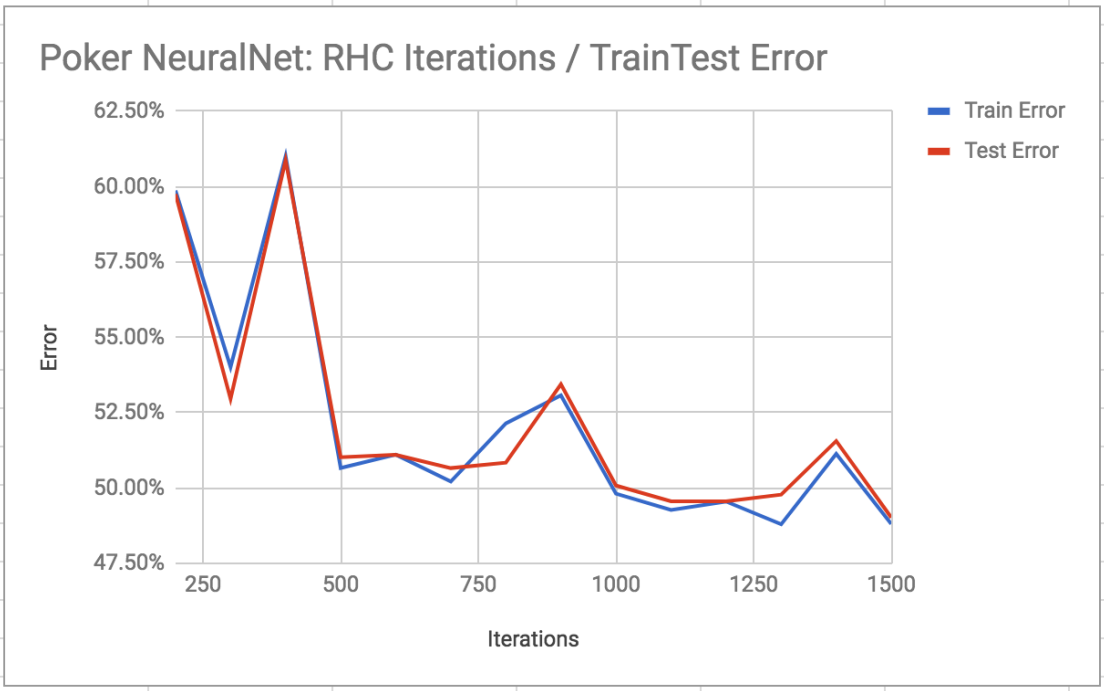
CS4641 Project 2 Analysis Report

Kristian Suhartono / 903392481

I. Neural Network

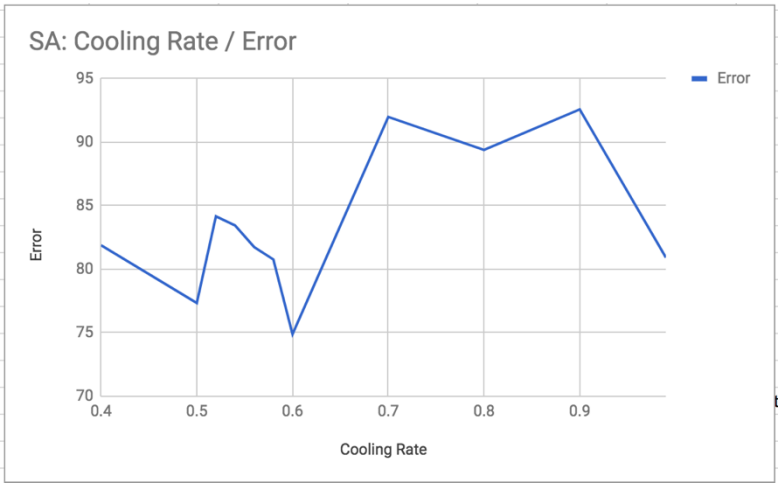
The neural network problem was ran on my Poker Dataset from assignment 1. As a reference, this dataset tries to classify a type of poker hand given 5 poker cards.

