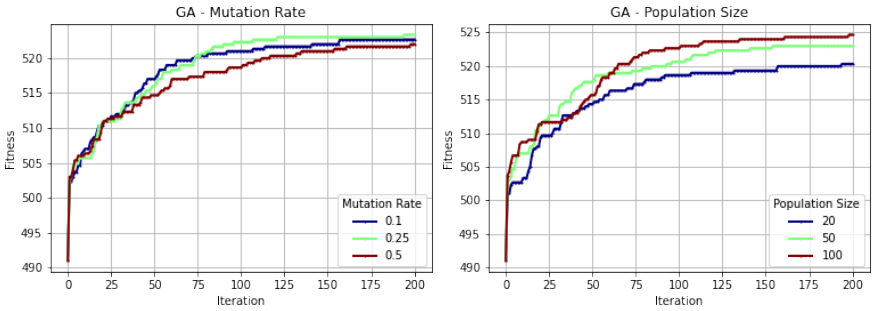
Simulated Annealing(SA)-

SA, the stochastic hill climbing algorithm with nondeterministic search for the global optimum as we accept decreased fitness state with the probability function of current state, future state and temperature in order to explore and escape the local optimum. Basically, **if the temperature is high we explore more and if the temperature is low we exploit, and it works as hill climbing algorithm.** For our problem of 33 queens with our selected SA algrithm, we tried multiple temperature algorithms with RHCrunner, As we are constrained with 500 iterations. Only the algorithms with temperature < 0.1 are near convergence.



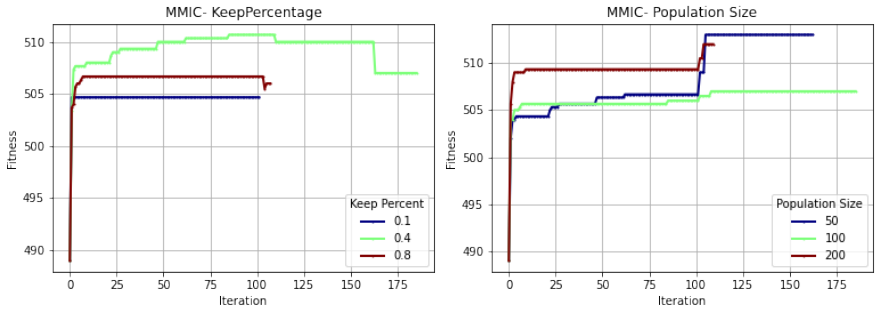
From the plot we can see, the high temperature (10,20,25) is still in explore mode until iteration 500. SA doesn't do as many functional evaluations as other algorithms does. The best fitness score 524 is for temp= 0.05 and FEvals=588 (around one function evaluation per iteration). For temp 25 the maximum evaluation for 500 iteration is 977. Also the mean fitness score of best temp 0.05 is 514 which is slightly less than RHC 520.

Genetic Algorithm(GA)-



GA is metaheuristic population optimization method inspired by natural selection (DNA mutation and crossover of two parents chromosomes to produce new offspring). In GA, **we don't need to compute derivative (slope) as opposed to RHC and SA where we climb the slope.** Mlrose uses standard genetic algorithm, which is one-point crossover. The GA is somewhat like parallel random restart, as we mutate in the population size and generate different states as new population. From the plots, we see, increasing the population and increasing the keep percent has positive impact on the fitness score. For the problem of 33 queens, our best selected Ga algorithm with population size 50 and mutation rate 0.25 produces the best fitness score of 525 with function evaluation of 6236, and it converges at iteration 121. Also, **the mean fitness score, across different population size and mutation rate is greater than 520, which make GA a better choice than RHC and SA.**

