Jeremy Kang
Jeremy.kang@gatech.edu
CS 7641: Machine Learning – Assignment 2

Datasets

For the implementation of the first three randomized optimization algorithms, the car evaluation data set from assignment 1 was use. To recap, the dataset was used for a car evaluation system that rated the car unacceptable, acceptable, good, or very good accordance to six dimensions: overall price (v-high, high, med, low), price of maintenance (v-high, high, med, low), number of doors (2, 3, 4, 5-more), person capacity (2, 4, more), size of luggage boot (small, med, big), and safety (low, med, high). There are 1728 instances with a total of 7 attributes (i.e., including the class attribute).

The data required preprocessing in order for ABGAIL to handle the data set. First, features and the class label had to be separated. Furthermore, the data was split into 80% (n = 1382) training and 20% (n = 346) testing sets. The variables were then converted to numeric values with the starting value of 0 (e.g., for overall price, v-high = 0, high = 1, etc.). For ease of implementation, the class label was converted into binary code in which unacceptable and acceptable were 0 and good and very good were 1. This has important implications to the implementation of the three algorithms and is further discussed in the conclusion.

Software and Approach

ABAGAIL and Microsoft Excel were exclusively used for the entirety of this assignment. For the first part of the assignment, the dataset used 80% training and 20% training split. Running the NNClassificationTest (i.e., back propagation) suggests that the data converges with as little as 30 iterations and up to around 5000 iterations. To make sure the algorithms were not stuck in a local optima, the iteration cap was set to 10,001 and were sequenced at 500 starting at 1 iteration (i.e., the first iteration was 1, second was 501, etc.). This was because even at 100,000 iterations, the error rate was approximately the same as at 10,001 and even as low as 501 iterations in some of the algorithms. Performance was measured in two ways: error rate as a function of iterations and time as a function of iterations.

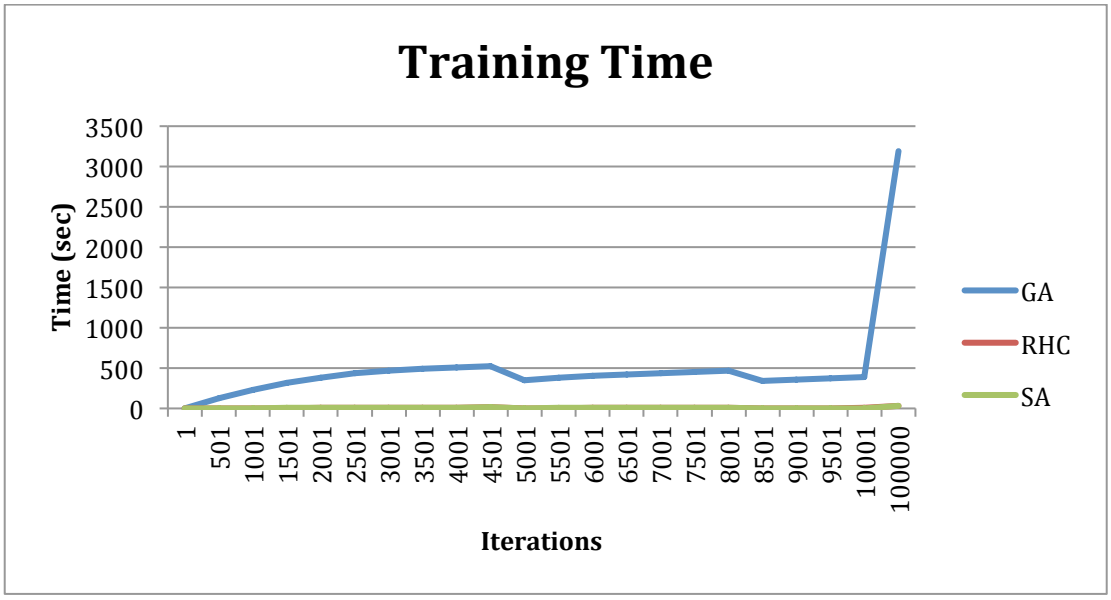
Five tests were run in order to see which algorithm (i.e., genetic algorithm, simulated annealing, and MIMIC) performed the best: FlipFlopTest, FourPeaksTest, MaxKColoringTest, NQueensTest, and TravelingSalesmanTest. RHC was also included in the tests. Iterations were also capped at 10,001. This cap was more because of lack of computational power and time constraints. In fact, if more time was allotted, the data suggests the algorithms could have continuously been improved with more iterations and, consequently, the data presented may only be a local optima and not a global optimum. Performance for these tests was measured in two ways: the optimum as a function of iterations and time as a function of iterations.

