Assignment 2: Randomized Optimization

Naziul Talukder
ntalukder6@gatech.edu

1 INTRODUCTION

In this paper I will analyze random optimization algorithms like Randomized Hill Climb, Genetic Algorithm, Simulated Annealing and MIMIC. I will employ these algorithms to various problems to highlight their strengths and weaknesses. In assignment 1, I used the breast cancer dataset from UCI. I trained a NN model on that dataset using back propagation and gradient descend. With the same network architecture, I will compare the result of this supervised learning implementation and usage of randomized optimization algorithms like GA, SA and RHC. The dataset will undergo similar pre-processing step as assignment 1. I will analyze training time, minimization of loss, performance on accuracy score to obtain the best learner for this dataset.

2 OPTIMIZATION

Randomized Hill Climb (RHC) generates a random initial solution and iteratively takes small steps towards possibly better solution. **Simulated Annealing (SA)** is a stochastic optimization algorithm that probabilistically accepts candidate solutions using a similar idea as the RHC to ensure a balance of exploration and exploitation. The idea of exploration means, the algorithm will accept candidate solution even if they are worse than current solution, in the hopes for better solution in future iterations. In earlier iterations, the algorithm explores more and in the later iterations the algorithm only accepts better solutions. **Genetic Algorithm (GA)** is a metaheuristic optimization algorithm inspired by the process of natural selection. It generates new solutions using crossover and mutation to evolve into a better solution. **MIMIC** uses similar idea but it uses a probabilistic model to generate new candidate solutions as well as using crossover like GA.

I optimized the Continuous Peaks, Flip Flop and Four Peaks problems using various algorithms like RHC, SA, GA and MIMIC. The Continuous Peaks problem was chosen to highlight GA, Flip Flop to highlight MIMIC and Four Peaks to highlight SA.

2.1 Continuous Peaks

Continuous Peaks optimization problem relates to binary strings that maximizes a certain fitness function. The fitness function considers long sequences of 0 and 1. The fitness function for this problem can be described by

$$Fitness(x, T) = max(maxrun(0, x), maxrun(1, x)) + R(x, t)$$

where $maxrun(b, x)$ is the length of the maximum run of b in x.

$R(x, t) = n$ if $max(maxrun(0, x) > T$ and $max(maxrun(1, x) > T$; otherwise $R(x, t) = 0$. Here T is the threshold parameter, $T = t_{pct} * n$ where t_{pct} is expressed as a percentage of the state space dimension.

Figure 1 shows how changing the problem size impacts the fitness score in each algorithm. I used $t_{pct} = 0.1$ for all the algorithms. Maximum number of iterations for each algorithm was 1000. Maximum number of attempts to find a better neighbor at each step was 100. This was chosen because increasing this results in a high computation time for MIMIC and I wanted to compare similar hyperparameter for all the algorithms. MIMIC and GA used similar population size 500 (it was chosen because it provides high fitness score both for various range of problem size and iterations). Figure 1 shows that GA with the highest fitness score outperforms all other algorithms. MIMIC employs a similar method as GA. GA performs best for problems with constraints that need to be satisfied, large search space, many local optima, discrete variables. Figure shows that SA performs as good as GA for smaller problem size (simpler problems) outperforming MIMIC. But MIMIC and GA tend to outperform SA as the problem grows to be more complex.

Figure 2 shows the fitness curve of each algorithm when the problem length is 100 and number of iterations is varied. As expected, GA converges to the fitness score for fewer iterations than SA and RHC. SA with higher number of iterations provides similar fitness score as GA outperforming MIMIC and RHC. The right side of the graph shows number of function evaluations for number of maximum attempts for each algorithm. The number of maximum attempts define maximum number of attempts to find a better neighbor/state at each step. I constrained the number of maximum attempts and maximum iterations for MIMIC to be much lower than others due to high computation time. GA and MIMIC tend to have similar rate of function evaluations which are significantly higher than the

amount for SA and RHC. SA requires the least amount of function evaluations as shown in the figure. Though the number of function evaluations is similar for GA and MIMIC, if the algorithm is constrained by number of iterations, MIMIC would be a better option. Because MIMIC shows higher fitness score for same amount of iterations compared to any of the other algorithms. MIMIC converges to a fitness score using fewer iterations, because of its use of dependency trees and probability distribution. Figure 6 first row shows time required to generate the fitness curve observed in figure 2. GA and MIMIC converges with fewer iterations, but they still take much longer time compared to SA due to high amount of function evaluations.

