Napa Tan Vananupong

Dr. Brian Hrolenok

3 March 2019

Machine Learning: CS4641

Randomized Optimization

Introduction:

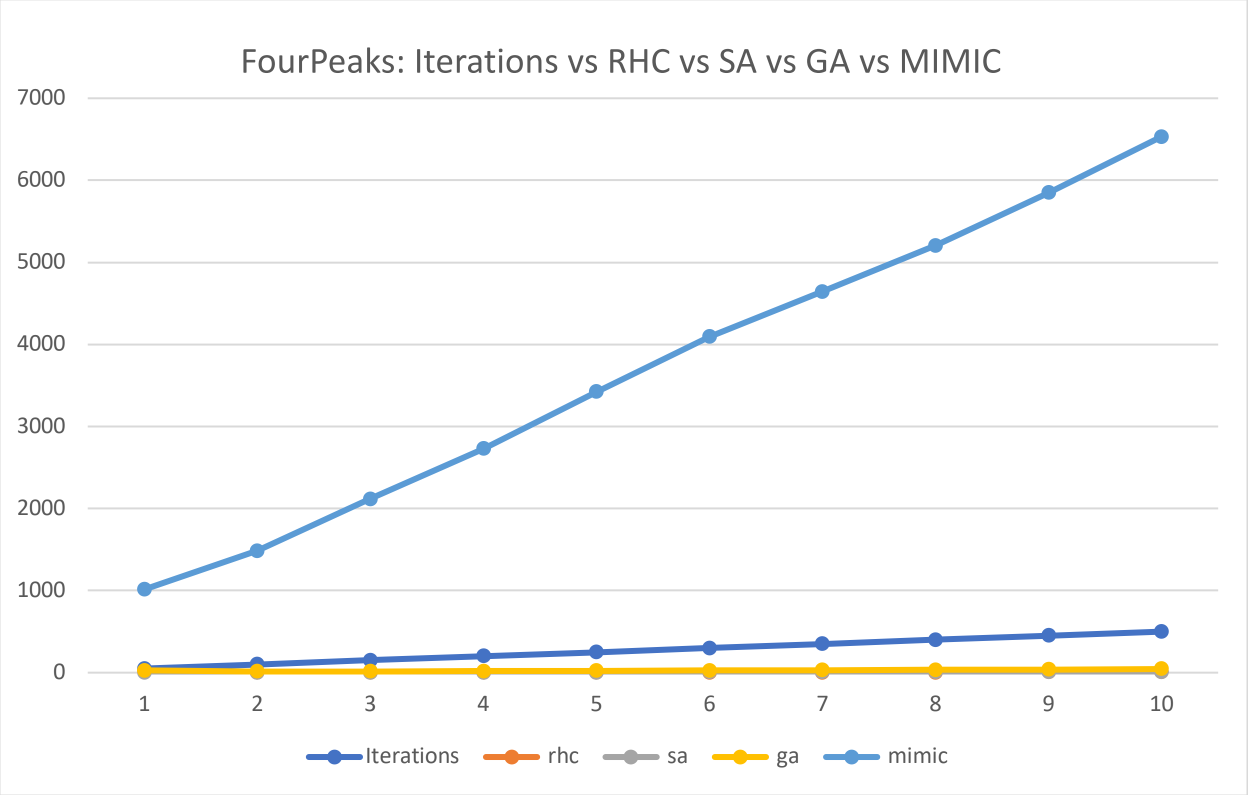
There are many interesting problems you could approach with randomized optimization, as there are many methods you could choose to implement, each with their own set of unique circumstances in which they perform best in. In this assignment, we explored four randomized optimization algorithms and their performance, being randomized hill climbing, some genetic algorithm, simulated annealing, and MIMIC. Each selected problem was chosen to outline the strengths and weaknesses of each Randomized Optimization algorithm. In the second part of the assignment, we apply all four search techniques to three unique and interesting optimization problems we chose, which for me, were: Max K Coloring, the Traveling Salesman problem, and the Four Peaks problem. In addition to all the above, I also applied the selected three algorithms to learning weights of a neural network.