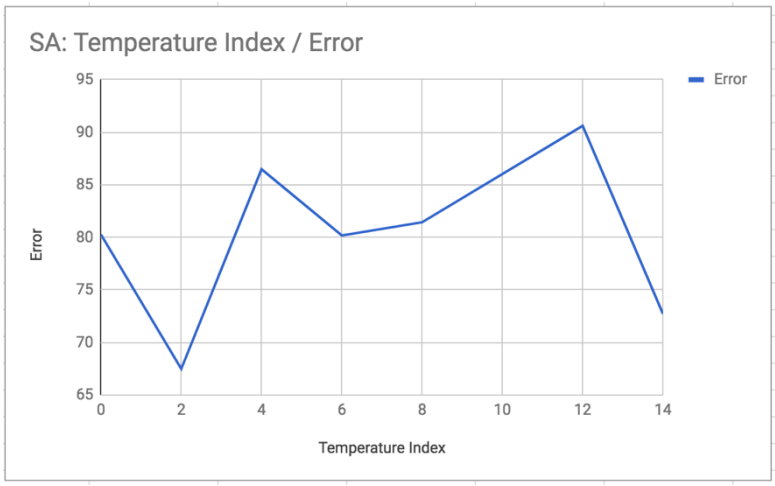
First approach that I tried was using RHC instead of backpropagation, since RHC had no actual hyperparameters, the only thing I played with was the number of iterations.



From the graph above we can see that even with RHC, the characteristics of test error following the train error is still realized here. And the error keeps on decreasing as the number of iterations increase, as expected since it has been allowed to train longer. It doesn't seem to have overfitted either as test and train error is still really close. Notice that at times the test error had worked better than train error, this is probably due to misclassification of some rarer cases, where in the test-set, due to the distribution of the data there is a lesser or even no amounts of these rarer datas.

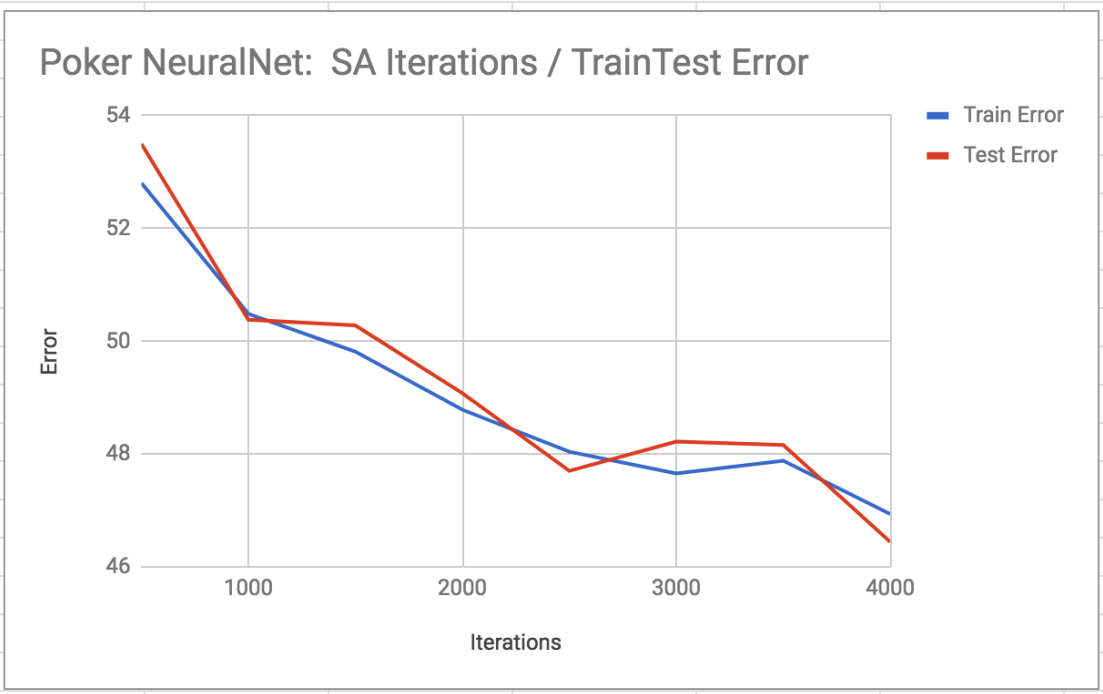
Next approach, Simulated Annealing. Here I tuned the temperature index and cooling rate to try and see what kind of effect it would have on the accuracy.