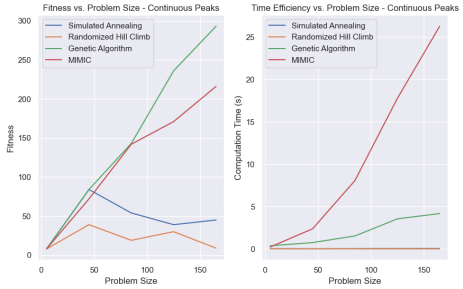

Figure 1—Continuous Peaks Fitness Score and Time Efficiency

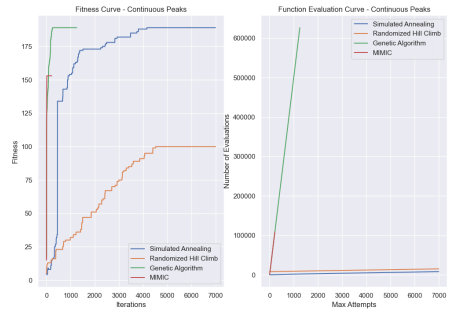

Figure 2—Continuous Peaks Fitness Curve and Function Evaluations

2.2 Flip Flop

Flip Flop problem counts the number of times bits alternation occurs. It is a combinatorial optimization problem where the fitness value can be considered the number of flips needed to be made in the string to make all adjacent bits the same. MIMIC demonstrates the best fitness score in figure 3. With more

complex problem (longer bit strings) RHC tends to perform poorly compared to others. All other algorithms tend to provide similar fitness score. For 100 length bit string (possible values 0 and 1) figure 4 shows the fitness curve of each algorithm as well as the number of function evaluations required. MIMIC requires the highest computation time and GA the second highest. Random Hill Climb and SA requires similar computation time. RHC tends to perform the worst and MIMIC obtains high fitness score for fewest iterations as observed in the previous problem. Max attempt here indicates maximum number of attempts of each algorithm to find a better neighbor/state at each step during the iteration. Similar to previous implementation, maximum number of iterations and attempts were constrained to be lower for MIMIC due to its high computation time. The right side of the figure shows that GA and MIMIC requires similar amount of function evaluations for a period, and the number keeps rising for GA with higher iterations. But even with higher iterations and attempts GA is still unable to surpass MIMIC's performance. With higher and higher iterations, SA shows similar fitness score as MIMIC but as the problem size increases MIMIC's graph is always higher than SA's in figure 3. The second row in figure 6 shows the computation time for each algorithm for the fitness curve in figure 4.

Figure 4 shows the strength of MIMIC using a probabilistic model that estimates the distribution of promising solutions and generate candidate solutions accordingly. It maintains a diverse population of promising solutions and uses the structure to guide the search process for a better solution. Exploring new solution from an estimated distribution of promising solutions allows MIMIC to explore the search space better compared to local search algorithms like Random Hill Climbing. MIMIC uses probabilistic model to estimate the dependencies which makes it robust against invalid solutions. Hence, we observe higher fitness score for MIMIC compared to RHC. Even though MIMIC is computationally expensive, it is the best option for Flip Flop problem with high complexity (higher problem size).

2.3 Four Peaks

This problem is similar to the continuous peaks problem we discussed above. The fitness function is based on two scenarios: one for the consecutive 1's in the string and another for consecutive 0's.

