Randomized Optimization

Chase Brooks

dbrooks43@gatech.edu

***Abstract*—** Four Randomized Optimization Algorithms (Random Hill Climbing, Genetic Algorithm, Simulated Annealing, and MIMIC) are used to solve 3 toy problems which highlights the strengths and weaknesses of each respective algorithm. After this, I also use these algorithms to tune a neural network's weights and measure performance of email spam classification.

1 ANALYSIS METHODOLOGY

To explore the efficacy of various Randomized Optimization Algorithms, I will use the MLRose Python package. This package comes with classes that implement Random Hill Climbing, Genetic Algorithm, Simulated Annealing, and MIMIC as well as various commonly analyzed optimization problems. I will rely mainly on the default parameters defined in the MLRose documentation. To ensure convergence, I will set the maximum number of iterations at an arbitrarily high number (1000) while also enabling early stopping. This way, I will minimize the chances of prematurely cutting off a model before it converges, but also end training when successive iterations yield little improvement in model fitness.

2 TOY PROBLEM 1: FOUR PEAKS

2.1 Problem Overview

This problem consists of an N-dimensional bitstring consisting of 1's and 0's that has 4 peaks (2 global optima and 2 local optima). A peak is defined as the point in which a series of bits changes from 1 to 0 or from 0 to 1. As demonstrated in subsequent sections, this problem becomes difficult as the problem size increases due to the ever increasing "basin of attraction" that can catch algorithms in local optima (De Bonet et al, 1997).

2.2 Algorithm Performance

Figure 1 below shows both the relative performance and the number of function evaluations required to maximize the Four Peaks problem for each respective Random Optimization Algorithm. As expected, the Random Hill Climbing Algorithm has both the lowest performance, and is among the lowest number of function evaluations to maximize. This indicates that the Random Hill Climbing Algorithm is quickly converging on a local optimum and is unable to adequately explore the hypothesis space. Simulated Annealing and MIMIC perform relatively similarly in terms of fitness; they both do relatively well until the problem space reaches ~60 bits, after which there is a precipitous decline in fitness. As the basins of attraction between global

optima increase, it appears that simulated annealing and MIMIC have difficulty converging on the global optimum. Interestingly, the Simulated Annealing Algorithm takes significantly more function evaluations to converge on its maximal evaluation as compared to the other 3 algorithms. That being said, each function evaluation in the MIMIC algorithm is computationally expensive, so it requires the most time for convergence of any algorithm. The Genetic Algorithm clearly has the highest performance of all algorithms. Its continued increases in fitness as problem size increases shows it is the best algorithm at traversing large basins of attraction. Even though the number of function evaluations for the Genetic Algorithm is relatively low, each evaluation is computationally expensive, resulting in the second longest computation time of all algorithms measured.

