Comparative study of Randomized Optimization Algorithms for N-Queens problem, Knapsack Problem, and Neural Network weights optimization

Umesh Jadhav
OMSCS
Georgia Institute of Technology
Georgia, USA
ujadhav6@gatech.edu

***Abstract*—Randomized Optimization is a problem solving technique leveraged to find an optimal solution among a set of random candidate solutions by scoring them. Though these solutions are in place, they might not be optimal in terms of various cost factors such as loss and fitness score. Hence, for randomized optimization to work, three entities namely input search space, a fitness function and the output (goal) are defined. Initially, the fitness function processes a randomly generated candidate solution as input space and map them to a fitness score, while iteratively producing new candidate solutions; where the expected result is to get the solution with the highest fitness score. In this study, three randomized optimization algorithms specifically, Randomized Hill Climbing (RHC), Simulated Annealing (SA), and Genetic Algorithm (GA) are comparatively studied by implementing them over N-Queens problem and Knapsack problem. Further, a neural network is developed using Back-Propogation technique in addition to the three randomized optimization algorithms and contrasted over performance measures. Here, it is found that for N-Queens problem with hyper-parameters tuning, Simulated Annealing algorithm outperforms Randomized Hill Climbing algorithm and Genetic algorithm in terms of fitness score with a faster runtime. Whereas for Knapsack problem, the Genetic algorithm showcased a better fitness score than the other two algorithms. However, the wall clock time for the former grows exponentially with the increase in problem sizes. For Neural network,Randomized Hill Climbing algorithm outperformed Simulated Annealing, Genetic algorithm along with Gradient Descent strategy for prediction accuracy and loss metric.**

I. Introduction

In Computational theory, every problem has an approximate solution, however there exist problems having multiple solutions. The best solution to the problem depends upon its efficiency in terms of cost that is incurred while arriving at the solution or the score that determines the fitness of the solution. The problems where cost is the criterion, it is required to find the solution with least possible value, whereas for problems where fitness score is the metric, the solution with maximum value is chosen. This is required as the algorithms in use might arrive at a solution which is not optimal, known to stuck at local maxima or local minima and might never reach at the desired state represented by global maxima or global minima. As a result, various optimization algorithms are developed that aids in finding the optimal solution with loss and fitness scores mapped to each of the solutions. These algorithms initially determine a random candidate solution based upon the defined nature of the solution, then optimize it on every iteration with specific number of attempts per repetition. At the end, the results are generated comprising of loss score and fitness score over every iteration.

In this study, Randomized Hill Climbing Algorithm (RHC), Simulated Annealing Algorithm (SA), and Genetic Algorithm (GA) are implemented over N-Queens problem and Knapsack problem to find the best solution with maximum fitness score in the first section. The second section deals with the implementation of the same three algorithms with the addition of Back-Propogation algorithm to find the best set of weights of a neural network model for Company Bankruptcy dataset.

Finally, all the algorithms in their respective problem sections are compared for fitness score over iterations for a specific problem size, fitness score over various problem sizes, the wall clock time incurred by these algorithms while also analysing the effect of hyper-parameters tuning on the fitness scores. In case of Neural network section, the loss curve replaces the fitness curve hence, the visualizations and results are analysed based upon the loss metric over iterations, model scalability through fit time over various training sized and the effect of hyper-parameters tuning on the prediction accuracy of the models.

II. Optimization Problems

For the scope of the study, discrete state optimization problems are preferred which means that every intermediate or final state are discrete.

A. N-Queens Problem

In N-Queens problem, the queens are placed on a chessboard of N*N dimension such that none of the queen intercepts vertically, horizontally or diagonally with one another. This problem has multiple solutions for size greater than 4. Here, as stated in lecture, the randomized hill climbing algorithm might get stuck at local maxima depending on the initial state it has chosen. However, in case of Simulated annealing, if a higher value for the initial temperature is chosen, it gives an increased

chance for the algorithm to explore different solutions, as it determines the probabilistic movement from one solution to the another based upon its fitness score. As for genetic algorithm, instead of random exploration, the algorithm performs crossover between the two random initial inputs for creating an offspring with further mutation. As a result, instead of completely changing the solution, the placement of queens bit by bit is done replacing the least fitted queen with the queen giving higher fitness score. Hence, genetic algorithm is expected to give better results than that of Randomized Hill Climbing and Simulated Annealing algorithms. However, the crossover and mutation in genetic algorithm on every iteration makes it a time consuming approach, making its choice debatable. Additionally, hyper-parameters are tuned for these algorithms to study their effect on the performances.

B. Knapsack Problem

Knapsack problem is defined as, given N items having weights w_i and values v_i associated with them, and maximum knapsack capacity W, place these items in the knapsack such that their total weight is less than W while the total value is maximized. As hypothesized in case of N-Queens problem, Genetic algorithm might outperform Simulated Annealing as it chooses highest scored solution and mutate it, with replacement for one of the least scored item in the knapsack. This effect is intuited as the algorithms are tested for incremental sizes of items in the knapsack. This is due to the fact that as the problem size increases, the search space for Simulated annealing also increases and making the probabilistic movement from one solution to another less feasible than that of the Genetic Algorithm in terms of fitness score. On the other hand, Randomized Hill Climbing will suffer from the problem of local optima as the problem size increases. Higher fitness score due to the population size of the Genetic algorithm would see a trade off with the runtime as compared to its compatriots.