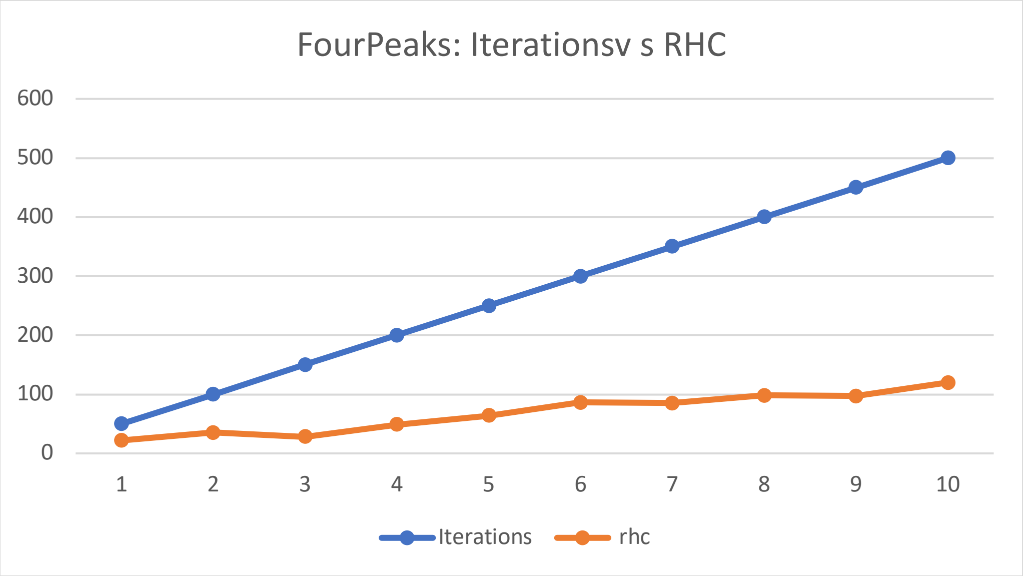
Chosen Problems:

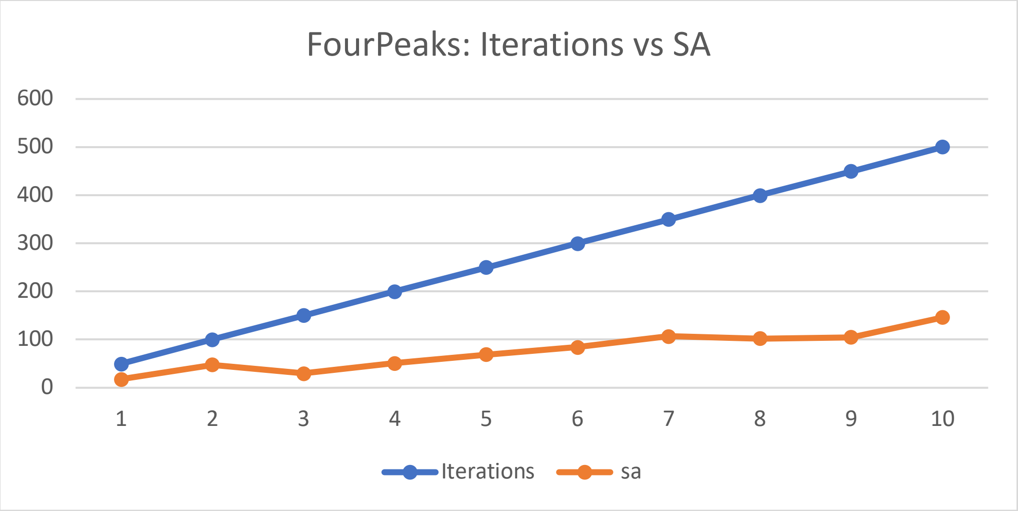
Each of the three selected problems were chosen to highlight the strengths and weaknesses of each algorithm.

