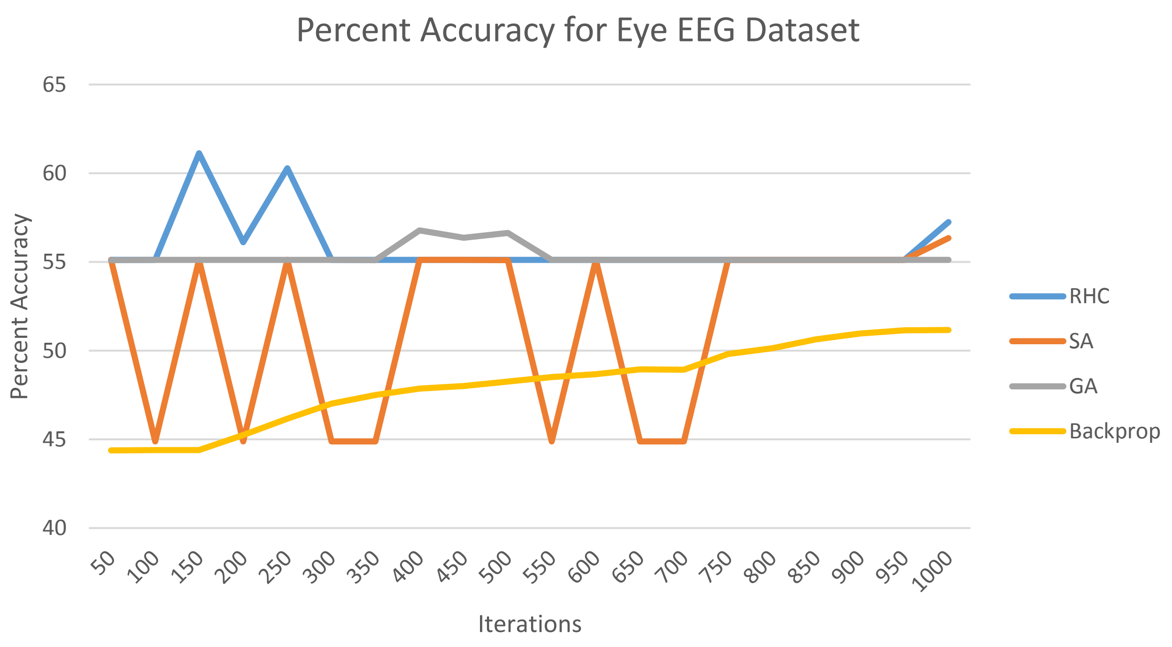
Nathan Dass
CS 4641 – Assignment 2

Randomized Learning Report

Randomized Optimization and Neural Networks

I used one of the datasets from the Supervised Learning assignment to compare the results obtained from backpropogation to results obtained from randomized optimization algorithms. The dataset that I used is the Eye EEG dataset. I chose this dataset over the Sick dataset for several reasons. The Eye EEG dataset was more evenly distributed and had no missing values. Furthermore, the Eye EEG dataset has 14 attributes and 14980 instances. The dataset classified whether or not the eye was open at a given frame. Additionally, when taking the Curse of Dimensionality into account, there is almost enough data for the number of attributes in the dataset. This means that results obtained are actually meaningful and not due to chance or other factors.

To obtain the results for backpropogation, Weka's MultiLayerPerceptron was used. The default values for a learning rate of 0.3 and a momentum of 0.2 were used when collecting data. Additionally, 5 nodes were used in the hidden layer as this was found to be the optimal amount of nodes in the hidden layer through testing the model complexity of the neural network.