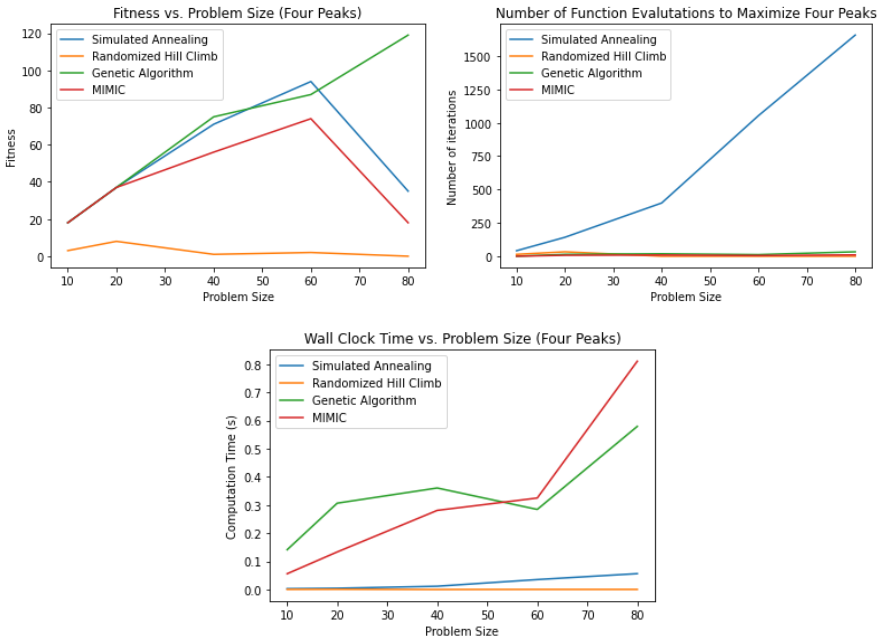

***Figure 1**: Performance of Randomized Optimization Algorithms on the Four Peaks Problem*

3 TOY PROBLEM 2: Knapsack

3.1 Problem Overview

The knapsack problem is a hallmark example problem for mathematical observation where one has a knapsack of fixed weight capacity, and n different objects, each with its own weight and value. We are tasked with finding the optimal combination of items that maximize total value in the knapsack while remaining less than or equal to the total carrying capacity of the knapsack. Due to the large number of possible combinations of weights, algorithms that excel in exploration such as the genetic algorithm should perform well on this problem. Additionally, MIMIC's ability to incorporate knowledge of structures will likely prove useful as well.

3.2 Algorithm Performance

Random Hill Climbing is the worst performer in terms of overall fitness. While this algorithm converges quickly both in terms of number of iterations and overall computation time, it clearly results in suboptimal combinations of items in the knapsack problem. Again, Simulated Annealing and MIMIC perform similarly in terms of overall fitness, but Simulated Annealing requires significantly more function evaluations to converge on an optimal combination. This is somewhat negated by the fact that each iteration of the MIMIC algorithm is significantly more computationally expensive than Simulated Annealing, so the latter does have a lower wall clock time to converge. The genetic algorithm again returns the highest fitness score with a relatively low number of function evaluations. However, the wall clock time for the genetic algorithm is nearly twice as long as the second longest algorithm, MIMIC. In larger problem spaces, this chasm will likely continue, so tradeoffs for overall fitness and computational complexity may need to be considered in higher dimension problem spaces.

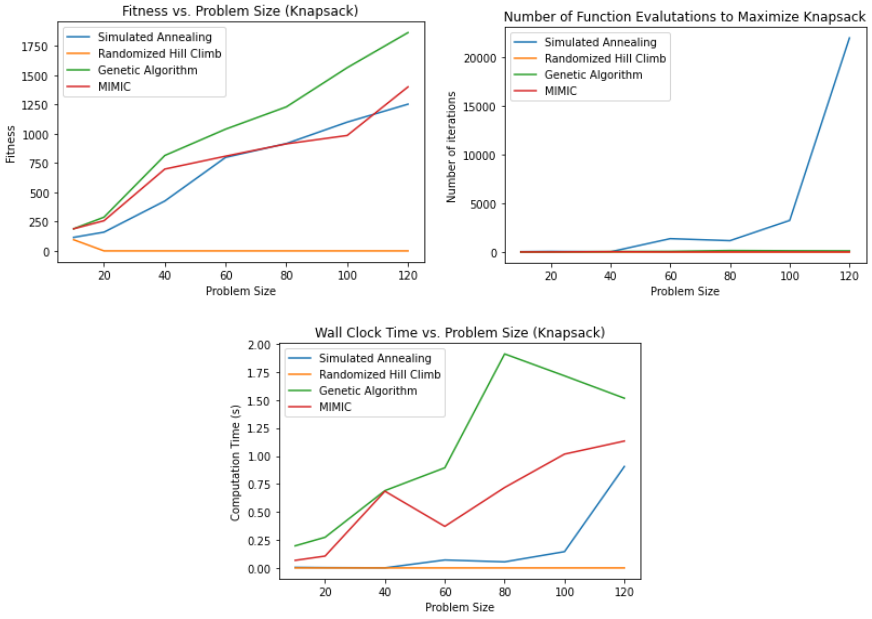

***Figure 2**: Performance of Randomized Optimization Algorithms on the Knapsack Problem*

4 TOY PROBLEM 3: OneMax

4.1 Problem Overview

The OneMax problem is a simple optimization problem whereby we are tasked with maximizing the sum of all bits in a N-dimensional bit string. We know the optimal solution should always be simply the string of all 1's, but our Randomized Optimization algorithms do not have this information at the outset of the problem. This problem highlights situations in which Random Hill Climbing algorithms can achieve similar performance to other algorithms, but with a much lower wall clock time.