C. Neural Network

In the second section, four Neural network models are developed for Company Bankruptcy dataset with Back-Proprogation approach along with Randomized Hill Climbing, Simulated Annealing, and Genetic Algorithm to optimise the weights and achieve higher accuracy with reduced losses. Here, the task of weights optimization is continuous state optimization problem as opposed to the N-Queens problem and Knapsack problem.The expected behaviour here is for the optimization algorithms to outperform the back-proprogation algorithm in terms of better fit of the model over various training and test sizes due to the fact that the fitness function is replaced by a pre-defined loss function with the goal to find the set of weights with least loss values. However, given the randomized search behaviour of the optimization algorithms, Back-proprogation strategy would outperform the other three algorithms in terms of runtime.

As for the Company Bankruptcy dataset[1], it consists of 96 features and 6819 observations with the dependent feature as "Bankruptcy?". The independent variables are of continuous nature whereas the dependent variable is a binary classification label.

III. Methodology

The entire experimentation is done using Python programming language with mlrose-hiive[2] providing the core framework for implementing the randomized optimization algorithms. For visualization purposes, matplotlib library is leveraged with an additional usage of pandas, numpy, and scikit-learn library for smooth function. Experimental methodology for N-Queens problem and Knapsack problem is illustrated in the left flowchart in figure 1, while for neural network it is illustrated in the right flowchart. The scope of this study is to get the solutions with maximum fitness score, hence the $maximise$ parameter for $DiscreteOpt()$ which specifies the fitness function to follow discrete state from mlrose-hiive is set to $True$ for N-Queens problem as well as Knapsack problem. To avoid "lucky run" concept, the algorithms are subjected to three random state values of [8, 29] for N-Queens problem and Knapsack problem.

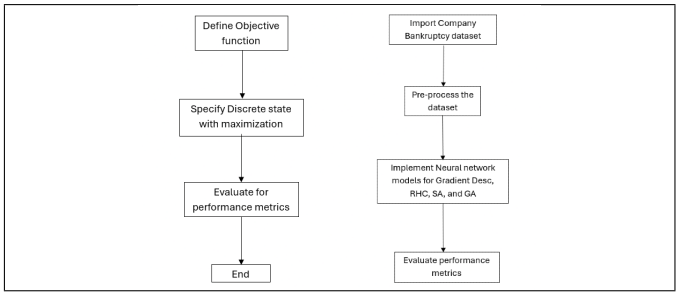

Fig. 1: Left - Experimental Methodology for N-Queens Problem and Knapsack problem. Right - Experimental Methodology for Neural Network problem

A. N-Queens Problem

N-Queens problem is implemented using a custom fitness function as per the pseudo-code in Algorithm 1. The objective is to increment the score each time the queens are in proper position as per the problem definition. For implementation of RHC, SA, and GA, their respective classes from mlrose-hiive are instantiated, with maximum iterations of 5000 for maximum attempt of 100 searches per run.

Algorithm 1 N-Queens Problem

```
for Q1 = 0, ..., (N − 1) do
    for Q2 = (Q1 + 1), ..., N do
        if position(Q1) != position (Q2) then
            score+ = 1
        end if
    end for
end for
```

First, the effect of hyper-parameters is studied for the three algorithms. First, RHC is tested for $restart$ for values between 10 and 50. Secondly, SA is analysed over initial temperature

for values between 1 and 5 whereas for decay, values between 0.001 and 0.1. GA is tested for mutation probalility and population size ranging between the values of 0.1 to 0.9 for the former and 2 to 16 for latter. As per the results of these hyper-parameters tuning, RHC restart is set to 29. For SA, Geometric Decay is used with the initial temperature of 1 and decay of 0.05 with minimum temperature of 0.0001. For GA, the mutation probability is set to 0.1 while the population size corresponding to the N. These settings are same for Fitness score per iteration evaluation, Fitness score for various problem size evaluation, Function evaluations per iteration, and wall clock time since the hyper-parameter tuning for every problem size took long hours for computation.

Finally, the fitness score on each evaluation is averaged over the random states defined and analysed to compare the trade-offs of the fitness score, convergence and the runtime of the three algorithms.

B. Knapsack Problem

In case of Knapsack problem, $Knapsack()$ fitness function from mlrose-hiive for weights and values randomly generated between the range of 0 to 200 and maximum knapsack weight percentage is set to 0.9. First various values for hyper-parameters are tested for respective algorithms. As per the results, $restart$ is set to 29 for RHC whereas for SA, geometric decay with initial temperature of 1, decaying at rate of 0.005 with minimum temperature of 0.0001 is preferred. GA is implemented with mutation probability of 0.3 while the population size corresponds to the number of items(N) for the Knapsack.