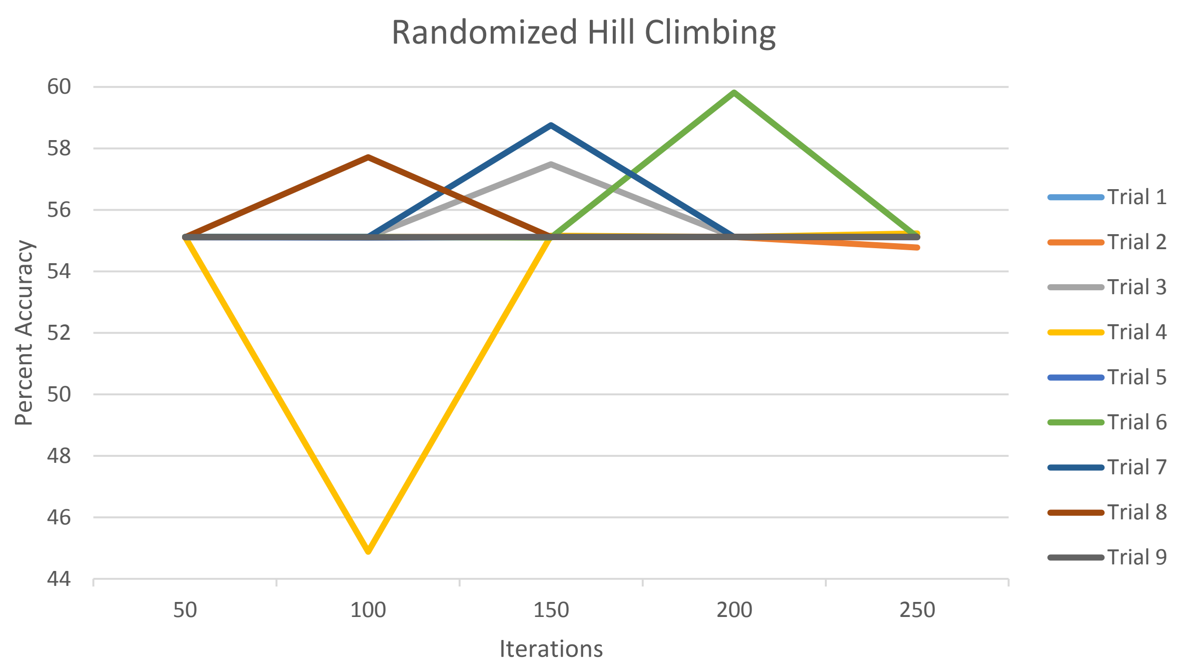


	Randomized Hill Climbing	Simulated Annealing	Genetic Algorithms	Backpropogation
Percent Correctly Classified	57.253%	56.346%	55.117%	51.1683%
Time (seconds)	58.916	59.395	2033.577	21.52

Overall, Randomized Hill Climbing performed the best for the Eye dataset. While Genetic Algorithms and Randomized Hill Climbing had similar percent accuracies, the two algorithms can be distinguished by looking at runtime. At 1000 iterations, Randomized Hill Climbing trained and tested in 58.916 seconds while Genetic Algorithms trained and tested in 2033.577 seconds. In general, Genetic Algorithms takes longer to run, but the additional time usually performs equally as well as Randomized Hill Climbing or sometimes Simulated Annealing. The drops in Simulated Annealing can be explained by bad turns made in the beginning of the algorithm that put the algorithm in a position that it could not find a higher maximum. The temperature and cooling used were 1E11 and 0.95, respectively. However, as the iterations increased, the accuracy of Simulated Annealing was comparable to Genetic Algorithms and even doing better than Genetic Algorithms while the runtime was still significantly lower. It should also be noted that while on average, the accuracy of backpropogation is lower than the randomized optimization algorithms, the backpropogation has a clear upward trend for the accuracy. This would imply that with more iterations, backpropogation would have a higher percent accuracy for this dataset. Furthermore, while backpropogation only took 21.52 seconds to run, this time increases exponentially with more iterations. With all of this in mind, we can say that with a smaller number of iterations, randomized optimization methods perform better than non-random methods. However, as the number of iterations gets larger and larger, the non-random methods would yield a higher percent accuracy than randomized optimization methods, but the non-random methods also become slower than some of the randomized optimization methods. This suggests that the dataset has some sort of structure that requires more than relying on randomness to optimize the weights, but taking feedback from the network itself would result in higher overall accuracy.