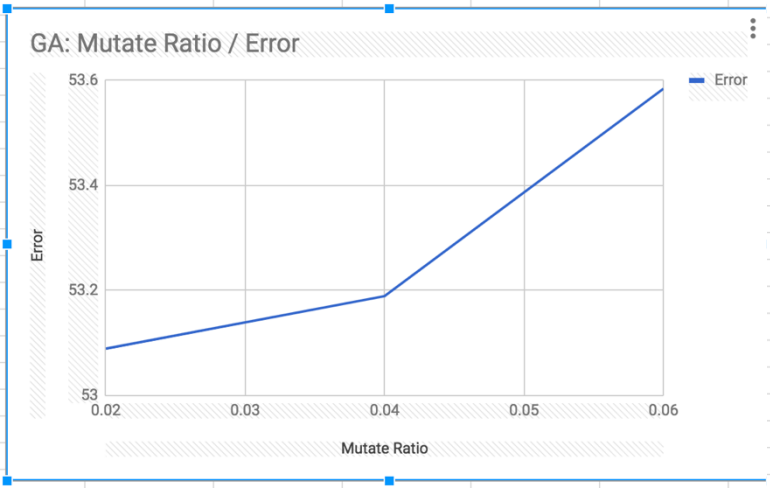


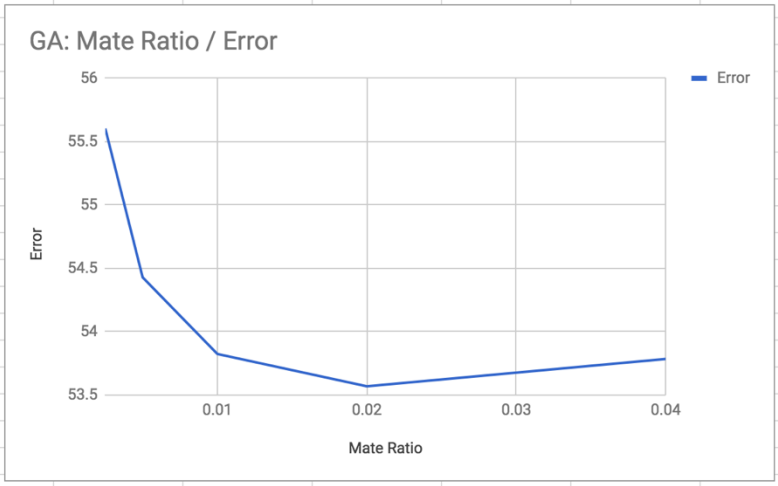
Note, the x-axis on temperature index is n in 1 * 10^n. Due to some labelling error I had to show it like this.

The above graphs show the effects of tuning the hyperparameters of Simulated Annealing. I don't see a particular pattern here rather than the error being ridiculously high with higher cooling rates and temperature indexes. In this case, I do believe it means that in the case of my dataset it is more inclined to do with more RHC like behavior, too much random-walk like behavior causes it to never converge to the optima.

