Michael Tong
CS 7641 – Spring 2019

Assignment 2

ID: 903357308
mtong31@gatech.edu

Abstract: The following discussion analyzes the affect that different optimization approaches have on various problems. Random hill climbing, simulated annealing, genetic algorithms, and MIMIC are all explored. First, a MLPC network is reviewed using alternative methods to the backpropagation approach from a previous assignment, attempting to classify a student's gender through seemingly unrelated metrics. Next, we explore three problems where each hypothesis space is optimized by three different optimization procedures, demonstrating the no free lunch theorem.

Weight Optimization

From our previous analysis performing classification, specifically the multilayer perceptron neural network on the student test performance dataset, we focus on the crucial process of hidden unit weight updates.[1] This classification task remains the same, where a student's gender is attempted to be classified through features such as test scores and personal characteristics. Previously, we found that a two-layer deep network with 12 hidden units provided fairly good results with ~87% testing accuracy. Our goal will be to investigate if improvements can be made through altering the weight update method, where backpropagation was previously used. This is performed through the ABAGIL jython package developed by pushkar, with referenced code through mitian223.[2,3] The alternate optimization algorithms investigated will be random hill climbing, (RHC) simulated annealing (SA), and genetic algorithms (GA).

Kicking the analysis off, we'll run ABAGAIL's backpropagation (BP) algorithm with an identical network to ensure that it produces similar results to benchmark against the scikit-learn MLPC class.[4] This is captured by first performing 10 instances of backpropagation with a maximum number of iterations of 10,000 and examining the instance with the lowest mean squared test error. Luckily, as shown in Figure 1 below, a near identical testing accuracy and characteristic train-test curve is found between the ABAGAIL BP and scikit-learn algorithms. With this confirmation, analysis of the other weight optimization approaches can be performed. A summary of their results is displayed in Table 1.

Algorithm	MSE	MSE Iter	MSE Time (s)	Acc	Acc Iter	Acc Time (s)
BP:	0.0429	42	0.119	0.868	74	0.195
RHC:	0.0382	8024	3.907	0.910	3996	3.840
SA:	0.0311	3930	4.100	0.932	3660	3.832
GA:	0.0328	5750	274.0	0.928	5330	254.2