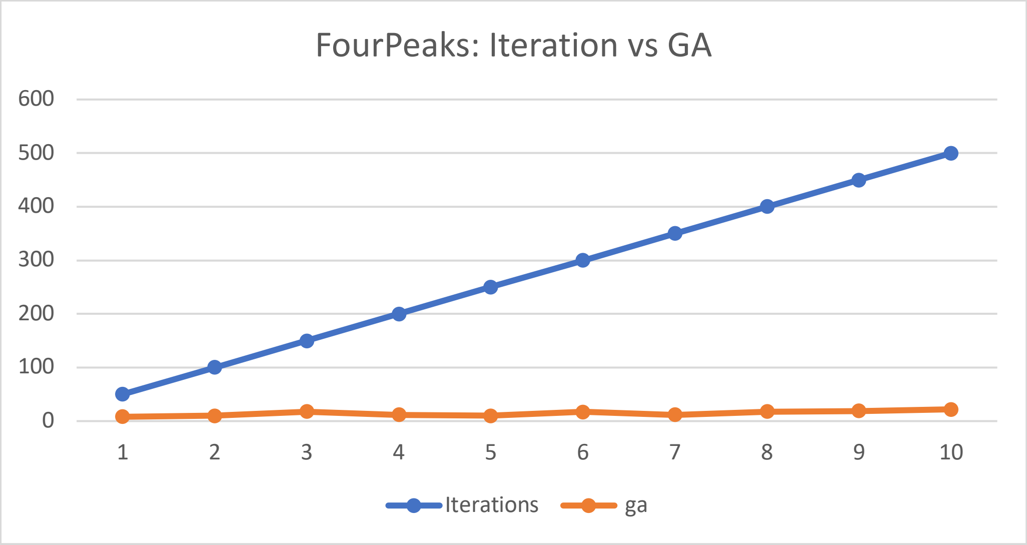
Fourpeaks:

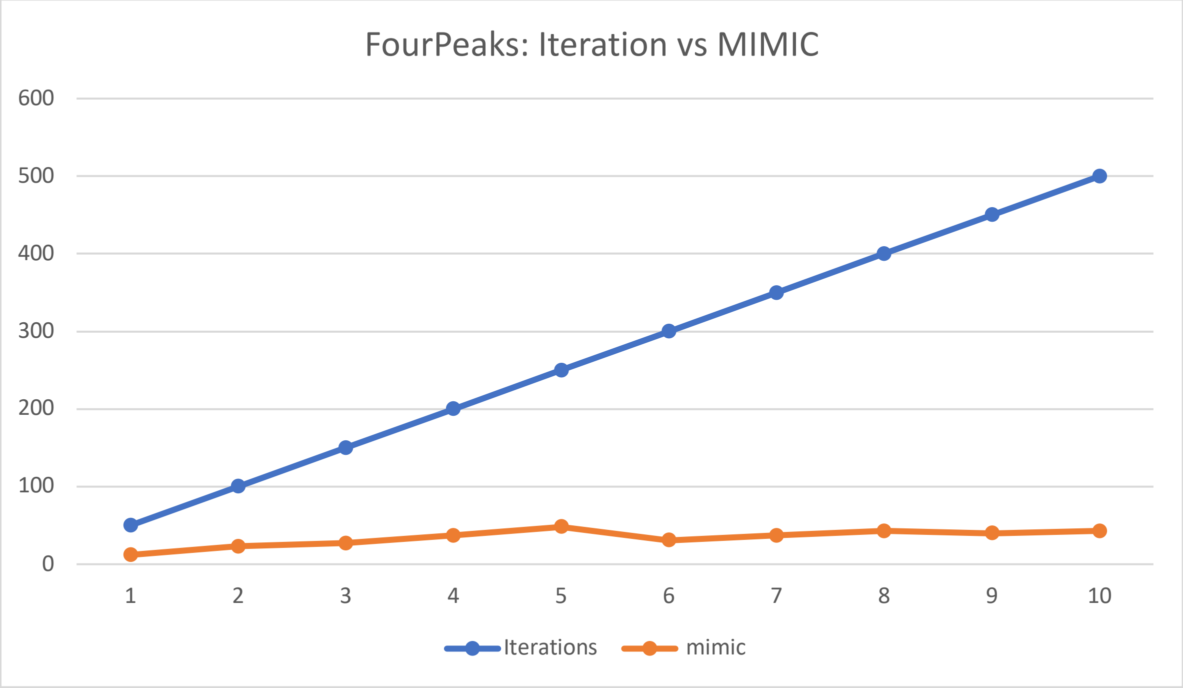
I chose the fourpeaks problem to show the strengths and weaknesses of Genetic Algorithms and MIMIC. The fourpeaks problem (Baluja and Caruana, 1995) is: given an N-dimensional input vector X, the four peaks evaluation function is defined as I(X, T) =:: max [tail(O, X), head(l, X)] + R(X, T), where tail/(b, X) = :: number of trailing b's in X head(b, X) = :: number of leading b's in X, and R(X T) = {N if tail(0, X) > T and head(l, X) > T , 0 otherwise. This function has two additional global maxima where there are T + 1 leading 0's followed by all 1 's or when there are T + 1 trailing 1 's preceded by all 0's. Results on running GA and MIMIC and SA and RHC vs their number of iterations is shown below. In all MIMIC trials, T was set to be 10% of N, which was the total number of inputs. This combination of N and T resulted in the graph below. The MIMIC algorithm clearly consistently maximizes the fourpeaks function with 1/10 of the number of evaluations. Meanwhile, it is clear that Genetic Algorithms excel more than the other two (RHC and SA) when the function computes quickly, and when the T+1 trailing starts with 1's and ends with 0's, or the opposite, that is, when the T+1 trailing starts with just 0's and ends with just 1's. Clearly, the best algorithm for the FourPeaks problem is the MIMIC.