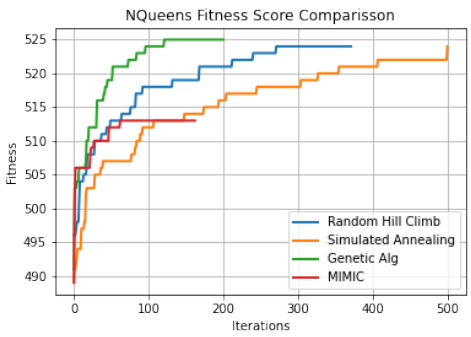
MIMIC-



MIMIC algorithm models the probability distribution as SA, it refines the model continuously which conveys the structure of the search space. **It iteratively shrinks the sample space to choose the global optima.** It estimates new distributions by sampling from the dependency trees, which we find by minimizing KL divergence between it and the true distribution. For the problem space, the MIMIC algorithm we looked at with population size < 200 is less as we see it performs the worst and converge around 505 to 510 for different population size and keep percent size. Also As we know the problem with N queen has a lot of local minima above 500 score which is why MIMIC is getting stuck for this complex problem which doesn't have a structure, which generally Mimic is good at.

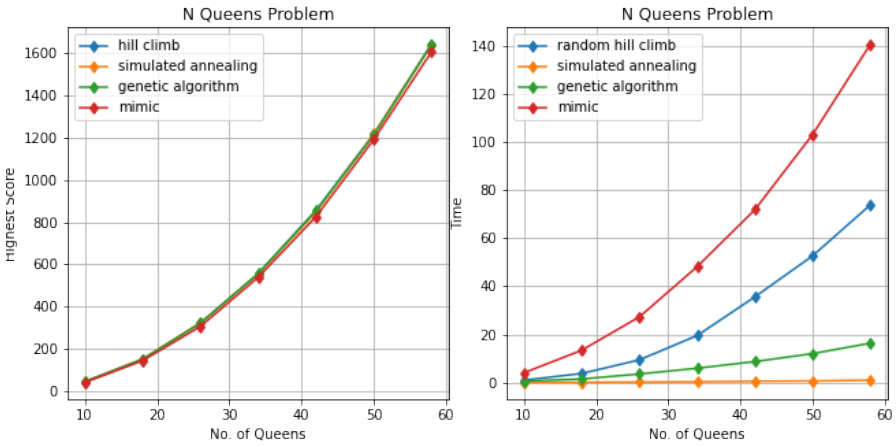
N Queens, Comparison of Best Algorithms-

If we see the function evaluation of the best algorithms for 4 different Random optimization techniques, RHC does around 6000 FEvals, SA does around 1,000, Mimic 8,000+ and GA as 10,000+ FEvals. Still GA takes less time than RHC and MIMIC (This is covered with plots later). MIMIC FEvals is less than GA because MIMIC converges faster with small population size. As we are choosing the algorithms based on the fitness score, Function Evaluation and Time taken by algorithms is not considered evaluating the best algorithm.
As we see from the Fitness curve plotted over iterations (200 for MIMIC and GA, whereas 500 for SA and RHC).



We can clearly see selected best Genetic Algorithm performs the best and selected best MIMIC performs the worst. (Note, if we increase the population size and don't use max attempts the results are different for same/different iteration count, MIMIC performs better than all three algorithms). **Selected GA performs better, because GA exploits the already explored the best state, which suits better to the N queen problem, whereby crossing over best states results in better states over iterations.**

From the two curves on the right, we plotted the top score to number of queen size in the problem, and we plotted Time taken with each algorithm for the problem size. For these plots, we used the best plots after tuning. Here also we can see the top score is very close for all the algorithms, but GA is slightly better. Interesting is even though the FEvals of GA is highest, it still takes less time than RHC and Mimic.



This is because GA doesn't calculate Derivative like RHC and complex distribution calculations like MIMIC.

2.0.3 *Problem 2: Knapsack Problem*

Knapsack is also an optimization problem which is classified as NP-Hard problems. It is the problem where we maximize the value of items that can fit in a knapsack with a constraint of not increasing the maximum weight of the sack. **Even though it suits GA with independent subspace argument as N queen problem, but when we constrain on the iterations, MIMIC should perform better.**