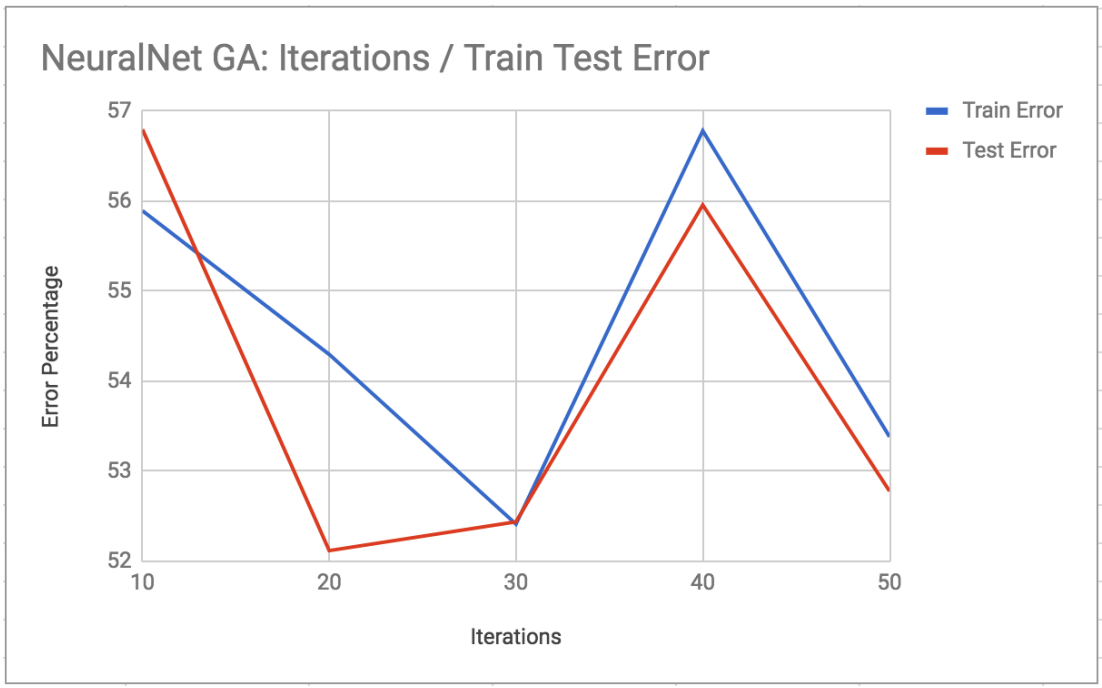


In the graph above, we can see that the Train Test Error of SA is similar to RHC, they both reached about 50% near the 1500 mark. I would attribute this to the fact that the hyperparameters were tuned to be more similar to RHC rather than random walk, causing a similarity in the results. Thus, I think the conclusion can be drawn that this dataset favors more RHC-like approaches.