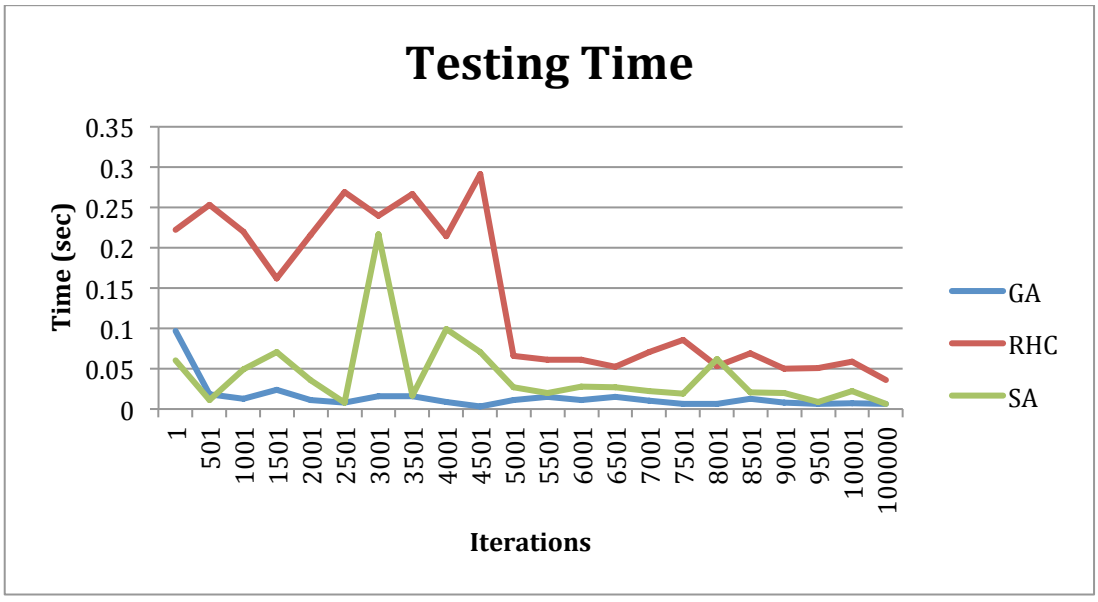
Classification Problem

Classifying car acceptability (i.e., unacceptable, acceptable, good, or very good) based on the car's overall price, price of maintenance, number of doors, person capacity, size of luggage boot, and safety.