Randomized Hill Climbing

Randomized Hill Climbing is a local search algorithm based on mathematical optimization. Hill Climbing is an iterative algorithm that starts at an arbitrary solution to the problem and attempts to find a better solution by incrementally changing a single element of the solution. If this change increases the accuracy or results in a more accurate solution, the new solution is accepted as the best solution up to that point and the process repeats. A new solution is accepted only if the accuracy increases. If the accuracy decreases with a change in the element, the new solution is rejected and another change is made. The problem with this approach is that Hill Climbing will get stuck at local maxima. In order to get around this, there are random restarts where a new arbitrary solution is picked and Hill Climbing process starts again. One advantage of this algorithm is that it is an anytime algorithm, meaning that at any given time during runtime, it will always have a valid solution even if it is not the optimal solution.

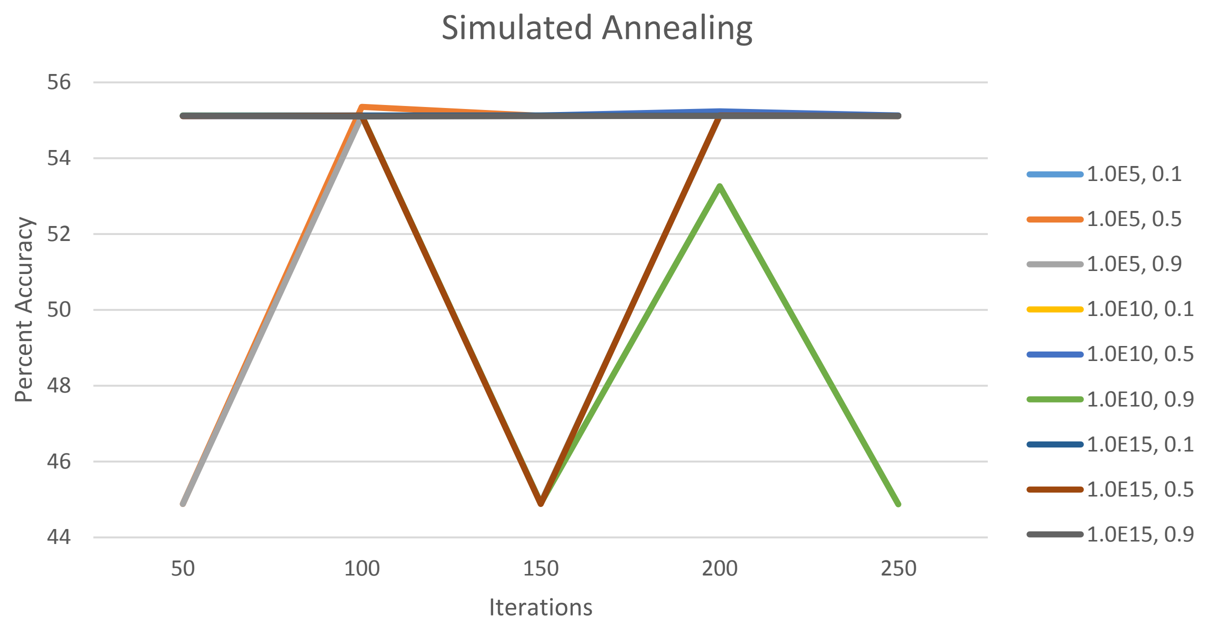


Iterations	Percent Accuracy
50	55.115%
100	54.269%
150	55.873%
200	55.641%
250	55.091%

The graph above shows the percent accuracy for nine trials of randomized hill climbing for 250 iterations on the Eye dataset. The different accuracies at iterations for the trials is due to the random restarts. To the left is a table for the average accuracy for the nine trials at each of the iterations. There is a slight upward trend as the number of iterations increases. This can be explained by the upward spikes in accuracy. This shows that there is a higher maximum that is being stumbled upon, but the algorithm just needs more iterations to reach that solution.