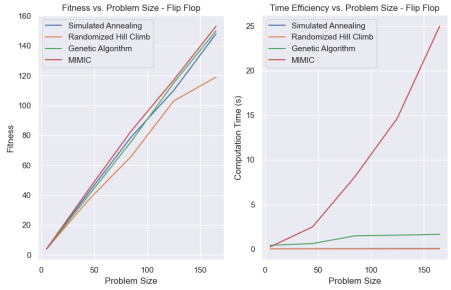

Figure 3—Flip Flop Fitness Score and Time Efficiency

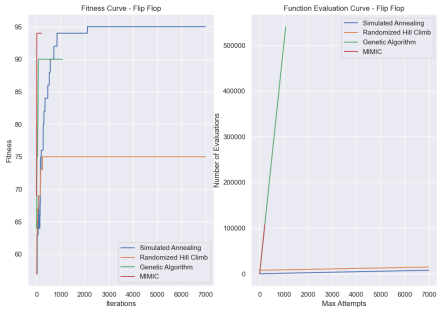

Figure 4—Flip Flop Fitness Curve and Function Evaluations

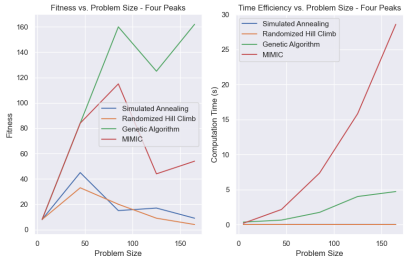

Figure 5—Four Peaks Fitness Score and Time Efficiency

	SA	RHC	GA	MIMIC
ContinuousPeaks	0.170991	0.115730	18.246399	20.012213
FlipFlop	0.205928	0.149905	18.391524	20.350468
FourPeaks	0.070703	0.047028	14.988357	20.213890

Figure 6—Time in seconds for each algorithm for each problem

The fitness function can be described by:

$$Fitness(x, T) = max(tail(0, x), head(1, x)) + R(x, t)$$

where $tail(0, x)$ is the number of trailing 0's in x and $head(1, x)$ is the number of leading 1's in x

$R(x, t) = n$ if $tail(0, x) > T$ and $head(1, x)) > T$; otherwise $R(x, t) = 0$. Here T is the threshold parameter, $T = t_{pct} * n$ where t_{pct} is expressed as a percentage of the state space dimension.

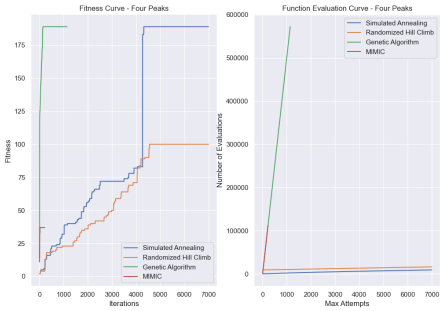

Figure 7—Four Peaks Fitness Curve and Function Evaluations

Figure 5 shows how changing the problem size impacts the fitness score in each algorithm. I used $t_{pct} = 0.1$ for all the algorithms. Maximum number of iterations for each algorithm was 1000. Maximum number of attempts to find a better neighbor at each step was 100. MIMIC and GA used similar population size 500. It shows that GA outperforms all the other algorithms and the difference increases as the complexity increases. For simpler version of the problem (with smaller length of the string) MIMIC and GA tends to have similar fitness score. But for more complex problems, GA outperforms MIMIC. The caveat is that our simulation restricts maximum iterations to be 1000 in this case. Figure 7 shows fitness curve of each algorithm when problem size = 100 is considered. As algorithms are no longer constrained by number of iterations: a higher fitness score for SA and RHC is observed. RHC never outperforms GA but SA outperforms GA when it can utilize more than 5000 iterations. The right side of the graph shows that when maximum attempts (maximum number of attempts to find a better neighbor/state at each step) is varied SA requires less function evaluations calls than GA or MIMIC. Figure 6 shows that the SA curve in figure 7 required less computation time compared with MIMIC or GA. Even though SA is a local

optimization model like RHC, it outperforms RHC because it can explore new possible solutions better than RHC as well as accepting better solutions in each step. This allows SA to avoid getting stuck in the local optima. SA requires a lot more iterations than GA or MIMIC. But it still requires less function evaluations to converge. Its computational complexity is simpler and it trains faster as shown in figure 6. In every problem, it requires less time to converge compared to MIMIC or GA.