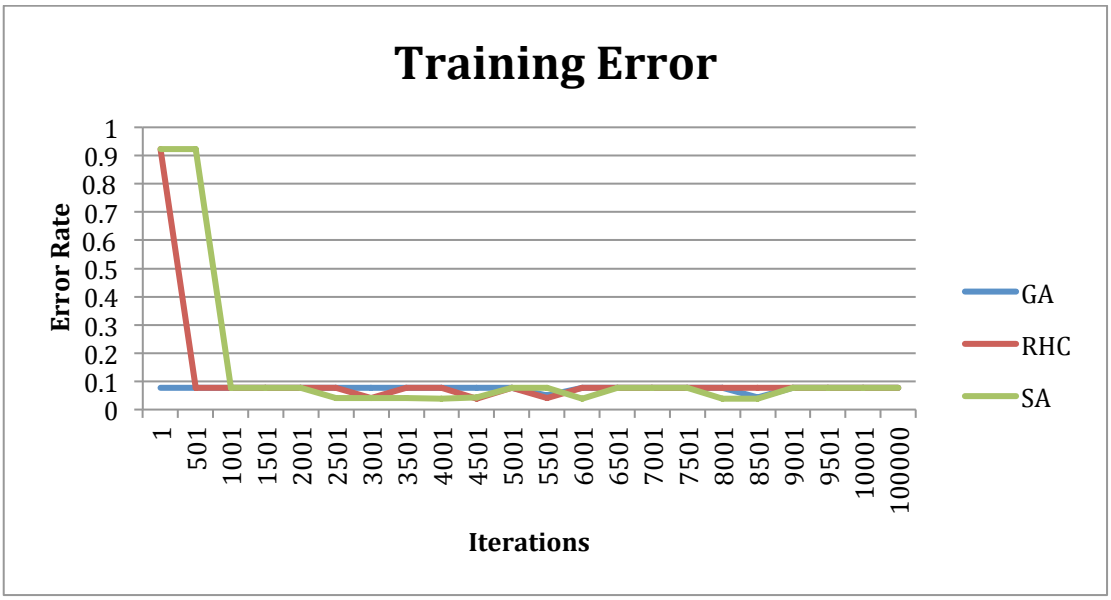
Algorithm Implementation

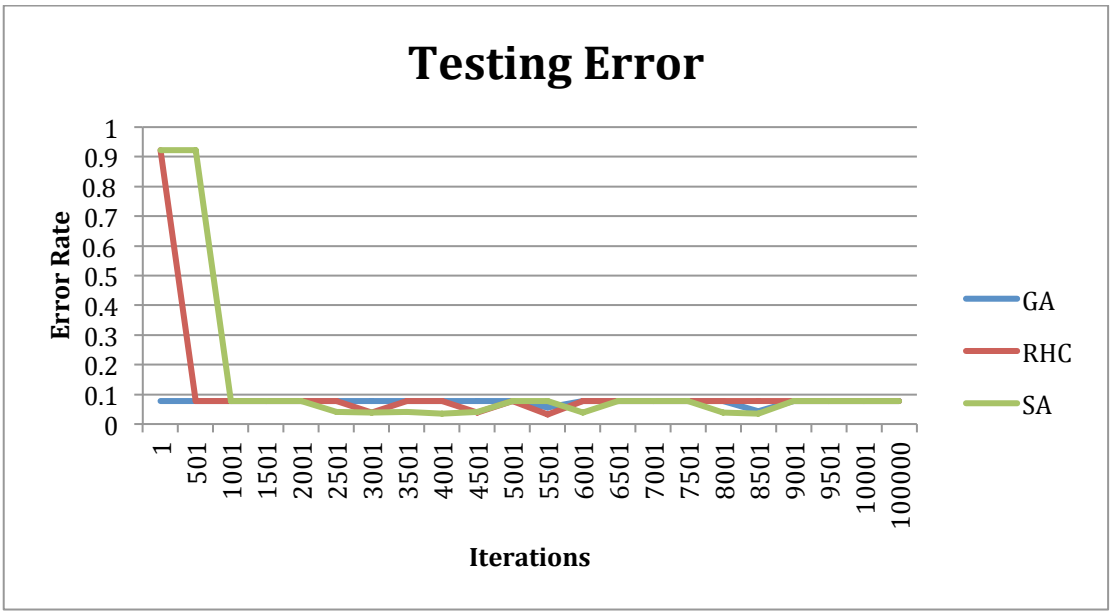
Genetic algorithm (GA), randomized hill climbing (RHC), and simulated annealing (SA) were applied to the car evaluation data set. Below are the results for the training and testing error, as well as the training and testing time.





Not surprisingly, the SA and RHC's training time is significantly lower than GA's training time. Crossovers take a significant amount of time and may account for an average of the 502.38 seconds versus the 6.41 seconds average for RHC and 0.14 seconds average for SA. However, GA seems to take significantly lower time than the other two algorithms for the test set. This may be explained by the fact that GA's algorithm selects the most fit points and even at 1 iteration, it converges to an average error rate of 7%.





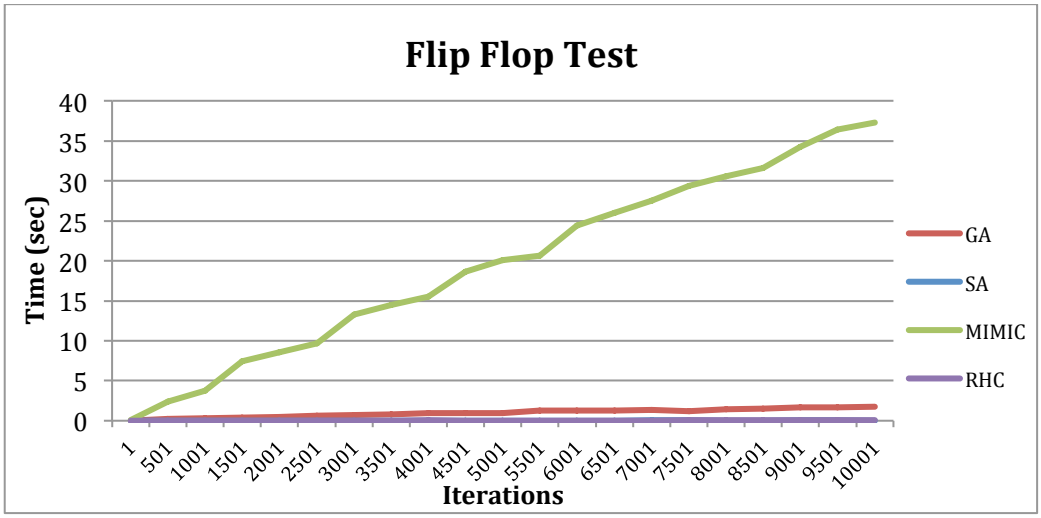
Interestingly, the training and testing error were essentially identical in the error rates respective to the each of the individual algorithms (and even to each other, excluding the iterations between 1and 501). Optimization algorithms are already