Fitness function of Knapsack problem(MLRose Implementation[3]) is -

Fitness(x) = Summation(i=0 to i=n-1) ViXi , if Summation(i=0 to i=n-1) Wi * Xi <= WSACK, 0 otherwise

where **W** is weight, **X** is value and **WSACK** is max sack weight. For our Problem we used

knapsack_length = 300 & max_weight_pct = 0.5

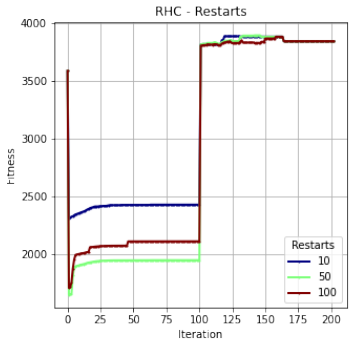
knapsack_weights = np.random.uniform(10,40,knapsack_length) 300 weights between 10 and 40

knapsack_values = np.random.uniform(20,30,knapsack_length) 300 values between 20 and 30

For this problem, we changed the population size for MIMIC and GA to choose from [200,400,600] from [50,100,200]. Also, for fair comparison, we reduced the maximum number of iteration for MIMIC & GA to 100 from 200 and for SA & RHC to 300 from 500.

Random Hill Climb(RHC)-

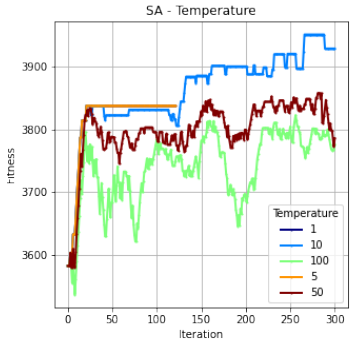
We ran the RHC runner with the three different restart [10, 50 and 200] as in the below figure.



For this problem, we see a large variance in the results for RHC. The max and mean varies by a lot. Where the max fitness score is 4037.685. The overall mean score is only 2078. We tried changing random seed to analyze, every time the selected RHC algorithm get stuck in local optima around 3000 and 4000. **Selected RHC algorithm is not robust for this size of the problem.**

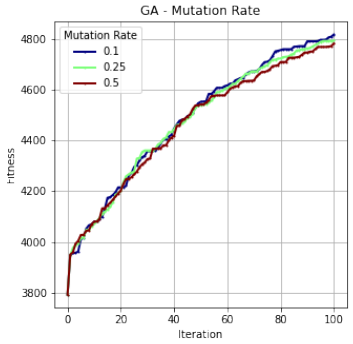
Simulated Annealing(SA)-

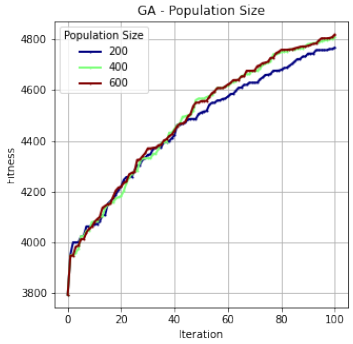
SA also performs similar, it's also a locally searched algorithm. SA does random tries to pick maximum value items for the knapsack, **But unlike RHC, SA picks the lower fitness state with the probability function of current state, next state and temperature. This is the reason we see a lot of steps in SA in comparison to RHC.** 300 iterations are less for SA for this problem. So We used 3000 iteration, temp values under 10 converged to maximum of fitness 4200, greater temp needed more iterations.



Genetic Algorithm(GA)-

The population algorithms, generally, work better on this problem. Because each successive generation is utilizing the best from the previous.





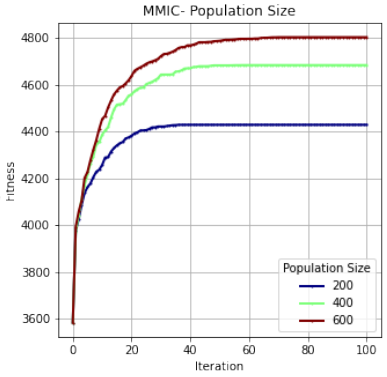
when we see the plots, **GA for knapsack problem, basically we will keep improving the set of the weights by mutating and crossing over the existing good weight independent subspace states.** From the plot, too, we can see the improvement over iteration. As the problem length is small and we run for constrained iterations, population size has less impact.