Simulated Annealing

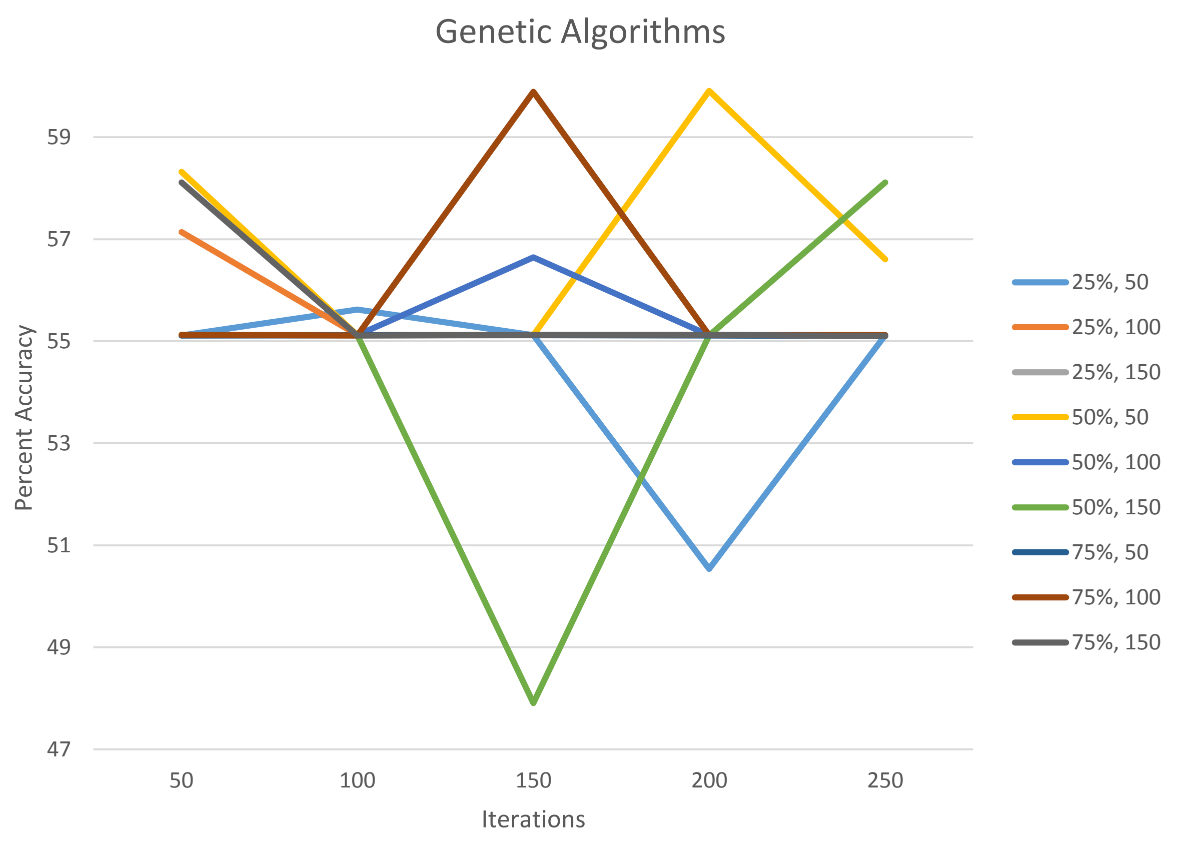
Simulated Annealing is very similar to randomized hill climbing, but adds two parameters called cooling and temperature to the algorithm. These two parameters are used to determine when to accept a worse solution. By cooling the temperature slowly, the global maximum can usually be found at a higher accuracy than for randomized hill climbing. The purpose of cooling and temperature can be thought of as something similar to a random restart, but not as severe. As the temperature decreases, the algorithm jumps less and moves more towards a standard hill climbing algorithm.