For experimentation, fitness score per iteration for 100 Knapsack is performed, followed by the assessment of fitness score for incremental problem size between 10 to 100. Function evaluations are analysed over the 100 iterations with the wall clock time taken by the three algorithms for execution. In all these demonstrations, the fitness score is averaged for the random states defined. The experiments follows similar settings across all the tasks as runtime for hyper-parameters tuning increases significantly with increase in problem sizes.

C. Neural Network

In this section four neural network models are developed using mlrose-hiive $NeuralNetwork()$ class where the strategy of weight optimization is defined by $algorithm$. Here, the model from previous study (Assignment 1) is freezed for settings such as learning rate of 0.001, activation function and the number of neurons. To implement Back-Propogation approach, the value is set to "gradient_descent" whereas, the other three randomized optimization algorithms are specified as "random_hill_climb", "simulated_annealing", and "genetic_algorithm" respectively with a single layered neural network of 48 neurons. The activation function is set to sigmoid as it calculates the probability of the final value being one of the outcome label interpreted as values below 0.5 to be 0 and greater than 0.5 to 1. To analyse the effect of loss curve, the loss score generated by $fit()$ of the parent class is compared for 100 and 200 iterations with maximum attempts of 100 is implemented for training size of 75% of the total data before terminating with a single random state of 8. Also, the maximum weights value is restricted upto 11 which forms state boundaries of (-11, 11). Additionally, the prediction accuracy is compared of the four algorithms is also compared. Finally, function evaluations, model scalability are some of the metrics for which the performance of the four strategies are analysed.

IV. Results

Please note here that, all the time related metrics may vary depending upon the system configurations.

A. N-Queens Problem

Here, the maximun number of iterations for every randomized optimization algorithm is set to 5000 with maximum attempts of finding better solution at each iteration of 100.

1) Hyper-Parameter tuning

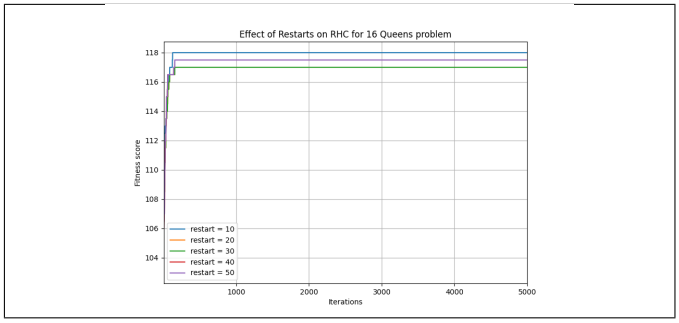

Fig. 2: Effect of Restarts of RHC on Fitness Score for 16 Queens problem.

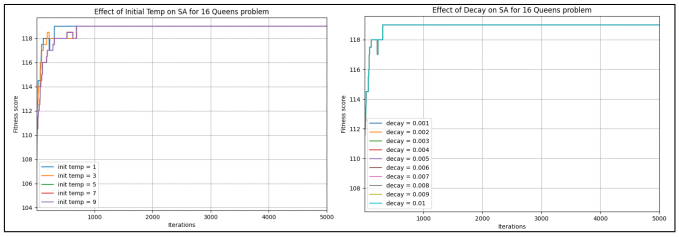

Fig. 3: Left - Effect of Initial temperature of SA on Fitness Score for 16 Queens problem. Right - Effect of Decay of SA on Fitness Score for 16 Queens problem.

From figure 2 it is evident that the value of 10 yields higher fitted soltions for RHC with 10 restarts. On the other hand, the two sub-figures from figure 3 for influence of Initial temperature and Decay for SA indicate that all the values of initial temperature converges to a maximum fitness score of 119 however, the rate of convergence is the difference making factor where, initial temperature of 1 showcase best convergence than its compatriots whereas. Similarly, while inspecting the left sub-figure of figure 4, population size of 11 and 16

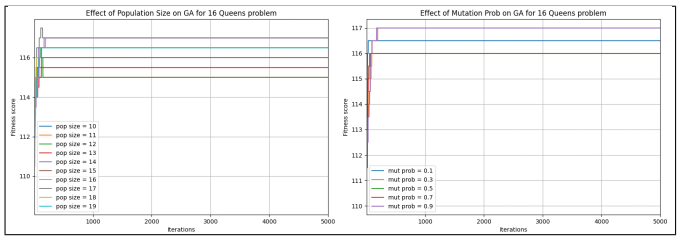

Fig. 4: Left - Effect of Pop size on Fitness Score for 16 Queens problem. Right - Effect of Mutat prob on Fitness Score for 16 Queens problem.

produce optimal fitness scores among all. Whereas from the right sub-figure indicate that the mutation probability of 0.3 and 0.9 generate give best fitness score among all. As a result, the population size is set to N and mutation probability of 0.3 is chosen for further experimentation.

2) Fitness score per Iteration
As evident from figure 5, SA showcase better fitness score for 16 Queens problem over RHC, and GA. The fitness score generated by SA shows a gradual increase during earlier iterations before stabilising its highest score pf 119 over the next iterations. Though, the convergence shown by GA is greater than the other two, it fails to reach the heights achieved by SA, limited to a fitness score of 116. As for RHC, it follows similar trend as that of GA, however it fails to outperform the latter by a minor degree.