Table 1: Global Results

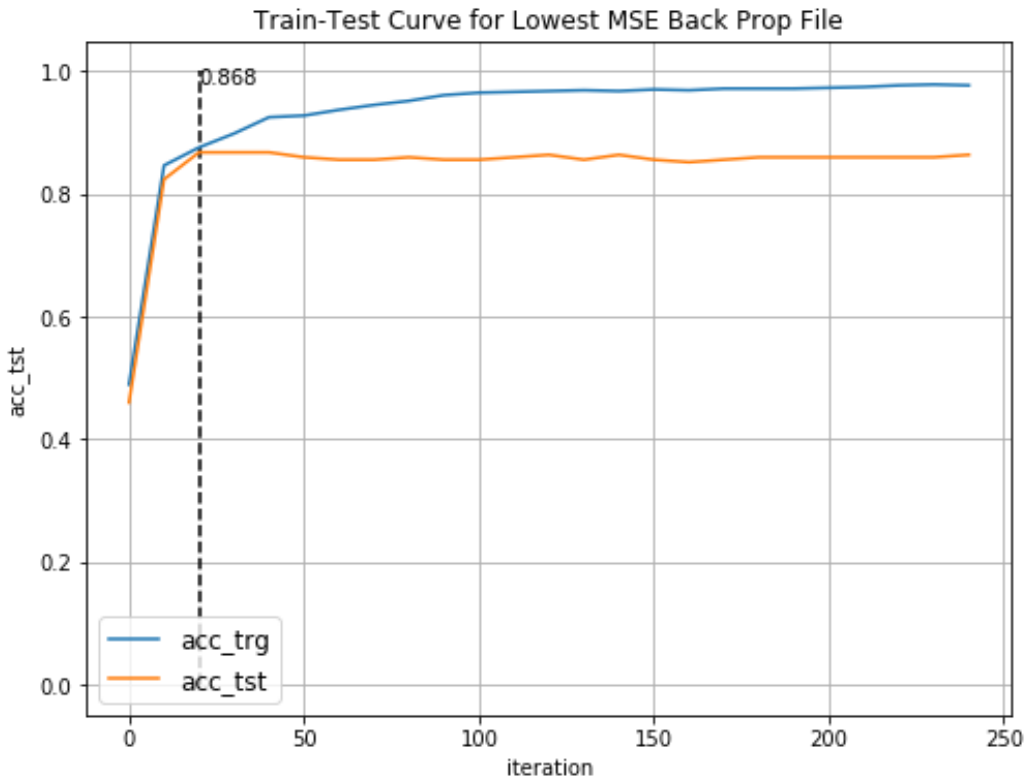

Figure 1: Train test curve for backpropagation by ABAGAIL

To clarify the results, metrics for BP and RHC is taken as an average of 10 runs, while SA and GA are recorded as their lowest MSE run, where various runs are taken with differing hyperparameters, which will be discussed in detail later. Both SA and GA were allowed up to 10,000 iterations. With our BP benchmark, we find that it performs astonishingly well in terms of convergence time, only requiring an average of 42 iterations and 0.119 seconds to complete, with a comparable test MSE and accuracy. In many situations, the almost 35x increase in computational speed may justify the 6.9% reduction in testing accuracy by the SA algorithm.

Random Hill Climb

Beginning with an analysis of the RHC algorithm, Table 1 shows that its testing metrics are middle of the pack but has comparable iteration requirements and time. Figure 2 below displays RHC metric results as an average of 10 runs.

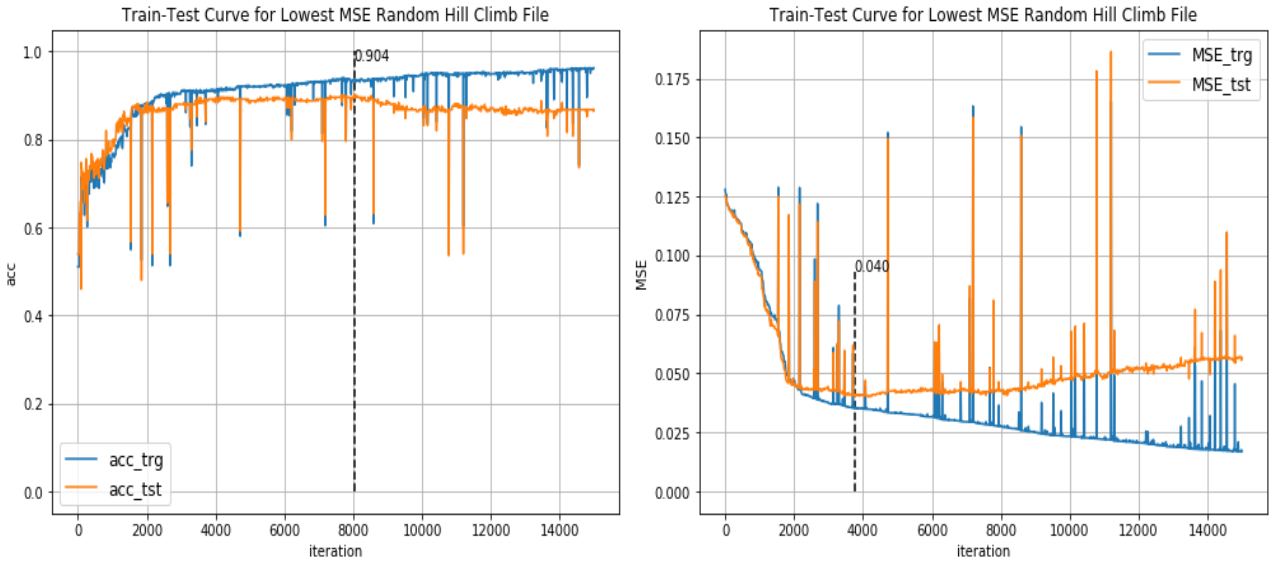

Figure 2: Test Accuracy vs Iteration (Left); MSE vs Iterations (Right)

In comparison with the BP curves, they're definitely messier, which is likely the result of the process inherently involving exploration. What is expected and nicely shown however is diminishing performance of the model as the number of iterations increases, likely the result of overfitting.