From the overall comparison of the accuracy of each of the algorithms, we can tell that Simulated Annealing does not work that well for the Eye dataset. By looking at different temperatures and coolings, we can see that the accuracy for Simulated Annealing basically has a maximum that it converges to for most combinations of temperature and coolings. Interestingly, a cooling of 0.5 and 0.9 tended to have sudden drops in accuracy which would suggest that a cooling of 0.1 was the best for the Eye dataset. A cooling of 0.1 would make Simulated Annealing close enough to Randomized Hill Climbing that it would follow the overall trend of Randomized Hill Climbing while sometimes converging to it faster. It does not appear that Simulated Annealing is a very good algorithm for this dataset, which would suggest that the underlying distribution has many peaks and dips which causes Simulated Annealing to not perform well compared to Randomized Hill Climbing.

Genetic Algorithms

Genetic Algorithms is based on natural selection. By incorporating inheritance, mutation, and selection, candidates with a higher fitness are more likely to thrive while candidates with a lower fitness are more likely to die off. With respect to neural networks, the population is the total number of weights in the domain space. The mates are then determined through the prespecified fraction of the population that performs the best. Then, mates are crossed and replace the poorer performing candidates. Sometimes, there are random mutations added to offspring to push the candidates towards the overall solution. The assumption for Genetic Algorithms is that the problem has an overall gradient towards the global maximum. Genetic Algorithms overfits to the underlying structure of the problem and the assumed gradient. The problem with this is that the underlying structure of the problem may not always be a continuous gradient. It could have many dips or local minimum that eventually converge to a global maximum or could not even be a gradient at all.



With respect to the Eye dataset, Genetic Algorithms performed well overall. The graph represents 9 different combinations of the ratio of population size to number of mates and number of mutations. A population size of 200 was used throughout the tests. With the population size being held constant, the number of mates were 50, 100, and 150 giving a 25%, 50%, and 75% ratio of mates to population size. The number of candidates to mutate at each iteration were 50, 100, and 150. Overall, a 75% ratio for number of mates to population size seemed to perform the best regardless of the number of mutations. This suggests that even candidates that perform poorly can be crossed and their offspring can result in candidates that perform well. Individually, a 50% ratio and a mutation size of 50 performed the best with an average accuracy of 57.0146%.