FourPeaks: Iterationsv s RHC
600
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
Iterations
rhc

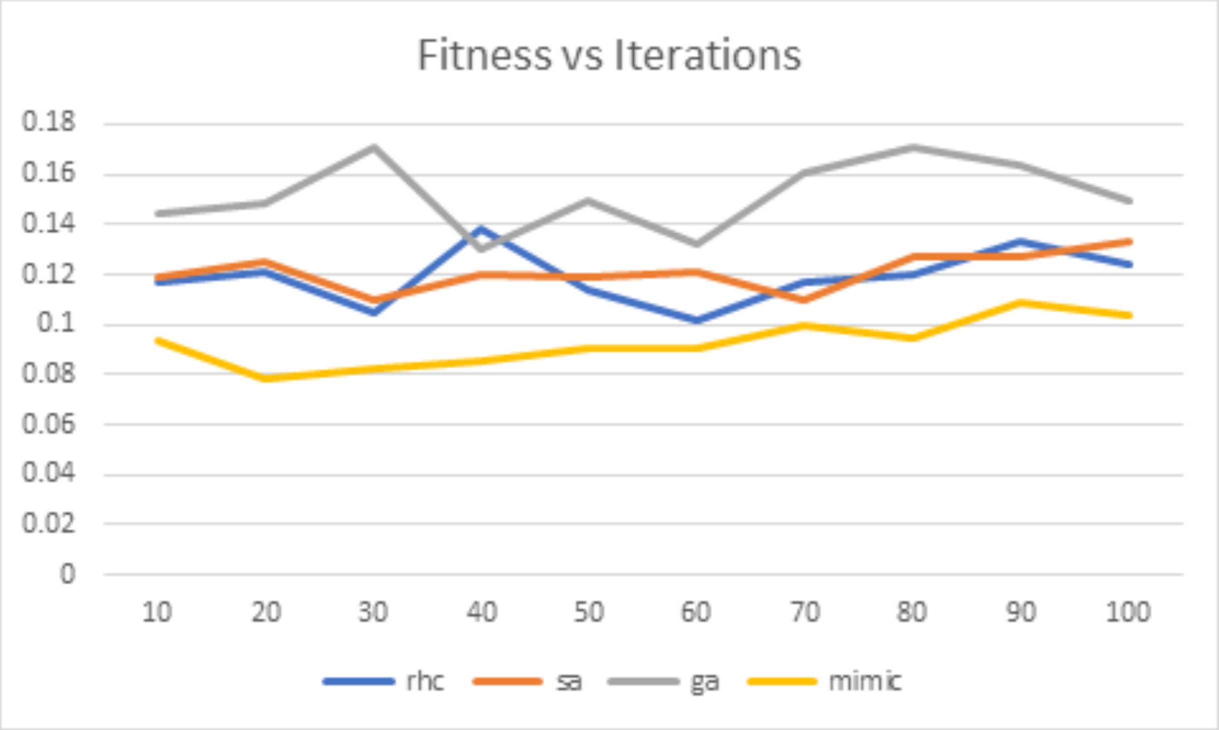

FourPeaks: Iterations vs SA
600
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
Iterations
sa