Simulated Annealing

Moving from the relatively simple RHC method, we now shift our analysis to the SA approach. The two hyperparameters available for tuning are the temperature and cooling rate. A plot of their MSE errors are shown in Table 2 below.

T / C	0.2	0.4	0.6	0.8	0.9	0.99	Average
1.00E+01	0.0397	0.0311	0.0358	0.0384	0.0385	0.0491	0.0388
1.00E+03	0.0391	0.0454	0.0369	0.0442	0.0381	0.0440	0.0413
1.00E+05	0.0346	0.0385	0.0400	0.0376	0.0355	0.0457	0.0387
1.00E+07	0.0419	0.0399	0.0336	0.0392	0.0505	0.0383	0.0406
1.00E+09	0.0358	0.0440	0.0364	0.0417	0.0453	0.0474	0.0418
1.00E+11	0.4471	0.0391	0.0414	0.0411	0.0344	0.0500	0.1088
1.00E+13	0.0502	0.0353	0.0402	0.0484	0.0351	0.0528	0.0437
Average	0.0983	0.0390	0.0378	0.0415	0.0396	0.0468	

Table 2: MSE for Temperature vs Cooling Rates

As a not super scientific approach, the averages across each cooling rate per temperature and vice-versa are calculated. The lowest MSE on average across all cooing rates appears at T=1e5, which we investigate further with a plot shown below.

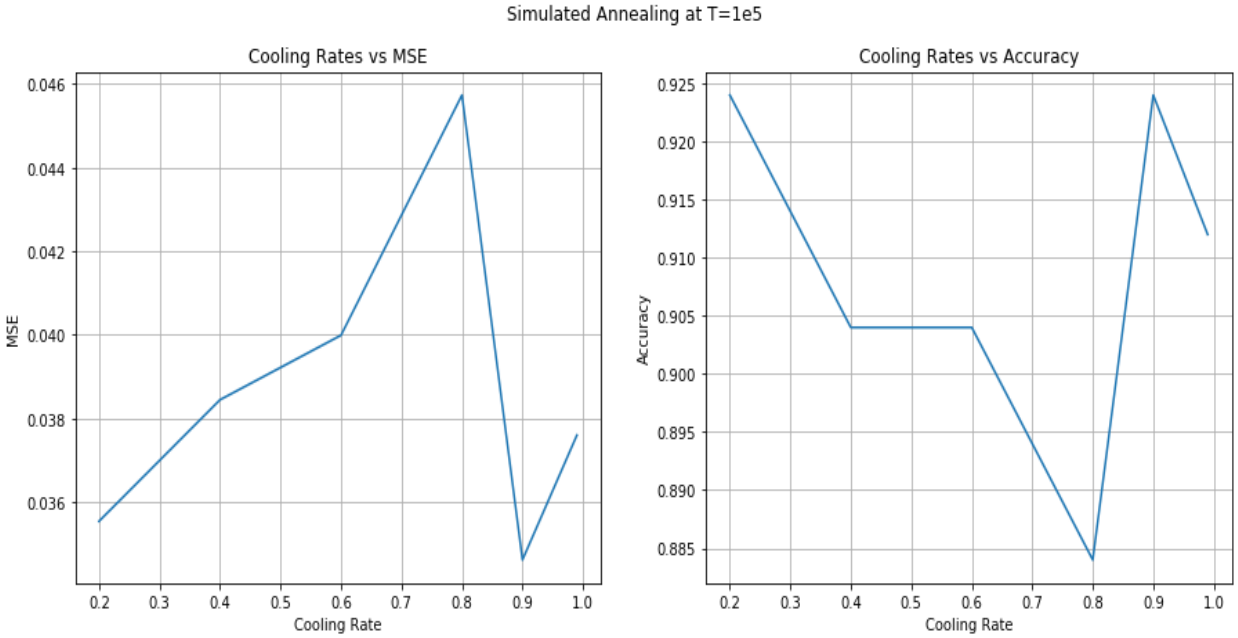

Figure 3: Simulated Annealing Cooling Rate Analysis at T=1e9

Figure 3 displays some interesting trends, where performance appears to decrease with cooling rate (higher the slower the cooling), until 0.9, where it actually exhibits a minima. It's unclear why this occurs but what is occurring mathematically is that the likelihood is exploration is decreasing more rapidly, and at 0.9 for these particular runs, it happens to not explore a region where it falls into a poor local minimum. The spread of performance is slightly concerning since it performance can vary greatly by incremental changes to hyperparameters.