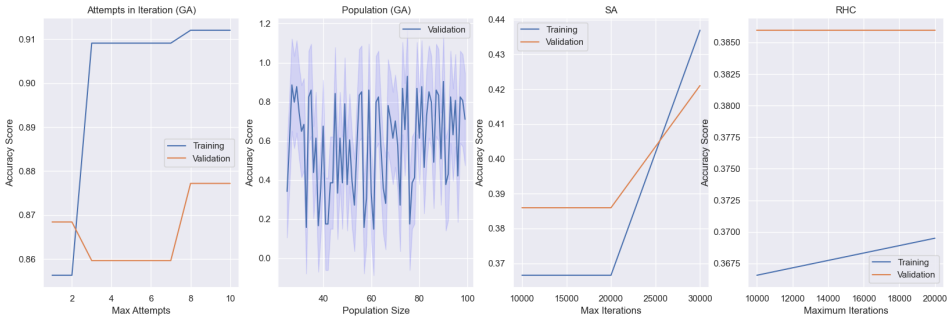

Figure 8—Hyperparameter tuning

3 NN

I obtained the breast cancer dataset from UCI database to perform a classification task. The features in the dataset describe characteristics of the cell nuclei present in the digitized image of a fine needle aspirate (FNA) of a breast mass. The dataset contains diagnosis: malignant or benign. The dataset contains 569 samples with 30 features describing the characteristics of the cell i.e. radius, texture, concavity, compactness, perimeter, symmetry etc. There are 212 malignant and 357 benign samples. In assignment 1, I trained a NN using gradient descend and back propagation(Backprop) to predict if a cell is malignant or benign by looking at the characteristics of the cell.

In this section I will compare Backprop with randomized optimization algorithms like GA, SA and RHC. The architecture of the NN will be consistent: using only one hidden layer. I tuned RHC, SA and GA to obtain the best model using each of the algorithms as seen in figure 8. For GA the highest performance was obtained varying max attempts at each iteration population size. As observed in the previous section, GA converges to its fitness score for few iterations. I observed convergence for 20 iterations and the accuracy did not improve changing it. I observed the best accuracy using 10 as maximum attempt and 75 as

population size. SA and RHC requires a lot more iterations than GA. Changing the number of maximum iterations impacted the accuracy score. Changing the number of maximum attempts did not impact the score. With higher iterations they both tend to show improved performance. SA required more computational resources than RHC due to the underlying mechanism of exploring and exploiting. Due to computational consideration 60,000 iterations were considered as maximum for SA. Changing decay and learning rate did not improve the performance further. Compared to SA, RHC is simply just stepping towards improved solutions. Because RHC iterations are lighter than SA, 100,000 maximum iterations was employed for RHC because with increasing iterations the accuracy did not improve much. Changing number of restarts and maximum clips did not improve the performance further. I updated the learning rate for the back propagation implementation to 0.000001 for optimized performance.