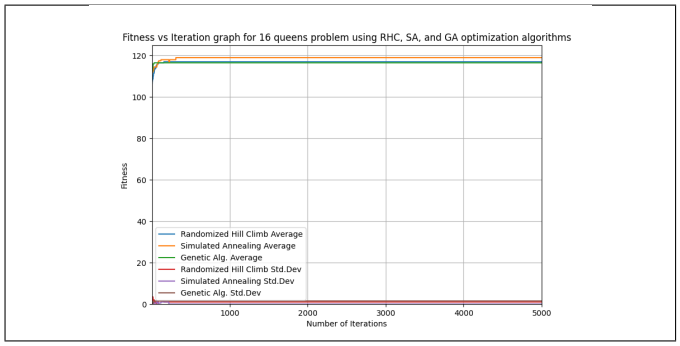

Fig. 5: Fitness score per Iteration graph of RHC, SA and GA for 16 Queens problem.

3) Fitness score per Problem Size
As illustrated in table 1, SA clearly outclass RHC and GA for every problem size, though the difference between the fitness scores of the three algorithms is minute. SA begins with a fitness score of 44.42 while growing upto the value of 4932 for the last size. GA falls short to achieve this score by 1 point difference. However, a major difference between the scores of RHC and SA can be seen.

4) Function Evaluations per Iteration
As seen in figure 6, the number of function evaluations taken by GA is highest, showing a value of 6000 average

TABLE I: Fitness Score over Number of Queens for N-Queens problem.

Number of Queens	Fitness Score		
	RHC	SA	GA
10	43.9736	44.4233	42.4952
20	187.8115	188.3619	185.9387
30	430.1321	431.8075	428.8682
40	771.4854	774.1842	770.8317
50	1215.944	1218.44	1212.757
60	1758.446	1763.039	1760.034
70	2396.44	2405.005	2402.38
80	3141.67	3147.752	3146.175
90	3981.697	3989.901	3990.025
100	4924.454	4932.283	4931.035

iterations. This is followed by RHC where the average number of function evaluations reaches 2000. Finally, SA showcased least number of function evaluations amongst the three algorithms though, the iterations taken to stabilise is higher than that of RHC and GA.

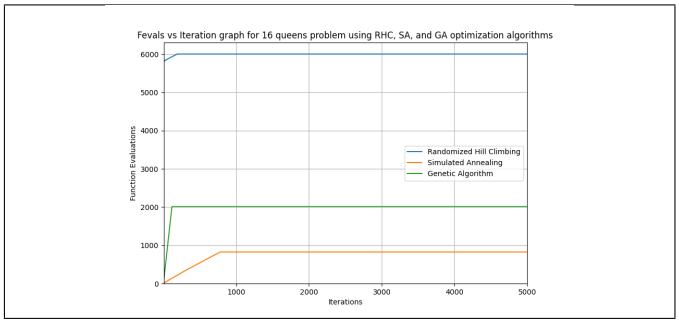

Fig. 6: Function Evaluations per Iteration graph of RHC, SA and GA for 16 Queens problem.

5) Wall Clock Time per Problem Size
From figure 7, it is understood that the wall clock time is directly proportional to the number of queens for all the three algorithms. Starting with SA, the runtime over each of the problem size does not cross the 50 seconds mark. This trend is followed by RHC though it shows an increase in runtime as compared to SA. Finally, an exponential increase in wall clock time for GA could be seen as the number of queens increases beyond 40, reaching the highest time of 280 seconds approximately.

B. Knapsack Problem

1) Hyper-parameters Tuning
Figure 8 illustrates the effect of various values of restart of RHC for 100 Knapsack problem, in which all the values moves in a closer proximity to each other however, the value of 50 gives best fitness score. On the other hand, the left sub-figure from figure 9 shows all the values of Initial temperature ranging between 1 to 9 converge to the same value of 4840 with constant convergence rate. A similar trend is visible for the effect of decay of SA from the right sub-figure of figure 9 where all the values converges to fitness score of

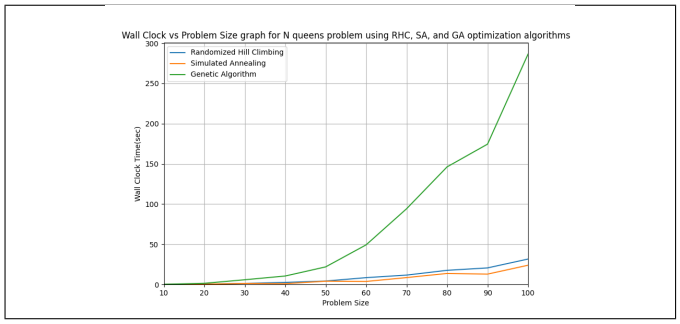

Fig. 7: Wall Clock Time per size graph of RHC, SA and GA for N-Queens problem.