Genetic Algorithms

The last approach to weight optimization explored is the genetic algorithm. From Table 1, we can clearly see that it exhibits the highest convergence time to reach both its lowest MSE and highest accuracy by a significant margin, while still being second to SA, let's explore in detail! Hyperparameters combinations for population size, the population to mate, and the population to mutate were tested with P = 100, 200, 300, mate = 20, 30, 40, and mutate = 20, 30, 40. Both the lowest MSE and highest accuracy are found at a population of 100, mating population of 40, and mutation population of 20. Figure 4 below displays the associated accuracies and MSE over the iterations.

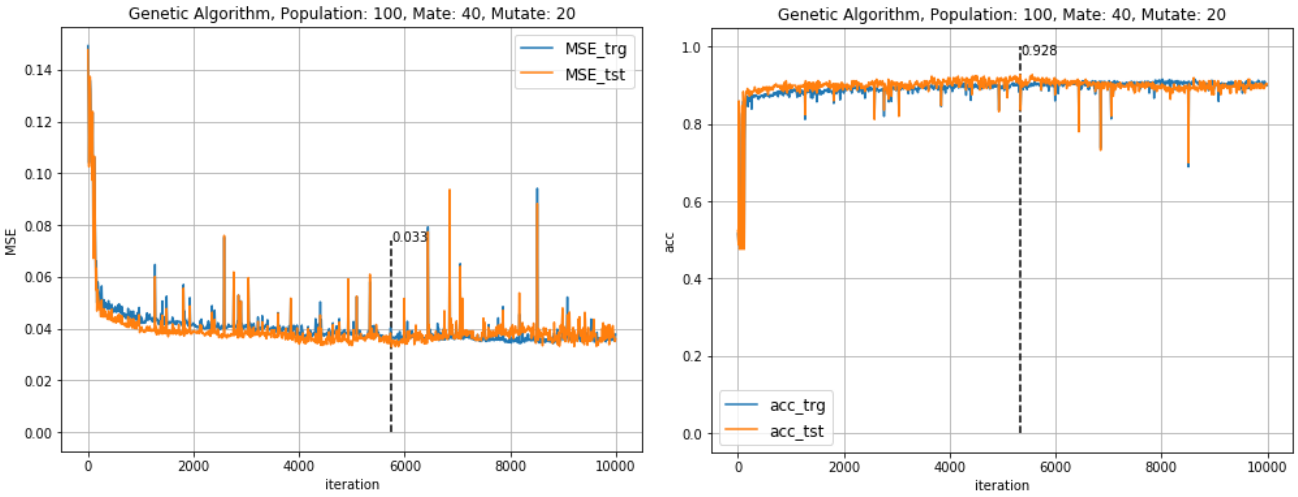

Figure 4: Genetic Algorithm metric results over iterations

In comparison with the RHC algorithm, these curves do not demonstrate similar levels of overfitting. Overall, the time complexity alone is enough evidence to avoid this method for this particular use case, and the addition of a lower test accuracy further diminishes its value.

Summary

Overall, we find that our original backpropagation approach to weight optimization performs rather phenomenally in terms computational efficacy but does suffer slightly in model prediction accuracy. Simulated annealing appeared to be the preferred approach for this particular problem, increasing the testing accuracy by ~7%.

The (Giant) Knapsack - MIMIC

In addition to a weight optimization problem, analysis now shifts to analyze a fitness optimization problem, specifically the knapsack problem. Inspiration for this analysis was captured through Analytics Vidhya and datamining apps.[5,6] The knapsack problem involves maximizing a utility metric with various items, each having a penalty term, globally limited by total weight. The primary metric for analyzing this problem is the model's fitness results, or how well the knapsack can be filled within the constraints. For our analysis, the number of items is limited to 100, each with a max weight and volume of 60, and a total volume of 4200. Highest resulting fitness measures for various hyperparameters is displayed in Table 3 below.