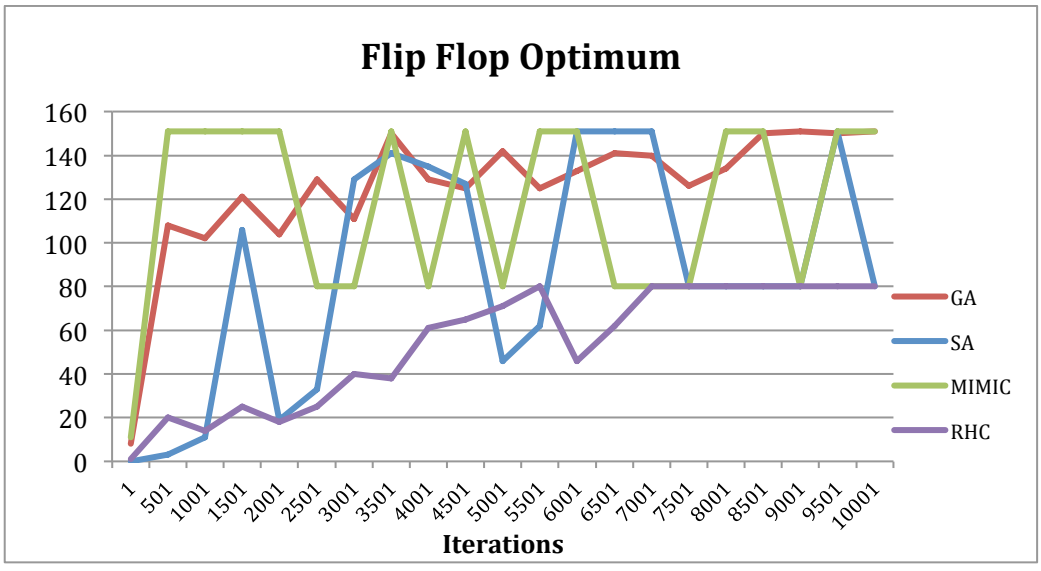
relatively fast in run time with a low sample size. Thus, the low instances may play a huge factor in the performance similarity. The differences for RHC and SA iterations between 1 and 501 (versus GA's convergence at only 1 iteration) can be explained by the way they randomly select a point (SA is just searching and RHC is a random point until convergence). As mentioned before, the error rate for the all three algorithms are at about 7%. This is almost two times higher than the error rate found when using the multilayer perceptron NN to run the data at about 4%. Nevertheless, the difference between 96% and 93% accuracy is quite marginal.

Test/Optimization Problems

Five tests were completed before finding the best performance for GA, SA, and MIMIC algorithms. RHC was also included for comparison. All of the tests were run with default N, T, (and L and K for MaxKColoringTest) values.