4840. Figure 10 demonstrates the effect of population size and effect of mutation probability from GA in left and right sub-figures respectively. In case of Population size, all the values converges to fitness score of 5210 approximately except the size of 20. Similarly, mutation probability of 0.1 shows reduced fitness score than the other values of the same where the score reaches the height of 5215.

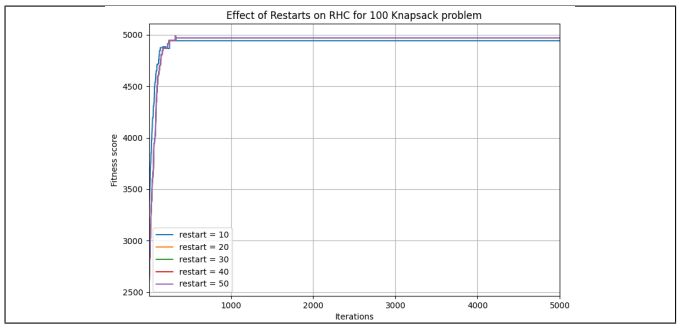

Fig. 8: Effect of Restarts of RHC on Fitness Score for 100 Knapsack problem.

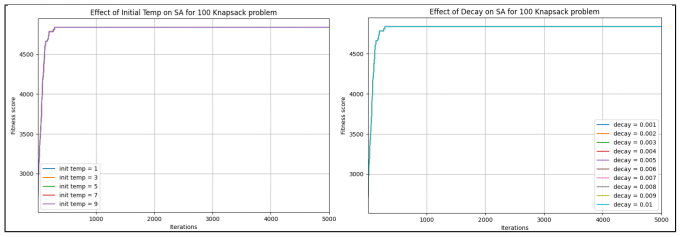

Fig. 9: Left - Effect of Initial temperature of SA on Fitness Score for 100 Knapsack problem. Right - Effect of Decay of SA on Fitness Score for 100 Knapsack problem.

2) Fitness score per Iteration
As illustrated in figure 11, fitness score for 100 items in Knapsack is analysed where similar trend and convergence rate between RHC and SA can be seen. However, the former outperforms the later in terms of better solutions represented by the average fitness score of 5000 and 4800 respectively. For GA, a better convergence can

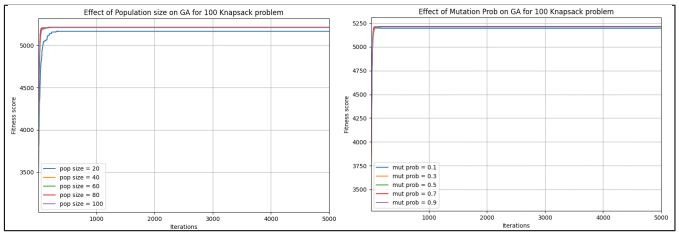

Fig. 10: Left - Effect of Pop size on Fitness Score for 100 Knapsack problem. Right - Effect of Mutat prob on Fitness Score for 100 Knapsack problem.

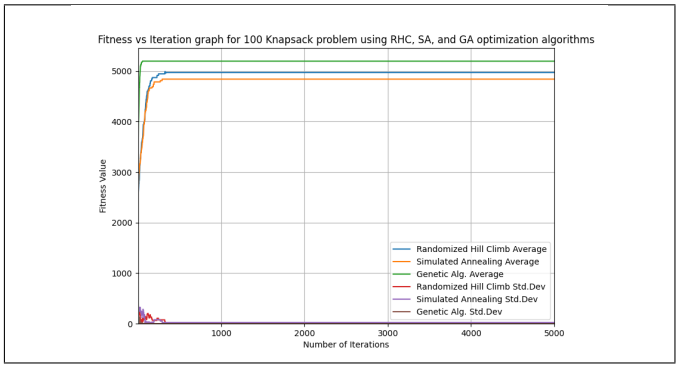

Fig. 11: Fitness score per Iteration graph of RHC, SA and GA for 100 Knapsack problem.

be seen than the other two algorithms along with the highest fitness score of 5300 approximately, providing better quality solutions.

3) Fitness score per Problem Size

TABLE II: Fitness Score per Item size for Knapsack problem.

Number of Items	Fitness Score		
	RHC	SA	GA
10	40.993	30.0072	41.0039
20	164.8865	127.9202	179.8887
30	356.6327	355.7861	408.7944
40	579.489	453.9375	637.1357
50	998.7116	845.3531	1167.988
60	1385.793	1160.499	1620.851
70	1862.133	1540.745	2241.219
80	2592.112	2555.078	3214.446
90	3175.509	2839.098	3759.097
100	3102.539	2525.343	3825.142

For every problem size in table 2, Genetic algorithm beats the other two algorithms by a large margin of fitness score as evident from the table. Specifically, SA provides worst results among others, followed by RHC. However, the difference between the fitness score of the two is not as significant as that of the difference between the fitness scores of RHC and GA.