4.2 Algorithm Performance

Figure 3 shows that the Genetic Algorithm and MIMIC are the top performers in terms of overall fitness and the number of function evaluations to return the global optimum, but the computational complexity of these algorithms results in wall clock times that are orders of magnitude larger than the other two algorithms Simulated Annealing and Random Hill Climbing result in only slightly lower fitness and a higher number of iterations to converge on the global maximum value. However, each of these evaluations is much simpler than MIMIC or the genetic algorithm, which results in low wall clock time for each. This performance behavior in the OneMax problems highlights a class of problems that, while the Genetic Algorithm and MIMIC, do slightly outperform other algorithms, this increased performance comes at a steep computational cost. As the search space in these trivial problems grows, so too will the wall clock time of these more complex algorithms. In certain situations, end users may simply need an algorithm that gets close to the global optimum as quickly as possible. Under this scenario, simple algorithms such as Random Hill Climbing truly shine.

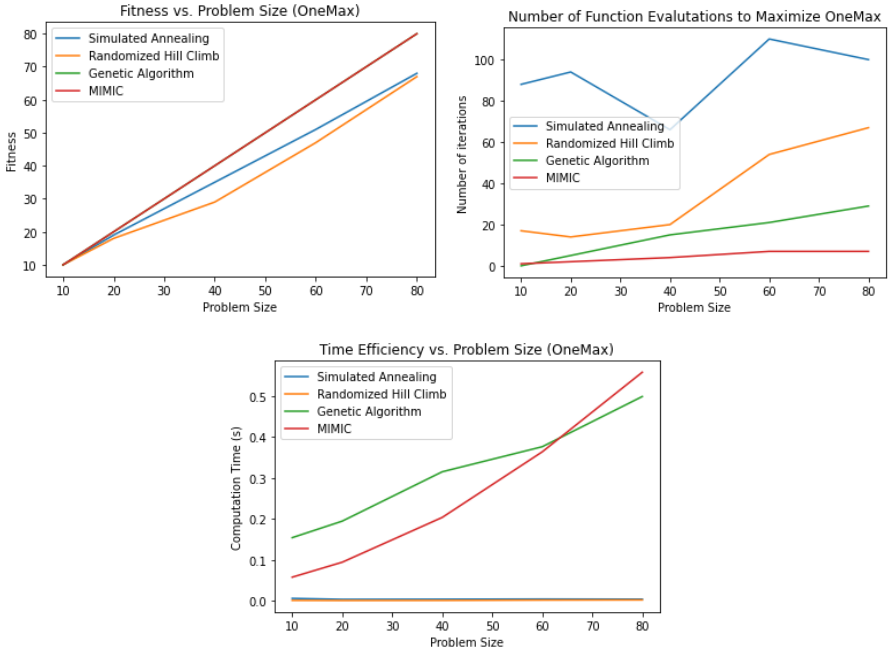

***Figure 3**: Performance of Randomized Optimization Algorithms on the OneMax Problem*

5 NEURAL NETWORK OPTIMIZATION

5.1 Problem Overview

The email spam dataset is a collection of 4.6k emails classified as either legitimate or spam. The data consists of various work frequencies commonly used in spam emails such as “free”

and “money” as well as indicators of emails likely to be legitimate, such as coming from a .edu email address.
Link to dataset: https://www.openml.org/d/44

5.2 Random Hill Climbing

I first used Random Hill Climbing to tune the Neural Network. This resulted in relatively poor performance across both the test and train set (Figure 4). I relied on the default parameters provided by MLRose except for increasing the maximum number of iterations to 1000.