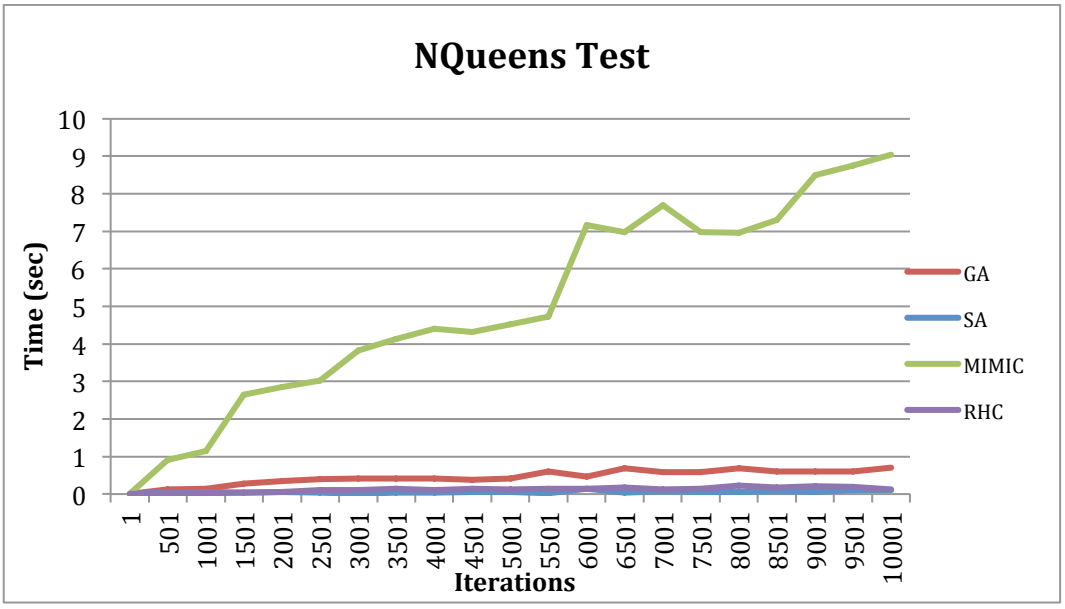
FlipFlopTest - GA

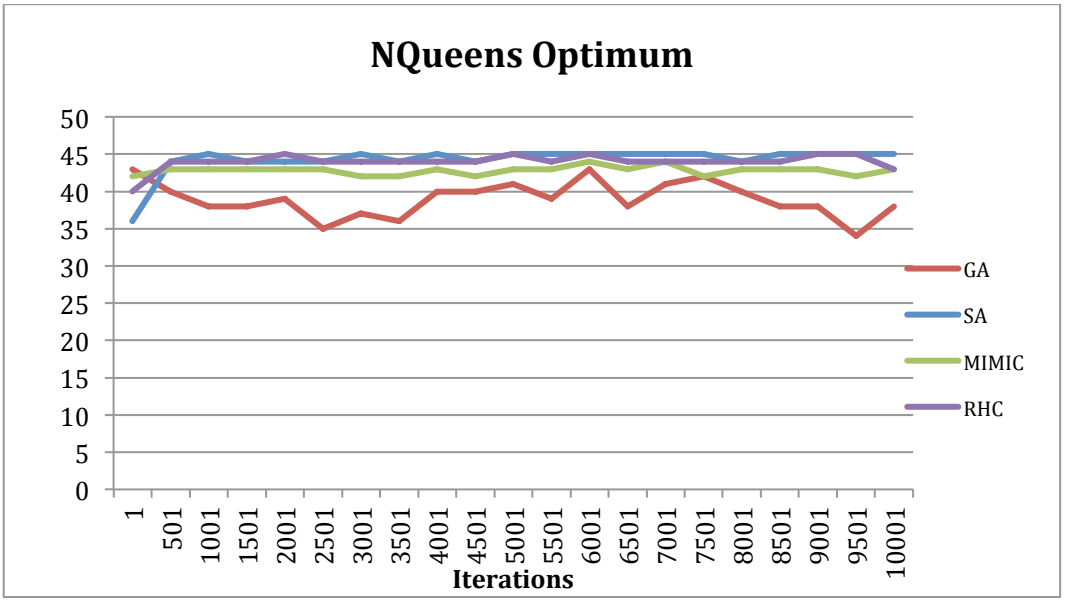




There appears to be an up and down pattern for all the algorithms that may be explained by the problem itself (i.e., flip flop). The FlipFlopTest used an N of 80 and T of 8. GA, on average, performed better than the rest of the algorithms even if MIMIC and SA reached highest optimum of 151. This is consistent with the findings from Aezina et. al (2006) in which a GA is used to optimize a transistor sizing system.