Algorithm	Highest Fitness	@ Iteration	Time (s)
RHC (10 runs)	5337	1185	0.00375
SA	6581	5090	0.0047
GA	10130	41430	5.5371
MIMIC	10433	30320	506.4743

Table 3: Knapsack Optimization results

Figure 5: Knapsack fitness over iterations for lowest MSE parameters

Compared with our weight optimization analysis, we see a similar trend in the time complexity of each algorithm, but with the introduction of MIMIC, an even higher candidate than GA. In contrast however, SA is no longer the optimal option for this problem, and both GA and MIMIC perform significantly better, demonstrating the importance of algorithm specification. Again, there is the tradeoff for computational efficacy vs model convergence, where the RHC and SA achieve a fitness score of about 50% worse than the MIMIC algorithm but computes the results magnitudes faster. Here, we find that the MIMIC algorithm performs best, likely due to the shape of the hypothesis space, where features happen to form stronger dependency trees with eachother.

Traveling Sales Person – GA

The next problem explored is the traveling salesperson, where we attempt to determine the optimal path for an agent to travel to various cities with a cost function (distance) associated with each path, and not returning to any city more than once. This is a fairly classical problem for exploring very large hypothesis spaces. For our analysis, we explore a map with 200 cities, and randomly assigned coordinates on a 2D plane. Results for each algorithm is displayed in Table 4 below.

Algorithm	Highest Fitness	@ Iteration	Time (s)
RHC (10 runs)	0.0667	9700	0.0329
SA	0.0707	9850	0.0135
GA	0.1150	1340	0.2987
MIMIC	0.0249	450	82.398