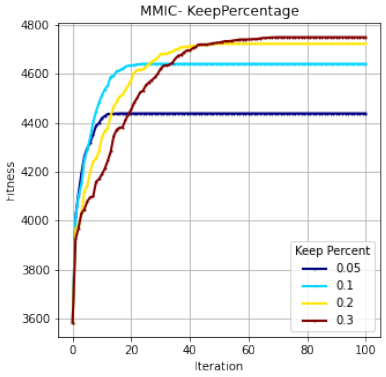
MIMIC-

Mimic is also the population algorithm which interpolates or extrapolates from previous states. This is the structural problem with constrained iterations, where meeting our assumptions, MIMIC outperforms other algorithms. Mimic models the probability distribution and successively refine it by choosing from the Dependency trees which are created using KL divergence to minimize the distance with the true distribution. By choosing values f(x) > Theta, **it successively shrinks the sample space, which make it easier to choose global optima, that's why it converges faster as seen in all the comparison plots.**

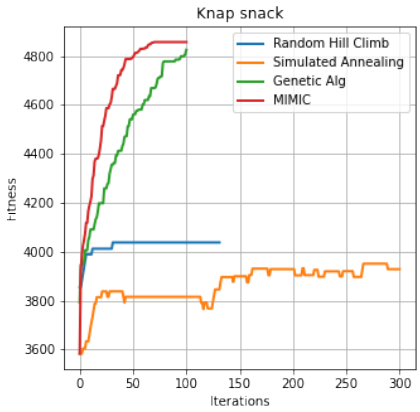
From the plots below, we can see by increasing the population size, there is an increase in fitness score, which is

because of the size of the problem, with **population size it covers the problem states better.** From the keep percentage plot, we see higher keep percentage works better for deep into the iterations and lower keep_pct works better for small iteration. This is because, with lower keep_pct we converge faster.



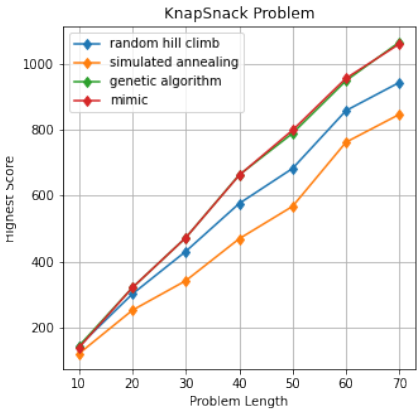


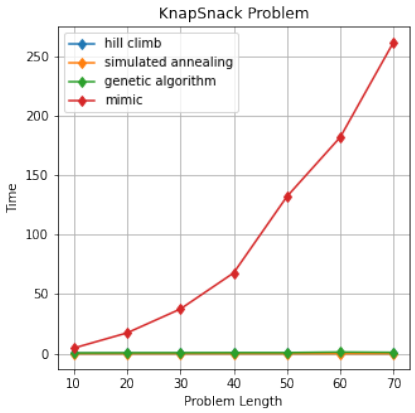
Knapsack, Comparison of Best Algorithms-



From both the MIMIC curves and the curve on the left, we see good convergence under 100 iterations for MIMIC. **As the problem favors the population algorithm, the GA is not yet converged under 100 iterations on running for 200+ iterations it surpasses the performance of MIMIC with same population size. But if we increase the population size, MIMIC performs better than GA as it is able to cover the problem states better with large population size.** Benefit of MIMIC is structural problems and problem with lower iteration constraint. when we compared the function evaluations, the GA and MIMIC does the same number of FEvals around 60,000.

Furthermore, from the plots (fitness over the size of problem and time over the size of problem) on the right, we can see the population algorithm works better on the knapsack problem as discussed earlier. Also, we see very high amount of time taken by MIMIC in comparison to other three algorithms, which is expected.

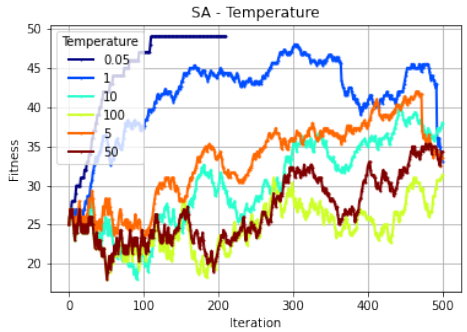




2.0.4 *Problem 3: OneMax Problem*

OneMax, is a simple optimization problem, It is the problem to find the binary string of given length with the maximum sum of its digit, which is all the digits will be 1's. **This means there is only one optimal solution with the whole state space as basin of attraction, which is why we choose this problem to show the benefits of SA.** RHC and SA should perform magnitude times faster than GA and Mimic for this problem.