NQueensTest - SA

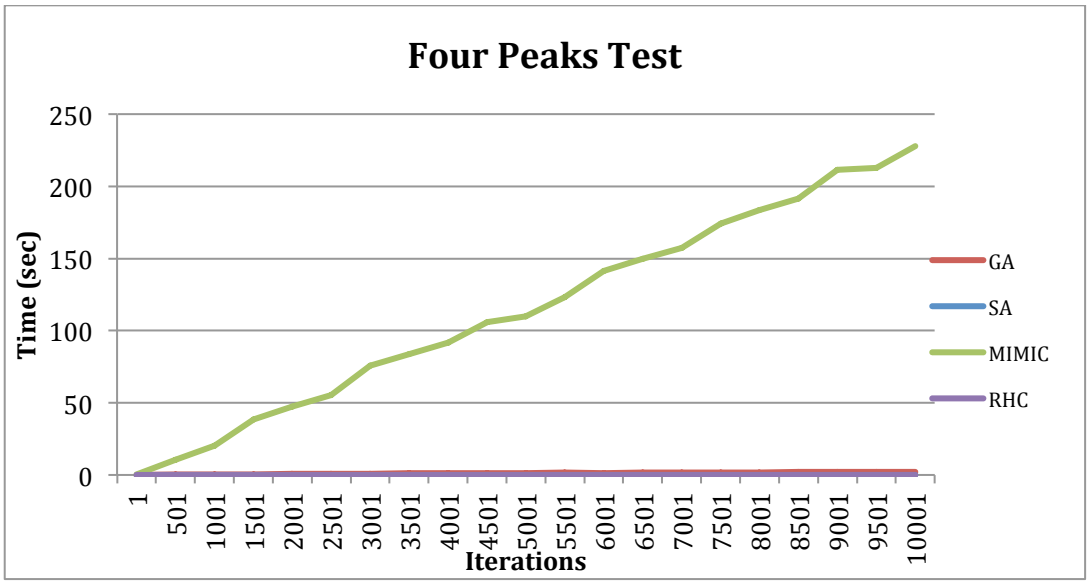


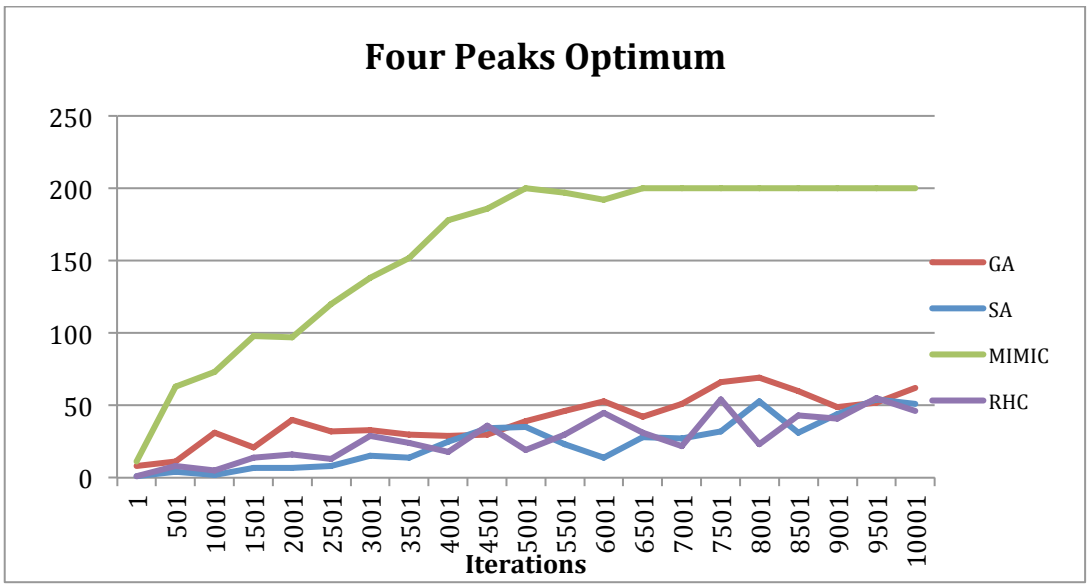


Although all of the algorithms performed very similarly, SA was found to be the best algorithm for the NQueensTest. This makes a lot of intuitive sense because SA randomly searches the board spaces in order to see which move might be the most

optimal. SA does not necessarily seek the best solution every single time, it just searches what is available and then finds what is the most optimal in the available choices. The NQueensTest is further described in Peter Alfeld's site.