Table 4: Travelling Sales Person Results

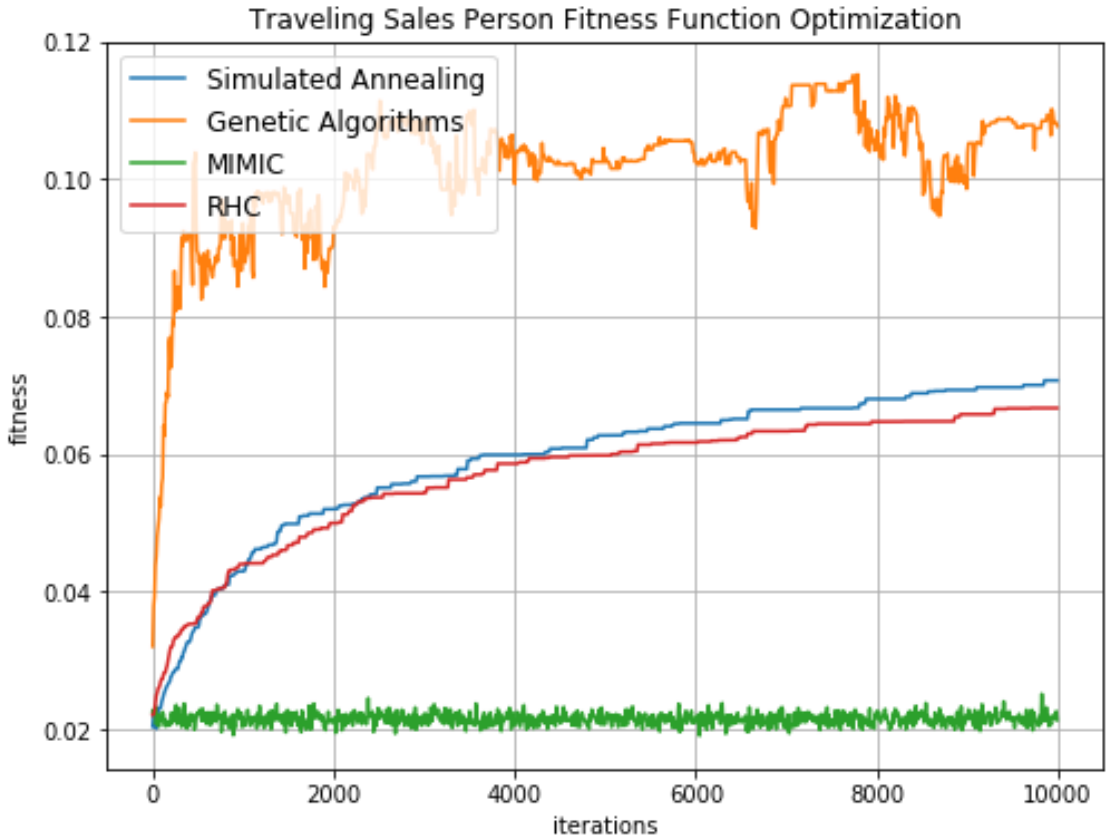

Figure 6: Travelling Sales Person fitness over iterations for lowest MSE

For the plot, SA, GA and MIMIC hyper parameters are identically explored to before, and the RHC is again performed 10 times. The lowest MSE parameters are plotted and displayed in Figure 6 above. Early termination is used, resulting in the GA and MIMIC algorithms not running to 10,000 iterations.

As expected, we find that the MIMIC algorithm performs poorly, likely due to poor generation of the algorithm's dependency trees. We find that the GA algorithm performs the best, but what is interesting is how much better it is in comparison, both in convergence time and accuracy. With the mate and mutate process, it's aptly named how this algorithm performs best for a human process. The alternative paths can be synonymous to various DNA patterns, where a survival of the fittest mentality can be translated into paths that offer the most return (lowest cost) and choosing between two paths of high return will likely still remain a good path. Mutation can be thought of as taken a potential hidden path that could be a shortcut.

K Colors – SA

The last problem explored is the K Colors problem, where the goal is to determine the minimum number of colors assigned to nodes on a unidirectional graph such that each node has a different color from all its neighbors. For this analysis, a size 500 graph is created with 2 colors trying to be fit. Results are displayed in Table 5 below and Figure 7 below displays fitness over iterations for the best parameters.

Algorithm	Highest Fitness	@ Iteration	Time (s)
RHC (10 runs)	408	3530	0.0339
SA	444	4060	0.0094
GA	376	4670	2.4675
MIMIC	394	790	129.62

Table 5: 2 Colors optimization results

It's pretty clear that SA performs the best on the 2-color problem, both in terms of fitness and computational efficacy. Similar to the travelling sales person, this problem has a lot of alternative paths to select from in the hypothesis space, so it's no surprise at this point that MIMIC performs poorly. However, in contrast with that problem, it's interesting to see that GA performs the worst, while for the previous problem it performed the best. The reason for this may be a result of each path not having an associated weight; the overall fitness of a path is determined once the entire grid is filled. It's unclear what about the grid makes it optimal, so selection of ideal mates for the GA algorithm is close to uniform. This unweighted graph may allow SA to excels since it is allowed to randomly explore the hypothesis space.