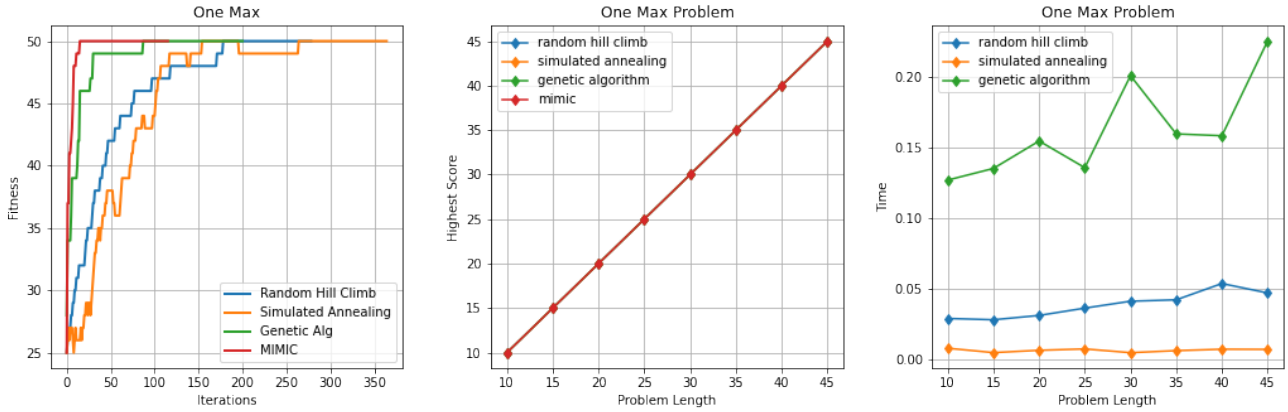
OneMax, Hyperparameter's tuning on the Algorithms-



For all the algorithms, the basic hyperparameters are enough to find the global optimal solution. To discuss SA, On our left we can see the plot for SA with different temperatures. For a problem size of 30. If the temp is 0.05 the algorithm converges the fastest and for higher temperatures it is still in explore mode after 500 iterations. **This is because as we discussed there is only one global optima, so basic hill climbing (Temperature low for SA) converges to the global optimal fastest.**

OneMax, Algorithms Comparison-

From the Figure 1 below, we can see the RHC and SA taking more iterations, but they are a lot faster than both GA and MIMIC. If we compare SA and RHC, SA is much faster, as in the below figure, plot (iii). **Conclusion:** - So for simple problems like OneMax with one global optima, SA is a better and faster solution to find the global optima.



In Addition to One MAX Problem, **SA is also good for problems where exploration is more important than exploitation. To explore, SA accepts bad steps with probability p, determined by a Boltzmann distribution.** Two factors impact this probability. 1) how bad is the next state, 2) temperature. When temperature is higher, the probability of accepting bad steps is higher.

3 PART 2: WEIGHT OPTIMIZATION OF NEURAL NETWORK

The Goal of this part is to find weights and biases for the Neural Network structure created in Assignment 1 using the random optimization algorithms and compare the Accuracy and other metrics of these feed forward Neural Networks with the one created using backpropagation.

3.1 Dataset and Previous Result Recap

Heart Failure Prediction Dataset - This dataset is created by combining 5 heart datasets over 11 common features. Cleveland: 303, Hungarian: 294, Switzerland: 123, Long Beach VA: 200, Stalog (Heart) Data Set: 270, Total: 1190 Duplicated: 272 Final dataset: 918 observations. All the 5 datasets originally sourced from UCI Machine Learning Repository.There are 918 rows 12 columns out of which 5 are continuous and outcome is

binary.Age, sex, fasting blood sugar, Exercise Angina and old peak are positively correlated to heart disease. The class distribution is as follows, 55.3% positive, 44.7% negative.(Almost Balanced). We did pre-processing on the dataset and then divided into two stratified splits, first split(70%) for training/validation and second split (30%) for testing. As false negative are most important for this dataset, recall evaluation is considered.