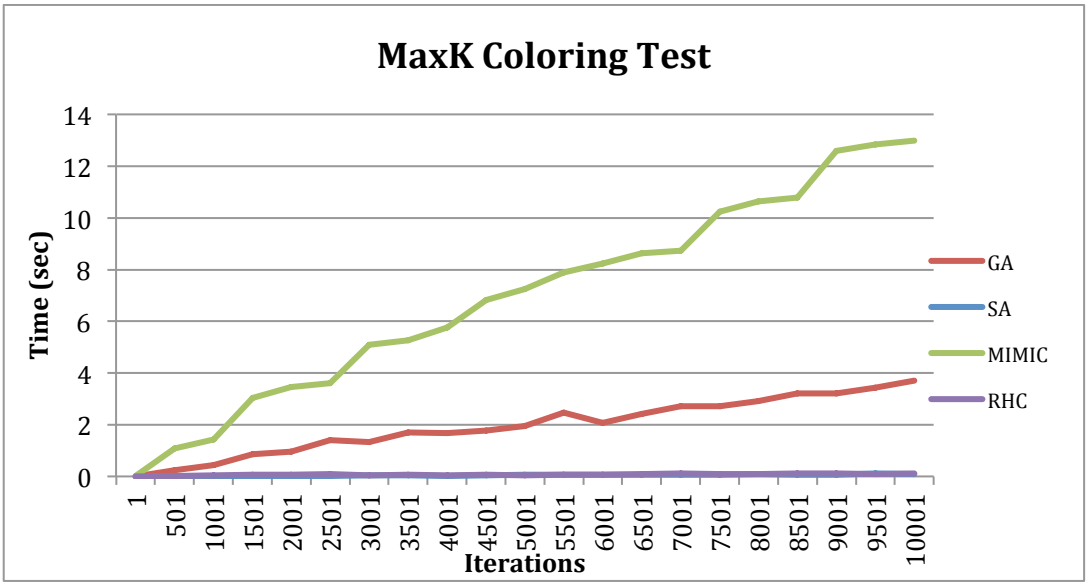
FourPeaksTest - MIMIC

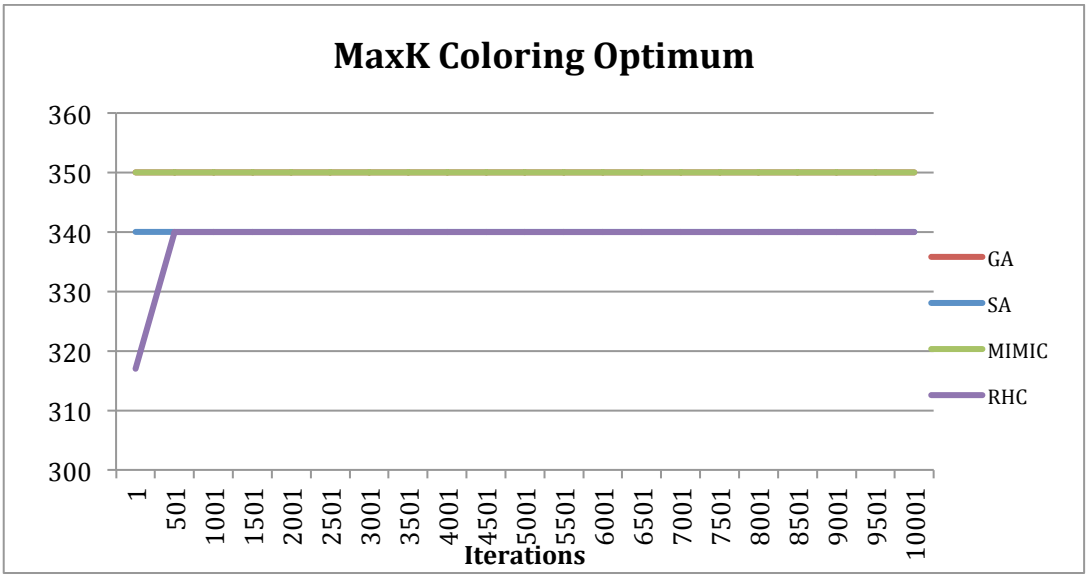




Compared to the other tests, it is much more clear to see that MIMIC performs the best with the FourPeaksTest. It appears that MIMIC either converges at about 200 or is stuck in a local optima. The rest of the algorithms suggest further iterations may be needed.