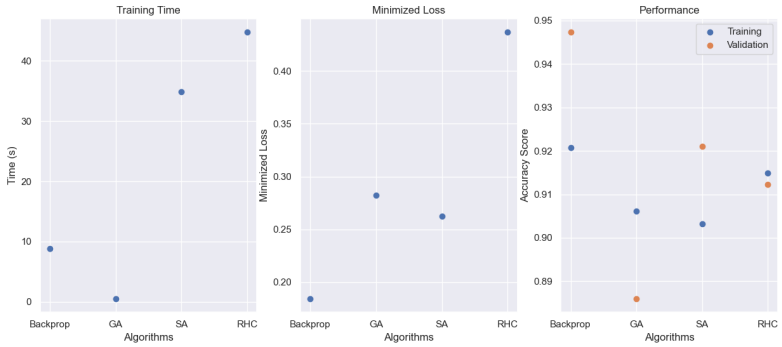

Figure 9—Performance on training and validation set

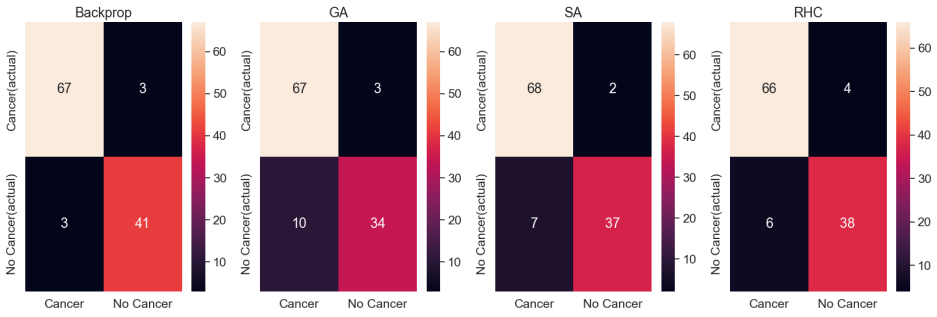

Figure 10—Confusion Matrix on predictions

Figure 9 shows the performance of each algorithm on the cancer dataset I used in previous assignment. Figure 9 shows that Genetic Algorithm trains the model

	Test Accuracy
Backprop	0.964912
GA	0.938596
SA	0.947368
RHC	0.929825

Figure 11—Accuracy Score on test dataset

fastest. Comparing the best minimized loss values for each algorithm: back propagation outperforms all of them. The performance of back propagation yields the highest validation score. For Backprop and SA validation score is higher than training score. It implies the training dataset used for those algorithms does not represent the validation set properly. The training sets are harder to learn compared with the validation sets. Figure 10 shows the confusion matrix for each algorithm. In terms of least misclassification: Backprop outperforms all others. RHC takes the longest to train yielding worst minimized loss values. SA takes a long time to train too because of enormous amount of iterations required to converge. Backprop might be the best option among all the algorithms. But if we are considering training time to be a huge factor: GA might be a good alternative as it takes $1/9$ time of backprop to train the data with comparable minimized loss value and accuracy score.

Due to the high performance of back propagation we can infer that our dataset is best suited for gradient based optimization. It is updating the weights in the right direction based on the Gradient of the loss function compared with other ideas like crossover of solutions, exploring and exploiting, stepping towards better solution. Genetic algorithm has inherent bias that assumes crossover of good solutions will provide better solutions. The solutions might be biased towards certain area of the entire search space. The idea that known good solutions can be exploited is not as effective as gradient descend of loss functions. The decreasing temperature for better solutions in SA is also not as effective as Gradient descend. During tuning, changing the annealing schedule did not affect its performance. Due to computational resources I only considered maximum $60,000$ iterations. Due to its computational complexity this algorithm was taking much longer compared to GA or Backprop. RHC was computationally inexpensive compared to SA as it did not have to consider exploring like SA does. But even with much higher iterations $100,000$ RHC still could not outperform Backprop. With more and more random restart did not change the performance of the algorithm. The search space was too complicated for RHC converge quickly. RHC was the most

expensive algorithm in terms of training time.

Figure 11 shows the accuracy score on test dataset and as expected Backprop outperforms all other algorithms. In assignment 1, we observed linear separability in the dataset. There are some non-linear relationship in the features of the dataset. The non-linear relationships were not prominent enough to use other activation functions. Hence, the identity function performed best compared with **tanh, relu, sigmoid**. Similarly, Backprop outperformed all unsupervised models because with help of gradient descend it was able to step the right amount to the right extent to minimize the loss function. The bias associated with GA, SA and RHC hindered those models to perform as good as Backprop. Because of the linear relationship of the features in the dataset, it was more suited for Gradient Descend. The dataset consisted of a lot of local minima. GA, SA And RHC were prone to getting stuck in those local minima. Backprop allowed the weight to be adjusted in the right direction with use of gradient descend avoiding the local minima.

4 CONCLUSION

Various optimization problems were considered to analyze random optimization algorithms. I chose **Continuous Peaks** to highlight **GA**, **Flip Flop** to highlight **MIMIC** and **Four Peaks** to highlight **SA**. I also discussed the main idea, bias, strength and weakness of each algorithm discussing those problems. I compared the **Back Propagation** with Gradient Descend algorithm with **GA, RHC, SA** algorithms on the breast cancer dataset. I observed that Backprop outperforms all the algorithms considering the minimized loss, performance on validation dataset, test dataset and confusion matrix of the prediction. GA provides performance comparable with Backprop with improved training time: $\frac{1}{9}$ training time of Backprop.