Best Neural Network Assignment-1 Recap - From Assignment 1, using supervised learning, the GridSearchCV, the Best neural network structure is 4 hidden layers with 4 nodes and learning rate of 0.1. This network performed good on the Dataset with accuracy of '0.915033'. Learning Curve with cv=5 and Confusion matrix from Assignment 1 is as below.

P.S. To have a fair comparison, we created the Backprop Neural Network again with same structure using mlrose_hiive and used same constraints max_iteration as for other algorithms. Loss Curve is as below.

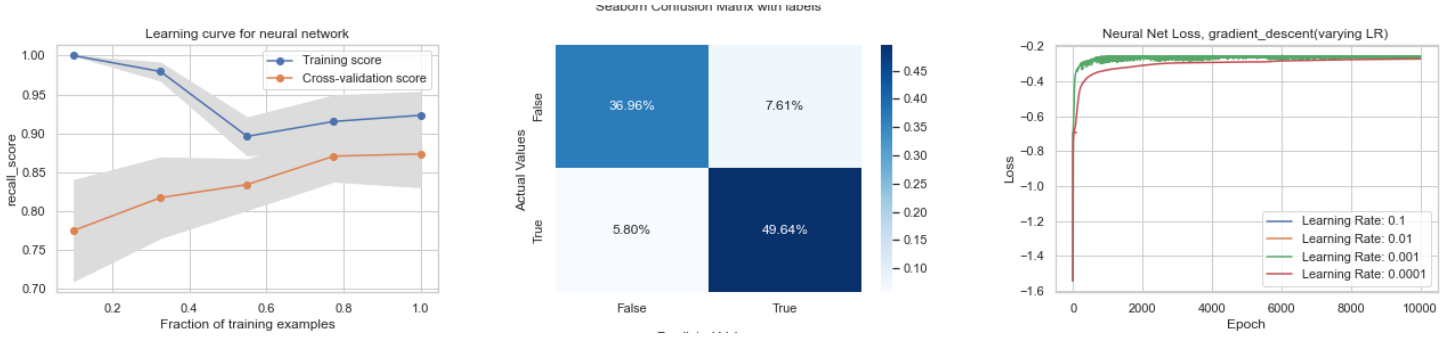


Figure 1—(a & b) Results from Assignment 1. and (c) Loss Curve (mlrose hiive implementation)

For this assignment, we have preference bias for simple models and those that are faster and easier to run for multiple iterations. We will discuss more on these preference constraints and hyperparameters for the mlrose implementation in the next section.

3.2 Neural Network Weight Optimization (mlrose Hiive) and Comparison

Common Methodology -

Paramaters	Values
Learning rate	0.1,0.01,0.001,0.0001(For All)
max_iters	10,000(For All)
max_attempts	default(np.inf)
random_state	42(For All)
restarts	10,25,50,100,200(For RHC)
pop_size	100,200,500,1000,2000(GA & MIMIC)
keep_pct	0.1, 0.4, 0.8(MIMIC)
temperature	0.05, 0.1, 0.5, 1, 3, 5, 8, 10, 20(SA)
mutation_prob	0.1, 0.25, 0.5, 0.8, 0.9 (GA)

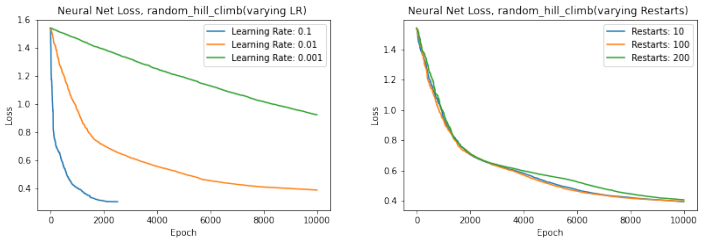
For network weight optimization, I have used mlrose hiive[4] NeuralNetwork function. For backpropagation as baseline, I have used 'gradient_descent' algorithm and compared it with feed forward weight optimization using 'genetic_alg', 'random_hill_climb' & 'simulated_annealing'. For all the four implementation I observed, the fitness score varies a lot with learning rate, so first we tuned the best learning rate for each of the algorithm, then we tune for the best parameters (listed in table), and later we compared these best networks for all algorithms based on different metric on our test and train data split.