4) Function Evaluations per Iteration
From figure 12, it is clear that SA takes the least number of function evaluations to arrive at the solution, showing an average of 50 approximately. Whereas, RHC stands

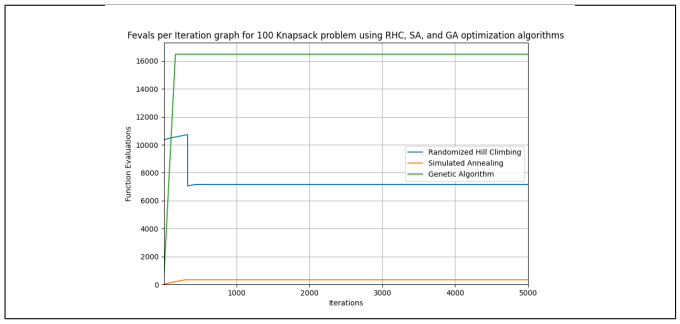

Fig. 12: Function Evaluations per Iteration graph of RHC, SA and GA for 100 Knapsack problem.

at second position displaying a minor increase before decreasing steeply at iteration mark of 200, carrying the same trend thereafter. On the other hand, GA crosses 16000 mark for function evaluations quickly, stablizing for the remaining iterations.

5) Wall Clock time per Problem Size

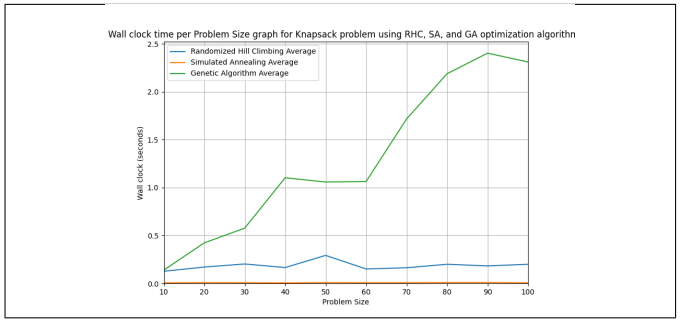

Fig. 13: Wall Clock Time per Problem Size graph of RHC, SA and GA for N Knapsack problem.

From figure 13, it is clearly seen that GA takes the highest runtime for incremental item number. First, a linear growth is evident from the trend followed by GA, starting with minimum of 0.20 seconds for 10 items and crossing 2 seconds mark for 100 items. Secondly, RHC and SA maintains their runtime throughout the experiment, though former taking a bit longer than the later. In case of RHC, an average runtime of 0.1 seconds can be approximated whereas, an average of 0.08 approximated for SA.

C. Neural Network

1) Hyper-parameters tuning
First, RHC is tested for restart values of 10, 50 and 100, where value of 10 gives loss of 0.22, followed by loss of 0.29 for 50 and 0.704 for 100. For SA algorithm, initial temperature and decay is tested for values between the range 1 to 100 and 0.01 to 0.1 respectively. For both the hyper-parameters, the loss metric shows an initial loss of 0.602139 and ends with the loss of 0.602125 with a step of increase and decrease between the iterations. Similarly, in case of GA, the losses generated by mutation probability and population size, identical pair of curves are seen where, the loss starts at 0.602139 before decreasing at 40^{th} iteration. However, an instant rise is seen at 50^{th} iteration before culminating at a value of 0.602178.

2) Loss Curve
From right sub-figure of figure 14, it is evident that all the randomized optimization algorithms outclass the Gradient descent strategy. Specifically, from the left sub-figure of figure 14, GA appears to produce least losses over 100 iterations, having an average of 0.45 approximately. It is followed by SA with a constant loss of 0.625; RHC following the trend with a loss of approximately 0.73. In this case, the Gradient descent strategy showcase spiked pattern, reaching the heights of 2.25.
A notable observation can be seen in the right sub-figure of figure 14 in which RHC appears to outclass GA along with the other algorithms; standing at constant loss of 0.15. Also, the losses in case of GA are reduced over the iterations, culminating at the loss of 0.25. However, SA and Gradient descent algorithms do not show any difference between the left sub-figure and the right-sub-figure.
As for the accuracy score, RHC leads the race with test accuracy score of 96.22%, followed by GA with the test accuracy of 95%. However, Gradient descent strategy and SA giving similar test accuracy of 83.07%.

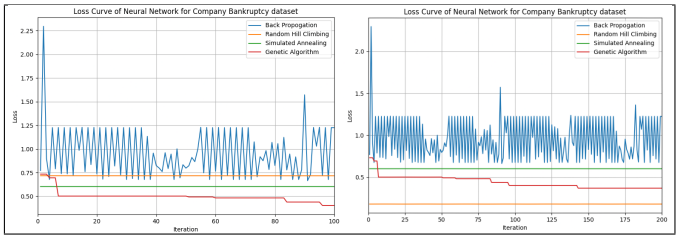

Fig. 14: Left - Loss Curve for 100 Iteration. Right - Loss Curve for 200 Iterations.

3) Type I and Type II errors

TABLE III: Type I and Type II errors of Neural Network for Gradient Desc, RHC, SA, and GA

Algorithm	Type 1 error	Type 2 error
Gradient Desc	217	37
RHC	2	51
SA	217	37
GA	22	51

From table 3, the type 1 error for Gradient Descent and SA are similar with a value of 217, followed by GA with the error measure of 22. Whereas, RHC illustrate the lowest type 1 error of 2. On the other hand, Gradient Descent and SA follow same numbers in case of Type 2 error while RHC pairing with GA.