Training

	precision	recall	f1-score	support
0	0.80	0.42	0.55	2228
1	0.48	0.84	0.61	1452
accuracy			0.58	3680
macro avg	0.64	0.63	0.58	3680
weighted avg	0.67	0.58	0.57	3680

Testing

	precision	recall	f1-score	support
0	0.81	0.47	0.60	560
1	0.50	0.83	0.63	361
accuracy			0.61	921
macro avg	0.66	0.65	0.61	921
weighted avg	0.69	0.61	0.61	921

***Figure 4**: Neural Network Performance with Random Hill Climbing Weight Tuning in Train/Test Set*

5.3 Simulated Annealing

Next, I used Simulated Annealing to tune the weights of the neural network. Figure 5 shows a marginal improvement over the Random Hill Climbing Algorithm across all metrics. Simulated Annealing also appears to underfit the data and performance can likely be improved by further tuning additional hyperparameters such as the number of hidden nodes and learning rate.

Training

	precision	recall	f1-score	support
0	0.79	0.61	0.68	2228
1	0.55	0.75	0.64	1452
accuracy			0.66	3680
macro avg	0.67	0.68	0.66	3680
weighted avg	0.69	0.66	0.67	3680

Testing

	precision	recall	f1-score	support
0	0.80	0.62	0.70	560
1	0.56	0.76	0.65	361
accuracy			0.68	921
macro avg	0.68	0.69	0.67	921
weighted avg	0.71	0.68	0.68	921

***Figure 5**: Neural Network Performance with Simulated Annealing Weight Tuning in Train/Test Set*

5.4 Genetic Algorithm

Finally, I used the Genetic Algorithm to tune the weights of the neural network. This algorithm yielded around an 80% F-Score in both the test and train set, which is clearly the highest performing algorithm of the 3 tested in this analysis. The behavior was expected since the Genetic Algorithm is able to combine/mutate existing weights with decent performance, which allows it to more efficiently explore the most promising regions of the hypothesis space.

Training

	precision	recall	f1-score	support
0	0.89	0.81	0.85	2228
1	0.75	0.85	0.80	1452
accuracy			0.83	3680
macro avg	0.82	0.83	0.82	3680
weighted avg	0.84	0.83	0.83	3680

Testing

	precision	recall	f1-score	support
0	0.88	0.79	0.83	560
1	0.72	0.83	0.77	361
accuracy			0.80	921
macro avg	0.80	0.81	0.80	921
weighted avg	0.81	0.80	0.81	921

***Figure 6**: Neural Network Performance with Genetic Algorithm Weight Tuning in Train/Test Set*

6 ALGORITHM COMPARISON

This analysis has clearly shown that there is no single algorithm that outperforms all others across all problem spaces. Simpler, instance based algorithms such as Random Hill Climbing and Simulated Annealing often result in lower computational complexity and faster execution with lower performance on average. On the other hand, distribution based algorithms such as the Genetic Algorithm and MIMIC often result in stronger performance but with significantly more complex computations and longer wall clock time. This tradeoff of model efficacy and computational time is very important when applying these algorithms to the real world. Certain problem spaces are very complex and require knowledge of structures to maximize performance. In these situations, MIMIC and Genetic Algorithms shine. In other examples, individual computations required are relatively straightforward, but we need to quickly conduct a brute force search to find the maximum value. In these situations, Random Hill Climbing and Simulated Annealing are the best choice.

7 SOURCES

1. De Bonet, J., Isbell, C., & Viola, P. (1997). *Mimic: Finding Optima by Estimating Probability Densities*. Georgia Institute of Technology. Retrieved March 5, 2022, from https://faculty.cc.gatech.edu/~isbell/papers/isbell-mimic-nips-1997.pdf
2. MLRose Documentation. (n.d.). Retrieved March 5, 2022, from https://mlrose.readthedocs.io/