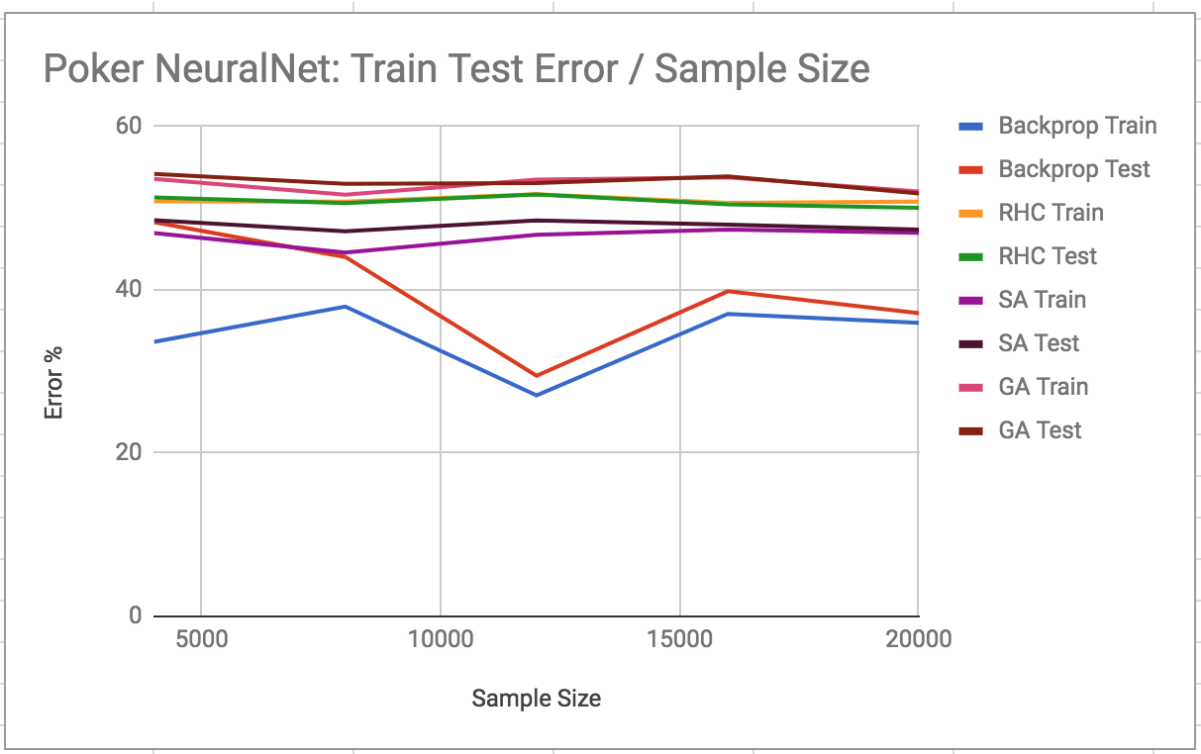




Next approach is Genetic Algorithms, above we have the results from tuning the hyperparameters. It seems that the mate and mutate ratio just fits in the 0.02 point, points other than that seem to just increase the error. I guess these ratios are the sweetspot for the GA.



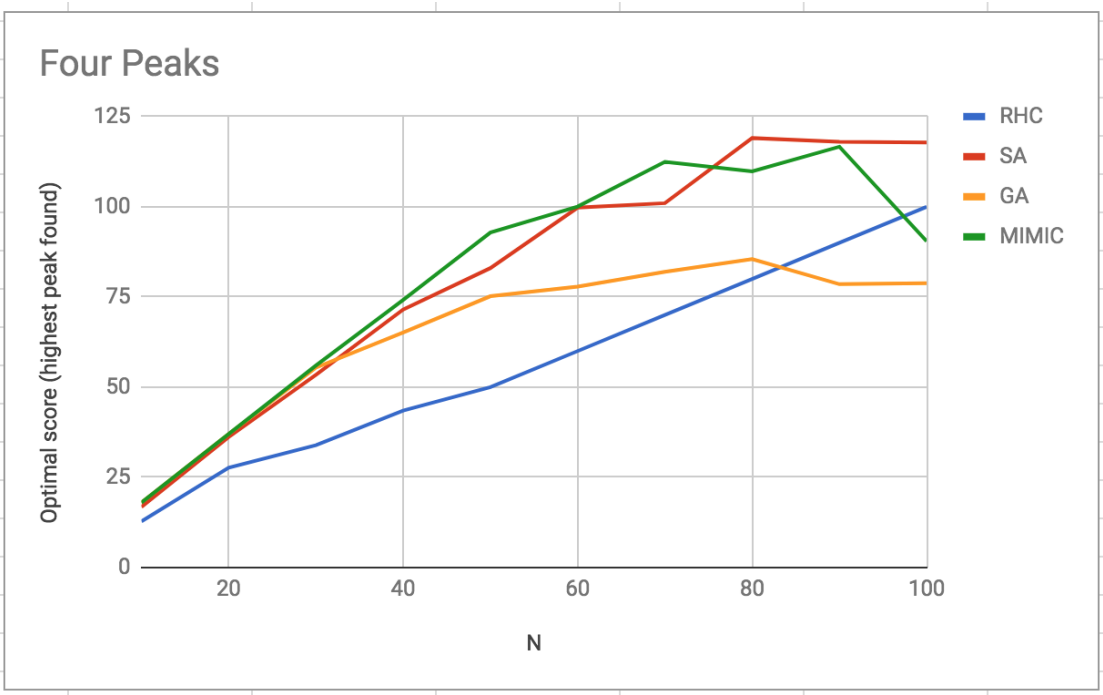
And the above graph is the train test error over number of iterations for GA. GA runs really slowly in this case compared to previous approaches. SA and RHC can finish in about 10 minutes usually, but GA took hours to run. Hence I had to limit the number of iterations to a small number. Here we can see that it's a very weird pattern. But the best point seems to be at 30 iterations where train test errors converge at a rather low point. After that they spike around more and train error is actually worse than test error, which again follows from previous graphs, and is likely caused by the same thing which is the test set has more easy to predict cases.