4) Function Evaluations per Iteration

From figure 15, Function evaluations per Iteration is highlighted for RHC, SA, and GA algorithms only in case of Gradient descent, such metric is not generated. Beginning with RHC, the function evaluations moves with a constant linear rate with average number of function evaluations of 100. A similar trend is closely followed by SA, averaging approximately at 3000. Finally, higher rate of growth is witnessed in case of GA, reaching to the height of 16000 function evaluations for the last iteration.

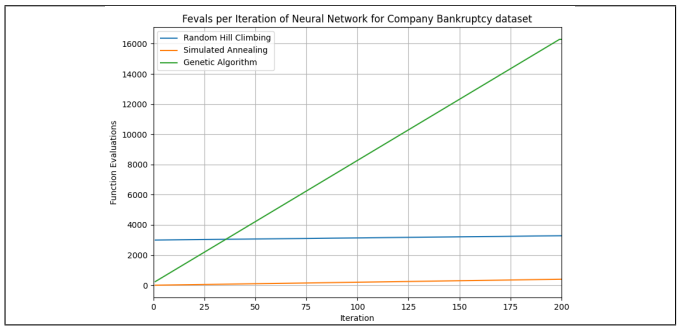

Fig. 15: Function Evaluations per Iteration Curve of Neural Network for Gradient desc, RHC, SA, and GA algorithm.

5) Model Scalability

TABLE IV: Fit Time of Gradient Desc, RHC, SA and GA for Neural Network over various train size

Train Size (%)	Fit Time			
	Grad Desc.	RHC	SA	GA
10	10.202141	78.366023	16.463391	2804.585503
20	14.805642	117.897614	15.855767	3616.181219
30	22.444699	165.369797	29.289307	4279.175424
40	42.183515	471.625738	58.895289	4908.199537
50	55.218388	663.591995	46.140555	4889.947578
60	159.856941	789.607877	137.748154	6052.500094
70	174.098531	1156.368558	126.992905	6782.794833
80	197.572059	1189.525564	116.565058	6795.728142
90	187.848927	1282.314162	164.193000	6846.977770

Model scalability defines the fit time and score time of the model over various training sizes which identify the suitable model for weight optimization considering the trade-offs between the time and accuracy. Starting with the Fit time from table 4, Gradient descent approach showcase lower runtime for initial training sizes before being overtaken by SA after 50% size mark where the latter finishes its process within 138 seconds. Secondly, a gradual increase in the fit time for RHC is evident during the higher training sizes, with maximum of 1282 seconds for the final size. Finally, GA begins with a fit time of 2804 seconds, rising quicky during the next training size before slowing down upto 50% of the training size followed by a mixture of rise, plateau and a fall crossing 6846 seconds during the final run. However, in case of Score time, none of the four algorithms breaches 1 seconds mark.

V. Discussion

A. N-Queens Problem

As per the results seen from figure 5, for 16 Queens problem, SA outperformed RHC and GA in terms of fitness score. Also, for various number of Queen, the majority of sizes witnessed higher fitness score of SA as that of the other two algorithms. Similarly, the results from figure 6 indicate that to achieve better fitness scores, the number of function evaluations are minimal for SA. These results as a whole highlight the strength of SA algorithm, where the tuned hyper-parameters over a particular problem size showcased their effect on generating better quality solutions to not only a particular number but for incremental problem sizes also. However, a trade-off between the fitness score of RHC and function evaluations can be seen, making it the leader for the later metric though beaten by SA for fitness score. The influence of these settings is also evident through wall clock time the algorithms had taken over various problem sizes in which, SA is seen to have maintained the lowest runtime, followed by RHC with a huge exponential difference shown by GA.

While analysing the results, although RHC stood at a closer proximity of the two algorithms, its inherent factor of getting stuck at local minima contributed to its shortcomings. In case of GA, it works by initiating with set of solutions in form of initial population, iteratively replacing the worse solutions with the better ones. Whereas SA leverages wide search space and making it narrow over the run, specified by initial temperature and minimum temperature. However, the hyper-parameter tuning is the contributing factor behind the increased performance of SA over GA. Here, the N-Queen problem have multiple solutions which make it favourable for SA as its choice of moving to better solution increases the probability to find better solutions. However, this affects GA as the number of local optima aggravate GA's performance. Also, although GA have greater chances of generating good quality solutions with higher fitness score, trade-off with the wall clock time makes its choice debatable.

Hence, the null hypothesis that GA will provide better fitted solutions than RHC and SA is rejected as evident from the results, SA with the hyper-parameters tuned outclasses both RHC and GA in terms of function evaluations also.