Gradient Descent (GD)- As we can see in the Figure 1 (c), we varied the learning rate for GD algorithm, and we see the train accuracy for learning rate 0.1 and 0.01 was 0.5, for 0.001 as 0.84 and 0.0001 as 0.86. So for high learning rate, the NN with GD seems to miss the optima/ jump over the optimal, which suggest a narrow

basin of attraction, which works better for low learning rate as 0.0001 for GD.

Random Hill climbing (RHC)-

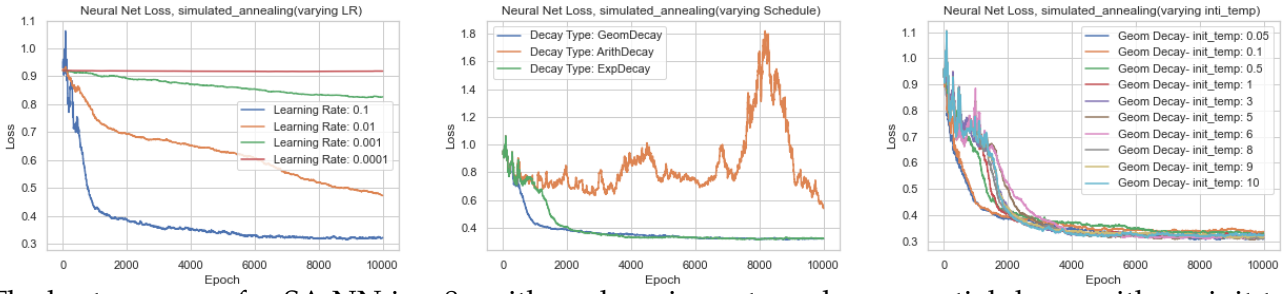
From RHC plot, we see it performs and converge better with the higher learning rate under 10,000 iterations, also, at higher learning rate, we don't see much impact of varying restarts, it suggests the RHC finds this optimal within 10 restarts. For low learning rate, the RHC plots are not converged within 10,000 iterations. Best train accuracy for RHC is 0.86(learning rate 0.001 and restart 200)



As the weights are continuous values, RHC performs better with NN, it works similar to GD and climb neighboring better points to local optima. This is true for SA too, as for low temp SA is Hill climbing.

Simulated Annealing (SA)-

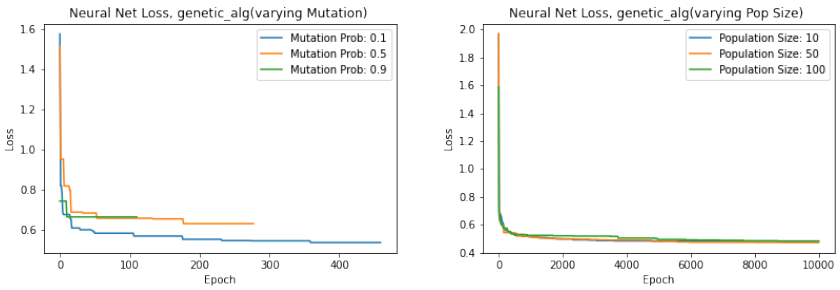
From SA plots, we see similar information for learning rate as RHC. The lower learning rates didn't converge in 10,000 iterations. As SA accepts bad step when the temp is high, we see high fluctuation on the plots for lower iterations and then the plots smoothens when the temp is high. The temperatures didn't affect much with best learning rate 0.1 as the dataset doesn't seem to have multiple local optima and all the learning rates converge to a similar score under 10,000 iterations.