The above chart is the train test error of all the approaches over training set sample size. Here we can see that GA actually performed worst, with RHC following, and SA being the best, however none of the 3 random optimization approaches could even come close with backpropagation results, a solid 10% difference for backprop accuracy. I'd guess that the poor performance by GA is caused by unsuitable functions for crossover, mutation, and reduction. These functions probably didn't suit the dataset very well hence the poor results. Then there's the case of SA being better than RHC, which seems likely due to the random aspect in SA which allows it to not get stuck in local maximas, making it more likely to reach the target classification. Lastly, the reason why I believe that backprop worked better here is that there are some cases that may be rarer in the dataset, thus causing the random optimization methods to be less able in learning the weights for these rarer classes. This causes more misclassifications in the case of the random optimization approaches for this dataset. Hence backpropagation worked better here.

II. Four Peaks

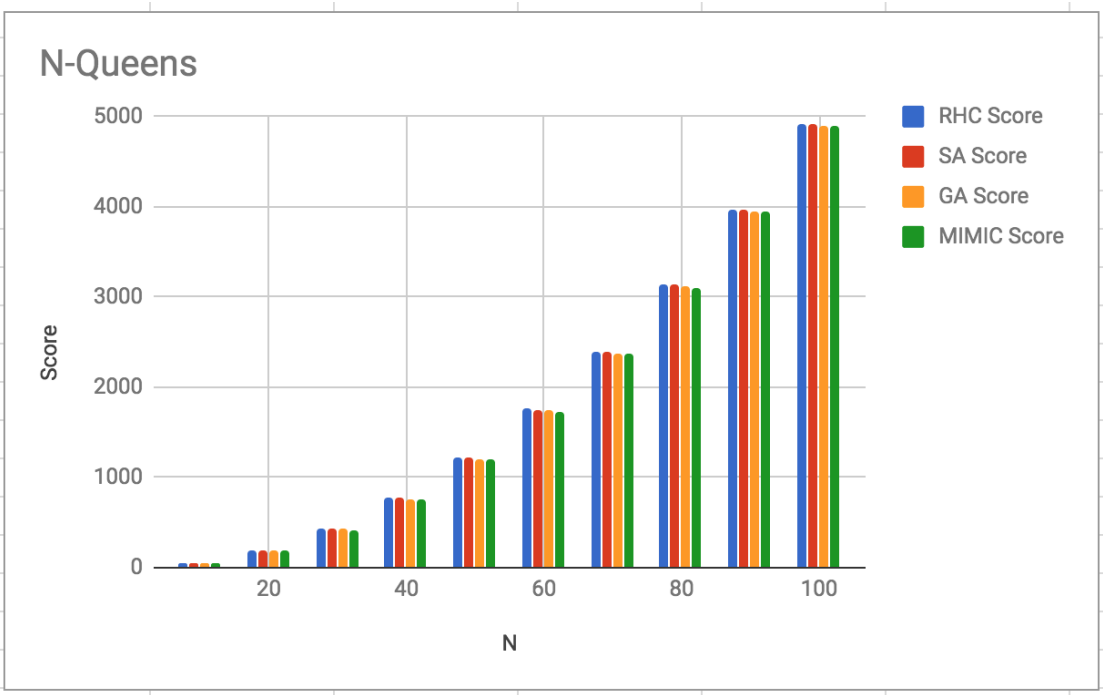
The four peaks problem is a standard optimization problem where there are 4 peaks in the data, where 2 are global maximas and 2 are local maximas. It's very easy for a human to understand this but random optimization methods may get stuck easily on the local minimas. The input is an N dimensional input vector, and the T defines the maximas, where a maxima is found when there are T+1 leading 1's followed by all 0's or T + 1 leading 0's preceded by all 1's. For my experiments, the T is always set to 10% of the N value, and also the results are an average value over 20 loops.