B. Knapsack Problem

For Knapsack problem, results from figure 11 illustrating Fitness score over iterations and table 2 illustrating fitness score over various problem sizes, indicate the best performance of GA over RHC and SA algorithm. GA converges to the highest value merely in least number of iterations whereas, RHC and SA takes additional iterations to do the same, yet unable to provide better solution. Secondly, when tested for various sizes, SA is outperformed by both RHC and GA by a significant degree, where GA showcase higher fitness score for each of the problem size. This indicate that SA suffers from the problem of getting stuck at local minima. However, the

number of restarts have clearly benefited RHC which enables the algorithm to escape from local minima by starting at a random point in the search space if the solution hits lowest score.

On the other hand, from figure 12 and figure 13, it is understood that the better quality solutions generated by GA have a trade-off with the number of function evaluations and the runtime of the algorithm. Here, GA has taken twice the number of function evaluations as that of the RHC whereas, the poor performance of SA is justified by the least number of function evaluations and wall clock time. Additionally, the Knapsack problem does not have multiple solutions, but can have various solutions with similar fitness scores. Due to this, although SA have widespread search space, the chances of reaching the optimal solutions is lower due to its inherent nature. Whereas, in case of GA, since a single optimal solution is desired, it performs well given the mutation and crossover with the population, wherein, the initial set of solutions do not have much influence on it.

As a result, the null hypothesis that GA will overpower RHC and SA in terms of fitness score over various problem sizes is accepted. Also, as stated that RHC will suffer from problem of local optima is rejected as it performs better than SA though standing behind GA. The third part of the null hypothesis that a trade-off between the runtime and Fitness score of GA is accepted although it is of lower degree.

C. Neural Network

Firstly, during hyper-parameters tuning, every value of the hyper-parameters of their respective algorithm generates similar loss curve hence, the further experiments are carried out choosing one of the values from the set. Secondly, the effect of iterations on the loss curve and accuracy is evident as RHC performs incredibly well during the 200 iteration, beating every algorithm under test with the highest accuracy. The same is also visible in the case of GA where the loss is reduced over the iterations however, SA is not affected in both the cases. Since the weight optimization task is continuous state problem, the randomized algorithms lays a grid for the weights space specified by max_pct and try to achieve the global minima since the objective is to reduce the losses. The loss curve comparison for 100 and 200 iterations indicate there exists a single global minima of specific set of weights. Hence, given the nature of the problem, with an increase in the number of iterations, the loss generated by RHC are reduced which is seen in case of GA also where a stepped decrease in the loss is seen. However, SA is not affected by the changes in iterations as the learning rate is not favourable by it which is evident from the loss curve of Gradient descent also. Hence, the null hypothesis that randomized optimization algorithms would outperform Back-Propogation (Gradient Descent) is accepted since the former fails to optimize weights. Also, the second null hypothesis that Gradient descent would outperform the other three algorithms in terms of runtime is rejected which is evident from table 4.

VI. Conclusion

In the first section of the study, RHC, SA, and GA algorithms are implemented over N-Queens problem and Knapsack problem, analysed over various metrics with respect to fitness score, function evaluations, and runtime. In N-Queens problem where multiple solutions are available, SA performs better than GA due to its inherent behaviour of probabilistic movement over the points in the search space. The choice of initial temperature increases its movement and land to various points closer to the global optima. In case of GA, the intermediate mutations and crossovers is a tedious task and takes prolonged duration, yet it fails to arrive at better fitted solutions than that of SA. As for RHC, it typically suffers from the problem of local optima although being tested for various restarts. Finally, the effect of hyper-parameters tuning is clearly evident in this case where SA outperformed GA along with RHC. For further studies, if the number of iterations and maximum attempts are increased, notable differences might be visible. Also, Arithmetic Decay and Exponential decay can be comparatively analysed. Whereas, parameters other than mutation probability and population size for GA can be leveraged, tested and analysed.

As for Knapsack problem where a single solution exists, GA outperforms SA because of mutations and crossovers that replaces the least fitted solutions with the best ones and thus moving closer to the global optima. Additionally, the number of function evaluations for GA are greatest however, RHC provide better solutions than that of SA in least number of function evaluations. This is due to the existence of single optima which is desirable to RHC to rise over the randomized point in the search space. Here also, if the number of iterations and maximum attempts are increased, a significant difference between the performance of RHC might be seen making the choice of GA disputable given its exponential runtime. For futher analysis, as specified in N-Queens problem, various decay techniques can be compared with higher random states.

As seen from the comparative loss curve of Neural network, RHC spontaneously optimized weights as the iterations are increased while outperforming other three strategies with the highest accuracy, evident from Type-I and Type-II errors. Given the performance of the four algorithms, there exists a specific set of weights (single global optima) which influences the loss metric. As the number of iterations increases, GA's losses decreases however, SA is least affected by it. Similarly, the Gradient descent strategy maintains its performance over the iterations. Hence in future works, if time is not a critical factor, the four algorithms can be tested for a higher number of iterations with various random states and comparatively analysed. Also, the schedule of SA can be tested for Arithmetic Decay and Exponential Decay as well as various learning rates, number of neurons and activation functions can be compared.

References

[1] Fedesoriano, Company Bankruptcy Prediction, https://www.kaggle.com/datasets/fedesoriano/company-bankruptcy-prediction.

[2] Genevieve Hayes, mlrose-hiive, https://github.com/hiive/mlrose.