The best accuracy for SA NN is 0.87 with 0.1 learning rate and exponential decay with 0.5 init temperature, **this also agrees with RHC and SA working better with continuous weight problem.**

Genetic Algorithm-

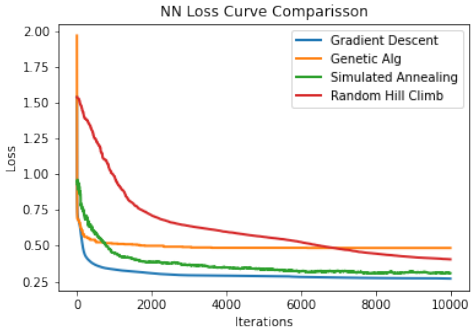
For the Genetic Algorithm, first we ran our experiments with max attempt as 100 for different learning rate and mutation probability(0.1 to 0.9) population sizes (10 to 1000) with max attempt 100, but it ran only for 400 iterations before converging with max accuracy of 0.74 for population size 2000 and mutation 0.1. 400 iteration were not enough, we increased the max_attempt to, 10000 and ran it for population size (10,50 & 100) as in the second plot. We see the Network learns fast in comparison to RHC and SA, but it converges at high loss. **In GA, mutation provides exploration and crossover leads to exploitation to converge at a good solution. It seems we might need to try a different crossover technique**



in the future work. For comparison, we choose pop size 100, mutation rate 0.1 with train accuracy 0.77.

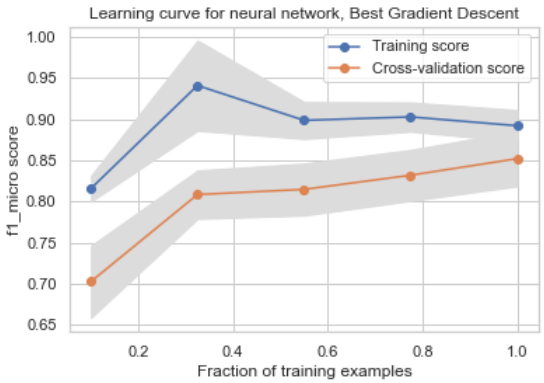
Metric Comparison for Selected Best Neural Networks -

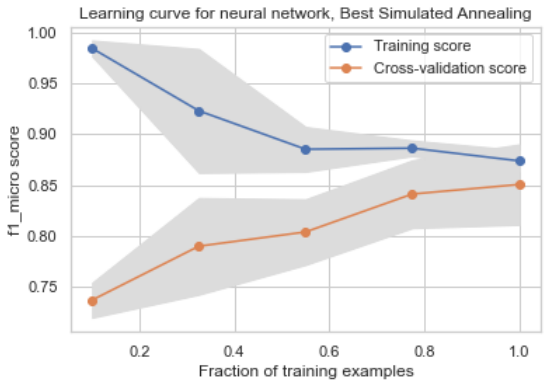
From the plot below for all the algorithms, we can see the RHC and SA did not converge for 10,000 iterations. When we run GD and SA for 15,000 iterations, it converges very close to each other around 0.25 Loss. I couldn't run GA with different crossover technique to increase exploration which I will do in future work.



	Algorithms	Recall Score, Train	Recall Score, Test	Time Train	Time Infer
0	Gradient Descent	0.883648	0.861629	21.056030	0.002000
1	Genetic ALG	0.771831	0.768930	1118.613122	0.002005
2	Random Hill Climb	0.836207	0.859796	2824.175060	0.001999
3	Simulated Annealing	0.878608	0.876216	16.726037	0.003015

From the table on the right, It shows RHC take the highest train time and GA score's the lowest, this is because I didn't train GA with population size greater than 100, and we had a preference bias to choose simpler model. **Also, SA performs better than RHC, show exploration is more important than exploitation for this problem, which also encourages me to try different mutation function for GA.** As we could run SA and GD for 15,000 iterations, they converged to a similar score, I have plotted the learning curve for both as below.





From these curves we can see for low data the variance is less for GD plot in comparison to SA plot. GD is definitely better for this data. Also, if more data provided, the model's will perform better as CV score is well below Training score.

Further work -

I would like to see how GA performs on this data with a different mutation function and high pop size. Also would run RHC longer to check what iteration the RHC converges for this problem. In addition to this, a complex dataset will have different challenges and would be a good learning to explore.

REFERENCES

[1][2][3][4] Mlrose-hiive. (n.d.). PyPI.
https://pypi.org/project/mlrose-hiive/