As shown in the figure above, as the N (input vector length) increases, MIMIC generally performs better. However, at N = 80 there is a falloff in the accuracy of MIMIC. But before this point, MIMIC is clearly shown to be the best and RHC being the worst. This is clearly due to the problem definition where RHC can only do well based on where it started, if the RHC started at a bad position, it wouldn't be able to climb the hill to the global maxima and will get stuck on a local maxima instead. MIMIC takes the longest time to train and run, with a noticeable 10x time delay as compared to the other approaches. SA does quite well in this case, the adding of the random walk seems to have benefitted SA, it is able to be comparable in results to MIMIC and even eventually overtaking it. GA surprisingly does quite poorly here, I would attribute this to probably a poor choice in the functions used for this algorithm. I believe MIMIC does better here as it is able to actually cluster to the maxima, thus consistently finding the maxima. I'm not sure what caused the accuracy drop-off in the end, but I would guess that it's an occurrence of MIMIC getting stuck in a local maxima due to the chosen probability distribution.

III. N-Queens

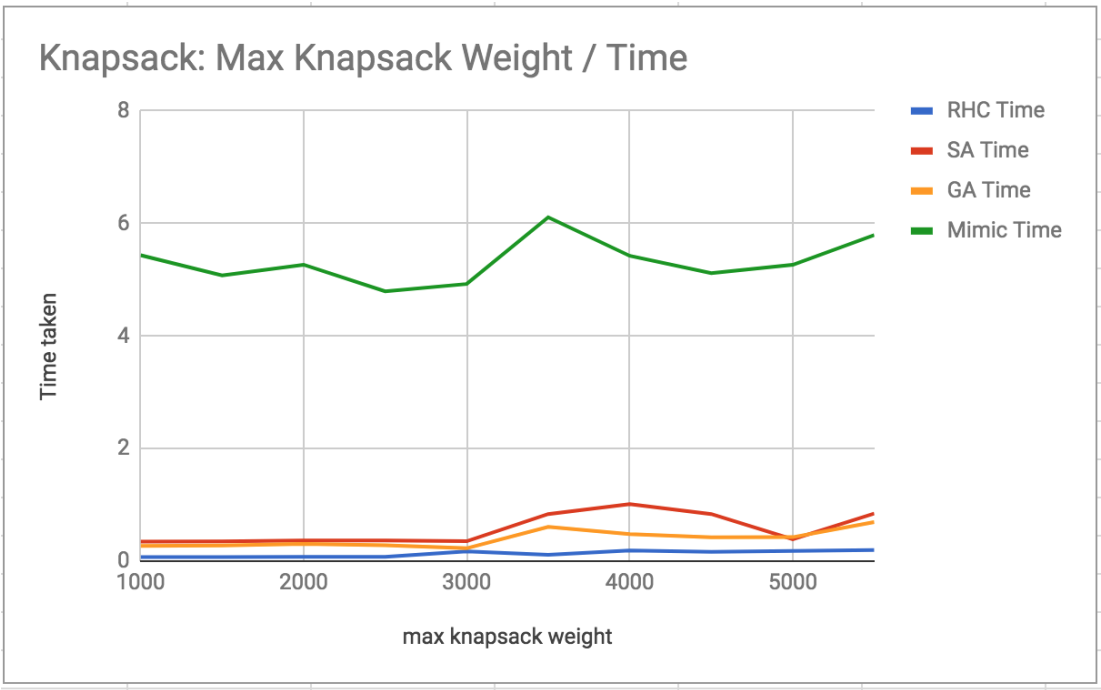
The N-Queens problem is a problem of placing N chess queens on a chessboard in such a way such that no two queens are able to attack each other. Similar as four peaks, the scores here are also averaged over multiple runs, but here it is only run 10 times.