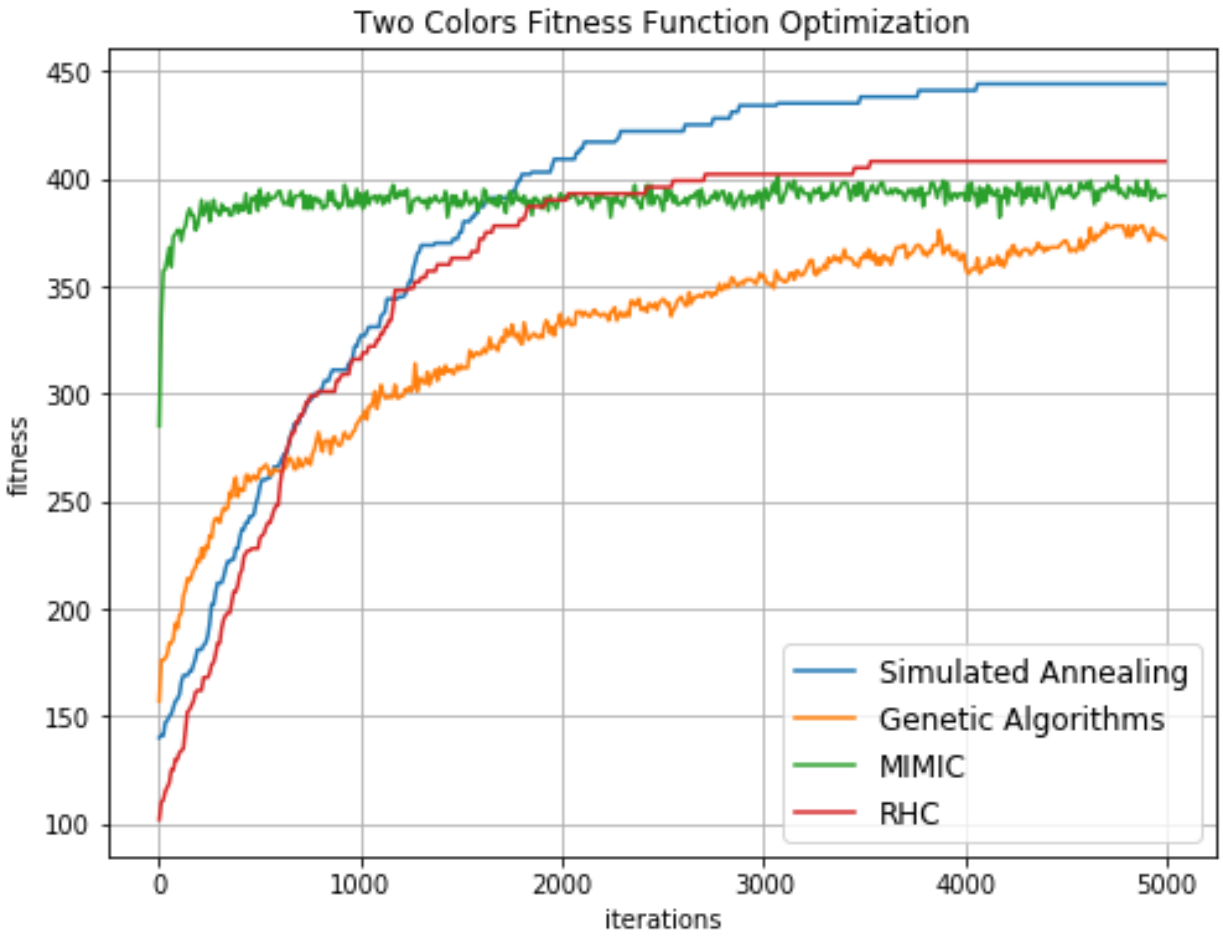

Figure 7: Two Colors fitness over iterations for best parameters

Summary

Through analyzing the three optimization problems, we find that there is indeed no free lunch. Each algorithm performed differently for each problem and each problem had an algorithm of the four options that produced a higher fitness score than the rest, often by a non-trivial amount. Generally speaking however, computational efficacy was a common trend throughout each problem, where MIMIC was the slowest to its lowest MSE solution, often by magnitudes higher. SA in contrast was the fastest of the four. Speaking of SA, in all three problems, it exhibits similar fitness performance to the RHC algorithm, which is expected since SA is essentially RHC with exploration and will eventually become RHC as the temperature reaches 0. What is also interesting is how the three highest fitness algorithms for one problem performs the poorest in another, SA performs the worst in the knapsack problem, MIMIC performs the worst for the travelling sales person problem, and GA performs the worst for the K colors problem. This further emphasizes the importance of domain knowledge in understanding the hypothesis space to optimally select a algorithm.

References

[1] Tong, Michael. "Assignment 1 – Classification". 11 February 2019. https://github.com/miketong08/CS7641/blob/master/assignments/assignment_1-Classification/assets/report/mtong31-analysis.pdf
[2] Kolhe, Pushkar. "Pushkar/ABAGAIL." GitHub, 8 Mar. 2018, github.com/pushkar/ABAGAIL/tree/master/src
[3] mitian223. "mitian223/CS7641." GitHub, 19 Aug. 2018, github.com/mitian223/CS7641/tree/master/HW2
[4] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011
[5] Jain, Shubham. "Introduction to Genetic Algorithm & Their Application in Data Science." Analytics Vidhya, 31 July 2017, www.analyticsvidhya.com/blog/2017/07/introduction-to-genetic-algorithm/
[6] Stripling, Eugen, et al. "Solving the Knapsack Problem with a Simple Genetic Algorithm." DataMiningApps, 12 Mar. 2017, www.dataminingapps.com/2017/03/solving-the-knapsack-problem-with-a-simple-genetic-algorithm/.