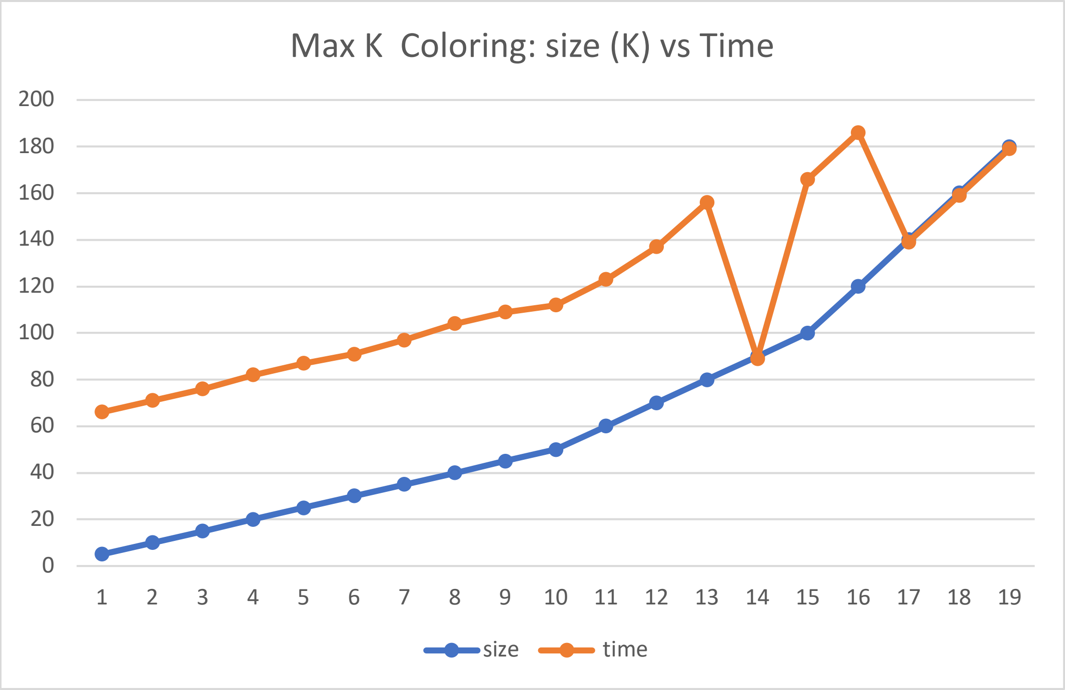
Traveling Salesman:

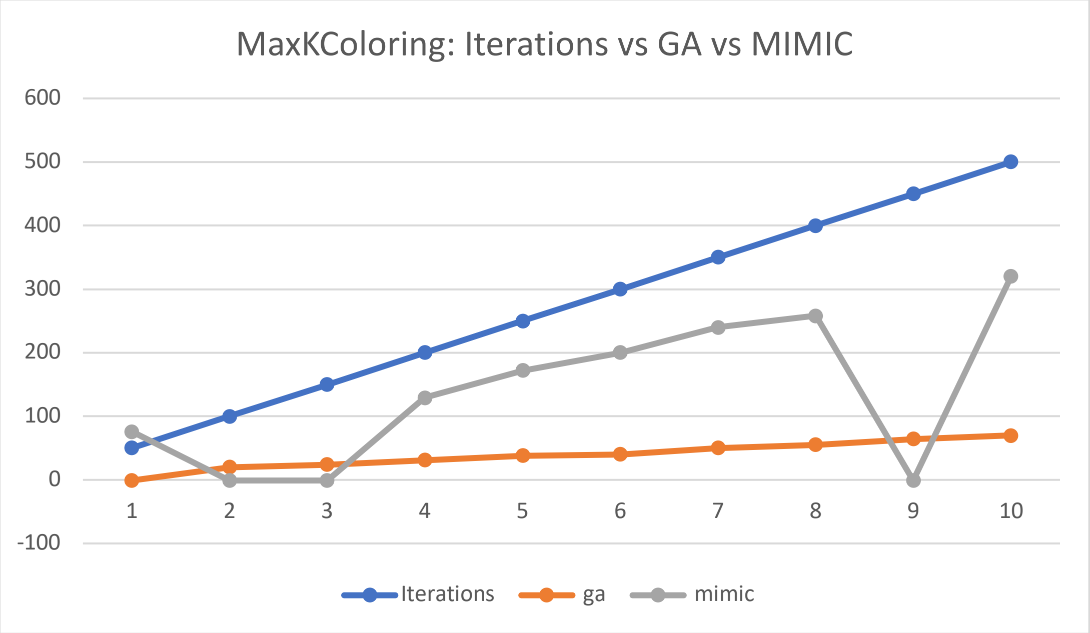
The traveling salesman problem is a very well known and well explored problem in both Mathematics and Computer Science, thus it seems perfect for Machine Learning. The problem is as follows: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city? I felt that this problem was interesting because it has been one of the most intensively studied problems in optimization and is often a benchmark for many new optimization methods, and it is still an NP-Hard problem, thus finding a polynomial solution would be extremely useful. I felt that it would display the strengths and weaknesses of each algorithm well because we can and will compare the performances of each randomized algorithm on this problem by comparing the number of iterations for each algorithm to the optimal solution. In this case, I have define the optimal solution as the minimal distance travelled, or defined as function: max(1/(distance travelled)). By examining the resulting graphs (_) below, it can be seen that as the number of iterations increases, both randomized hill climbing and simulated annealing's optimal solution improves. Interestingly, the genetic algorithm does better than the rest on all numbers of iterations. When we plot the optimal solution against the number of