Here we can see that RHC and SA have similar scores where RHC is the best and SA 2nd best through the whole test. I would attribute this to the fact that the search space is pretty clear, as in there aren't any tricky local maximas where the SA or RHC can get stuck in a lot, thus they are able to do their hillclimbing to its maximum potential and reach the global maxima repeatedly. The performance of the GA and MIMIC being lower than RHC and SA might be due to imperfectly tuned hyperparameters. I may be able to increase the accuracy of GA and MIMIC by playing around with the hyperparameters more.

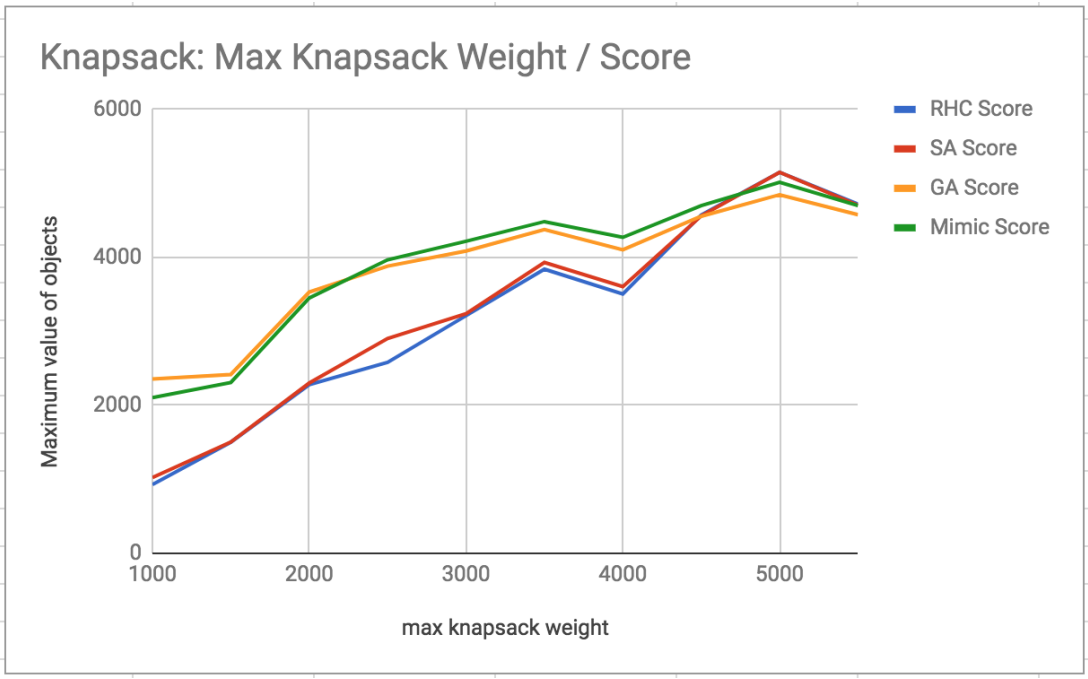
IV. Knapsack

The Knapsack problem is another standard optimization problem, where there are a number of objects each associated with a weight and value. We try to fit a knapsack which has a weight limit with as many high-valued objects as possible, as in attempting to maximize the value of objects in the knapsack.



Note that time taken in above is in miliseconds (ms).

The above graph is clearly shows that MIMIC takes much much longer than all the other approaches. RHC consistently takes the shortest amount of time, which makes sense since it only attempts to climb up from where it started. GA takes a bit longer since it has to do more computations in the form of mutation and crossovers etc. SA would probably take longer since it may have to do multiple random walks before finally reaching something it thinks is the maxima.



The above data was obtained by running with a constant number of objects, max value, and max weight of objects. There can also be 4 multiples of the same object.

It can be seen here that GA and MIMIC perform best here. With GA having an early lead and MIMIC pulling ahead as the max knapsack weights increase. This may perhaps be caused by again a less than optimal mutation and crossover functions. But GA seems to be the best approach here from the results it obtained with the amount of time it spent training. Near the end there, RHC and SA overpower MIMIC and GA. I believe this is caused by the fact that as the maximum weight increases, the maximum value converges to a certain amount, which is easily found by doing RHC, causing MIMIC and GA to perform less optimal.