MaxKColoringTest – MIMIC/GA

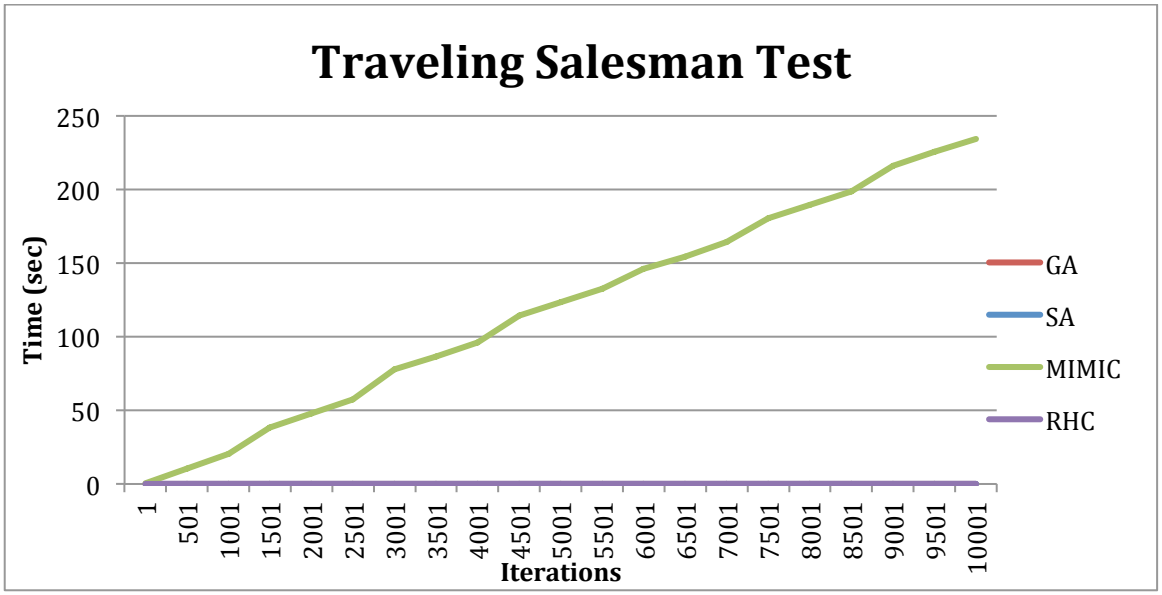


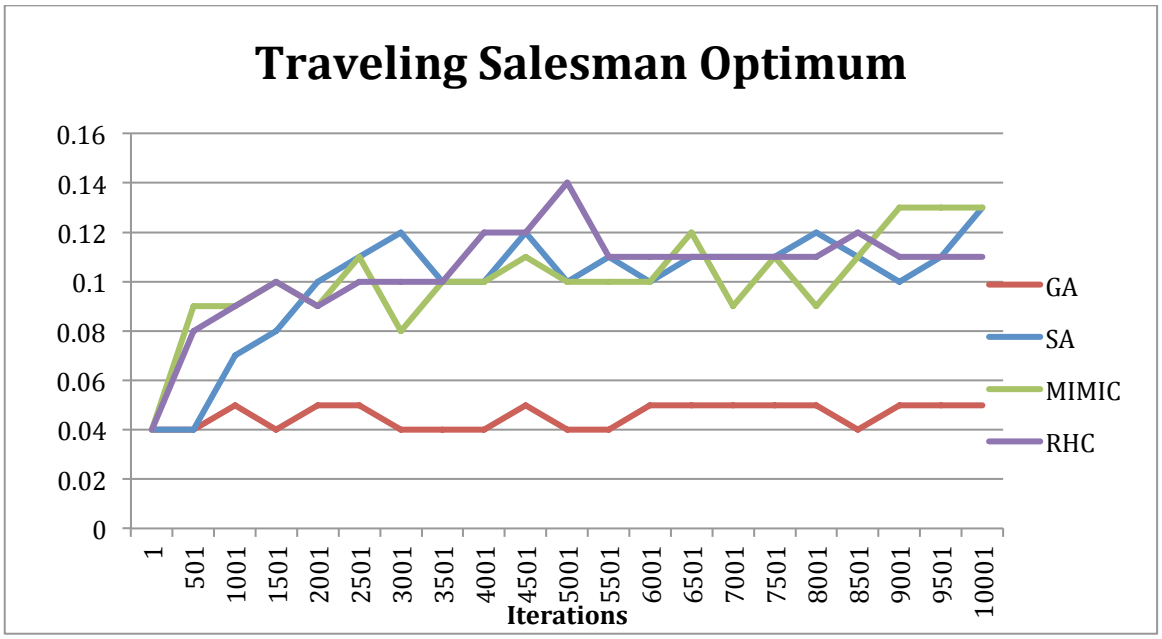


It is somewhat difficult to see from the graph, but MIMIC and GA perform exactly the same for the MaxKTest. Furthermore, RHC and SA seem to perform nearly the same as well (excluding 1 iteration for RHC which is explained by RHC choosing a random spot after seeing a local optimum. RHC chose 317 for 1 iteration and chose 340 for the rest of the iterations). These flat planes for nearly all of the algorithms suggest two things. Optimistically, all of the algorithms converged to a true global optimum and will no longer find any higher optimum with more iterations. However, if you look further into MaxKColoringTest's code, you can see that the number of vertices (N), adjacent nodes per vertex (L), and possible colors (K), are adjustable. Unfortunately due to time

constraints, I was not able to see how changing these parameters will affect the algorithms. In addition, a higher amount of iterations may have produced more variance.

TravelingSalesmanTest – SA/MIMIC





Interestingly, RHC had the highest optimum at 0.14 and highest average optimum at 0.10. MIMIC and SA perform significantly well and the results may have to do with the low sample size. GA was expected to perform the best according to a YouTube tutorial, but was found to be the worst performing. The graph suggests GA may even be stuck in a local optima because it's peak at 0.05.

Conclusion

It is important to acknowledge that a fundamental design choice may have skewed the results. Separating the class labels into binary code may not have been the most optimal choice. Specifically in this problem, unacceptable and acceptable ratings of the car are put together. Semantically, this does not make a lot of sense and may not be helpful in car evaluation choice. While the option to use vectors to discretize the class labels exists, I decided to follow the class label discretization method used in the AbaloneTest to apply to the car evaluation data. It might have made more sense to have unacceptable be 0 and the rest of the labels be 1.

Additionally, in regards to the first part of the assignment, iterations were capped at 10,001 because the error rates up to that point were rather stable at about 8%. Back propagation even suggests that the data converges with a significantly lower amount of iterations (i.e., as low as about 30). Although 100% accuracy might be ideal, 92% accuracy is certainly significant, especially considering that having 100% accuracy on the training error will allude to the algorithms overfitting. Testing the iterations at 100,000 didn't seem to produce better performance (i.e., the error rate was about 8%).

This is most likely due to a somewhat low number of instances available to the algorithms. 1728 (1382 training and 346 testing) examples for the algorithms to iterate through is comparatively low to other data sets that were available with $> 10{,}000$ instances. It can be argued that this allows for a tighter and more accurate fit depending on the data. However, the global optimum may not have even been present in the data because of a lower sample size. Thus, the algorithms may have been stuck at a local optima because of insufficient amount of data. This is may explain why the algorithms for the MaxKColoringTest all had the same outputs (i.e., suggesting all of the algorithms were either stuck, not enough iterations were completed, and/or the optimum was truly seen at about 340-350) relative to each of the algorithm. This is quite peculiar because even MIMIC – where restarts are essentially free - seems to suggest 350 are the best optimum (although MIMIC is not completely resistant to becoming stuck).

The iterations for the five tests were also capped at 10,001. This may also be a fundamental flaw because the data suggest convergence may happen with more iterations and the current results are only a local optimum. In addition, temperature and samples (and number of colors and nodes per vertex for MaxKColoringTest) were all adjustable. Unfortunately, I did not have enough time to see how these affected the performance, but it is also known that these also play a role in determining the local optimum (especially for SA).

There is an interesting pattern that is to be expected of MIMIC in which the training time is linear (except for the case in FourPeaksTest where it takes logarithmic time to converge). Just because MIMIC requires fewer iterations doesn't necessarily mean that MIMIC will take less time.

References

Aezinia, F.; Afzali-Kusha, A.; Lucas, C., "Optimizing High Speed Flip-Flop Using Genetic Algorithm," in *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on* , vol., no., pp.1787-1790, 4-7 Dec. 2006
doi: 10.1109/APCCAS.2006.342165

Alfeld, P. The N by N Queens Problem. University of Utah.
http://www.math.utah.edu/~alfeld/queens/queens.html

Solving the traveling Santa Claus problem with Genetic Algorithm: Tutorial by Student Dave.
https://youtu.be/KeoYepNcXTs