cities, we can notice that the optimal solution decreases as the number of cities increases, as expected. This is because the travelling salesmen problem will become exponentially harder as the number of cities grows larger, since we must find more shortest paths between all the cities, and since this is a well-known NP hard problem.



Max K Coloring:

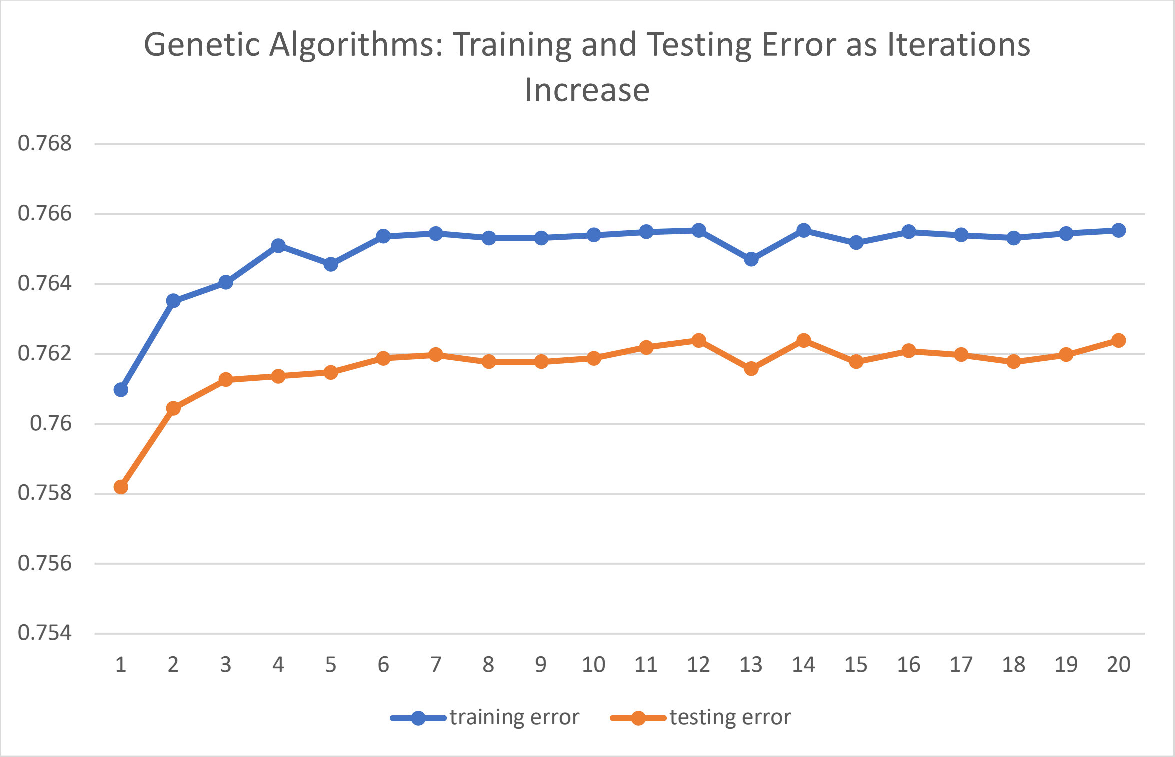
I chose Max K Coloring an arbitrary graph because it is one of the fundamental problems in graph theory and in computer science and combinatorics. The problem is as follows: Color the vertices of a graph so that the vertices to which they are connected by edges will have different colors, with the biggest number of different colors possible. The Max K Coloring problem has many different applications and solving this problem or attempting to analyze it with our four algorithms will hopefully provide very useful results. For this problem, I varied the number of iterations of ABIGAIL's max K algorithm and plotted the resulting chart of size (K) vs Time down below (the evaluation time) as well as the number of Iterations compared to the Genetic Algorithm's results and the MIMIC results, The other 2 algorithms were not fit for this problem, therefore no charts nor results were included for them.

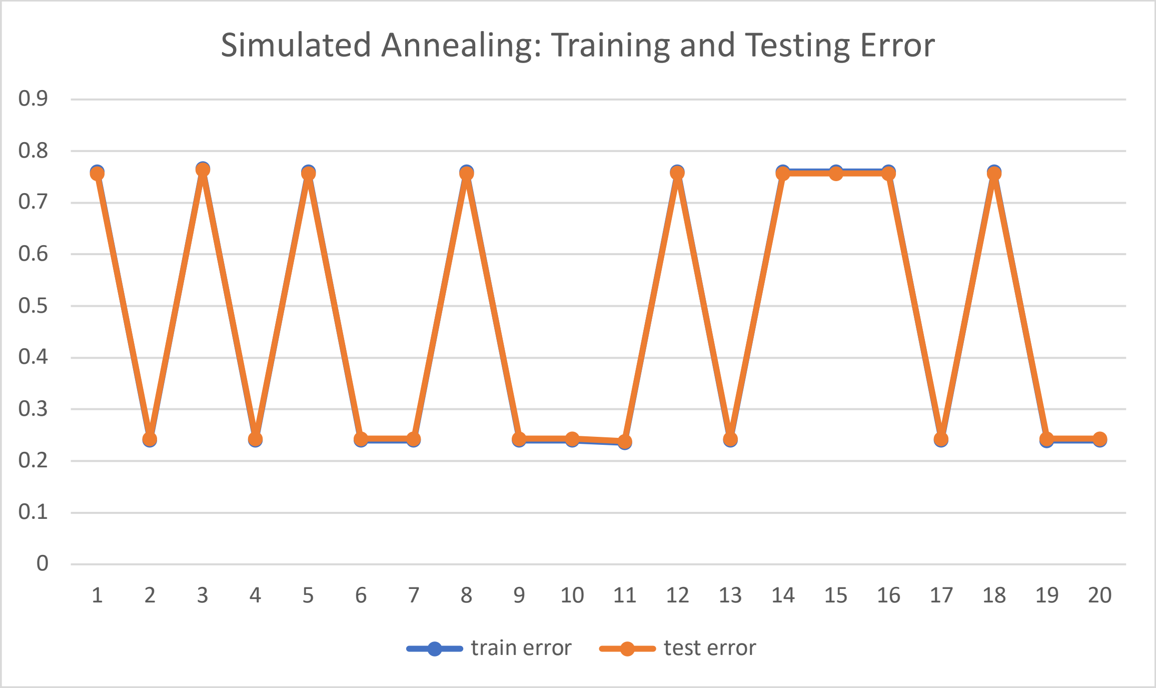


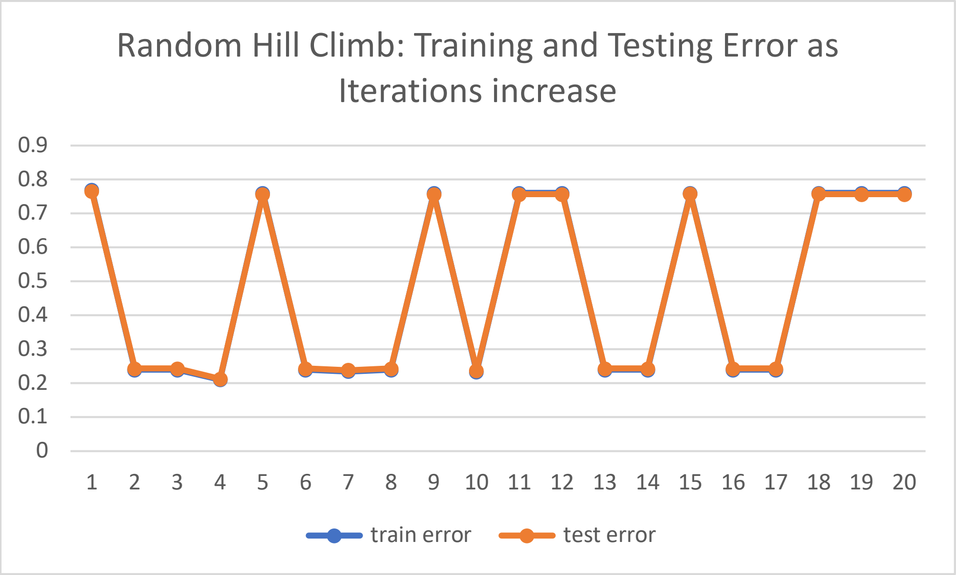


Neural Networks:

For this section, I applied the three randomized search algorithms to learning the weights of a neural network. I used the same adult dataset from my first project, which is very large, and contained numerous features such as an adult's age, gender, race, marital status, income, etc., and predicted their education level. This problem presented issues because the search space is not perfectly convex, and therefore we can easily get stuck in local optima is we don't explore our space enough. We first compare number of training iterations on both performance and time. We compare performance versus different hyperparameters to analyze their effects on finding better optima. The network architecture for the neural network stays constant throughout each randomized search algorithm. As can be seen from the graphs below, varying the number of iterations did not have a strong effect on the accuracy of the neural network, for both training and testing error. I hypothesize that this may be due to the algorithms getting stuck in the local minima, as I had feared. Despite this, all four algorithms performed relatively well with this neural network. There was a large discrepancy in the evaluation times of each algorithm, especially the genetic algorithm, which took a lot longer than everyone else to evaluate, and this evaluation time only increased more and more as the iterations increased. This may be due to the fact that genetic algorithms must evaluate the entire population for every generation so it makes sense that it takes longer. Additionally, genetic algorithms analyze mutation and crossover ratios, which take more time.







Analysis:

We have seen and analyzed the performance the four randomized optimization algorithms with applications in three different problem domains, namely finding weights for neural networks, solving the traveling salesman problem, solving the Max K Coloring problem, and solving the Four Peaks problem. For the neural network problem, we saw that GA takes significantly longer to evaluate as the number of iterations increases. In these problems, the accuracy of Simulated Annealing was comparable, especially when the cooling hyper-parameter was greater than 1. For the travelling salesman problem however, it is clear that GA outperformed the others on all metrics we considered. The evaluation time was comparable to the other algorithms, yet the optimal solution was consistently better, on all iterations. As mentioned earlier, I hypothesized that this was due to the nature of the problem, i.e. two individuals that have short paths between certain nodes could crossover and combine their short paths to produce a more optimal solution. After running all the algorithms and analyzing their weaknesses and strengths from their respective performance on the selected problems, we can conclude that there is still room for improvement. For future work, we could optimize the neural nets weight learning further by choosing different neural network architectures (different input, hidden, and output layers), or expanding the feature space beyond the rather low dimensional one utilized for this assignment. Furthermore, other optimization problems could be analyzed and trained on more selected problems. Overall, I have learned a lot from this assignment and analysis, and have developed a good understanding of the weaknesses and strengths of each randomized optimization algorithm and their applications to various problems.