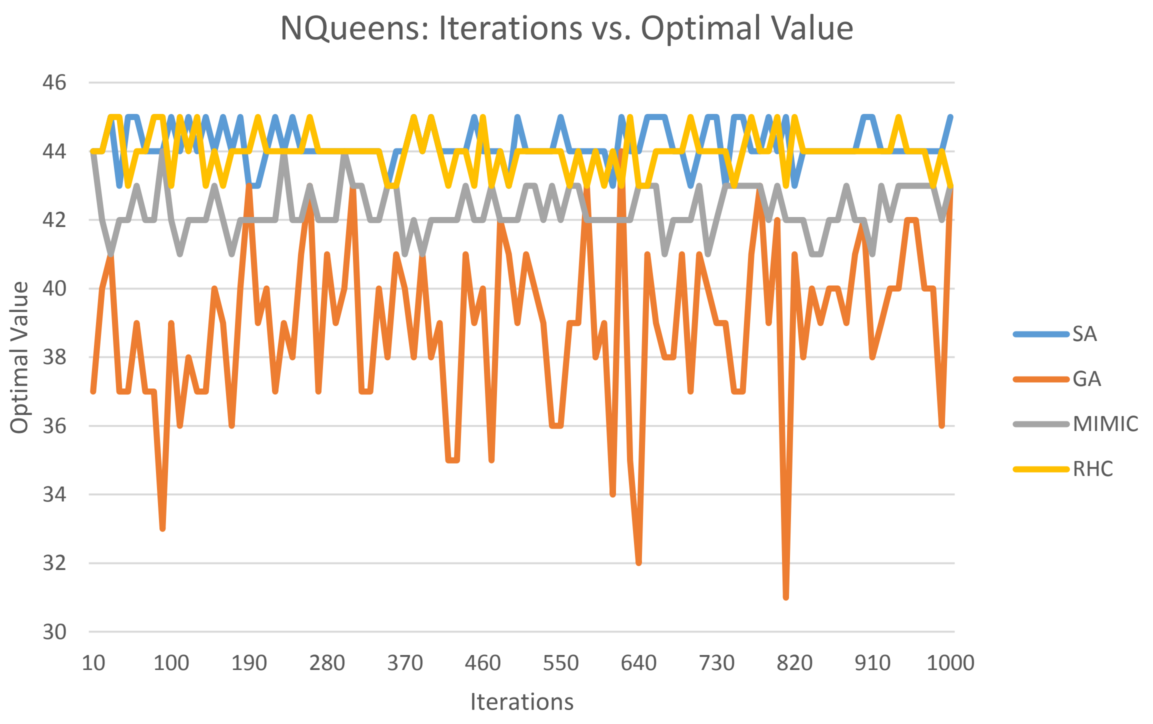
Number of mates to population size ratio	**Average percent accuracy**
25%	54.9786
50%	55.57
75%	55.63

Nathan Dass
CS 4641 – Assignment 2

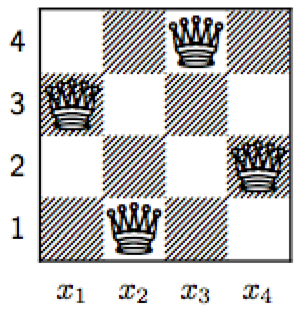
<u>Optimization Problems</u>

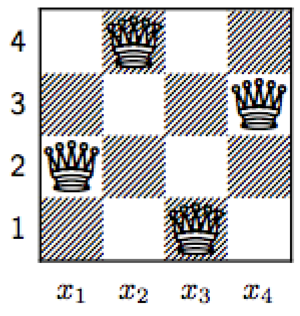
Simulated Annealing: N-Queens

The N-Queens problem involves placing N queens on an NxN chess board where n is at least 4.



The graph above shows the optimal value obtained for different iteration steps up to 1000 iterations. Simulated Annealing and Randomized Hill Climbing obtain an optimal value that is greater than or equal to the optimal value obtained through MIMIC or Genetic Algorithms. This can be explained due to the very nature of the problem itself. In this problem, you are trying to maximize the number of queens that don't collide with each other. Think about moving a queen from one position to another valid position. There are three scenarios, the number of collisions goes up, the number of collisions goes down, or the number of collisions stays the same. The second scenario is the preferred scenario, so a hill climbing algorithm would only update the position of a queen if it decreases the number of collisions. This iterative updating would either lead to a local maximum (there are a few collisions, but changing the positions of the queens does not decrease the number of collisions due to the way the queens are oriented) or lead to a global maximum (there are zero collisions). It should also be noted that there are multiple global





maxima for a given board. For example, with 4-Queens, the solution is to orient the queens a knight's move away (2 steps in one direction and 1 step in the perpendicular direction that places the queen on the edge of the board) from each other. From this, we can see that there are two possible solutions for 4-Queens. Naturally, a hill climbing based algorithm would excel at N-Queens and randomized restarts would serve as a workaround for being stuck at a local maximum. Since Simulated Annealing is an extension of Randomized Hill Climbing, Simulated Annealing would also excel at N-Queens. Additionally, Simulated Annealing outperforms Random Hill Climbing with an average optimal value of 44.19 while Randomized Hill Climbing has an average optimal value of 43.97.

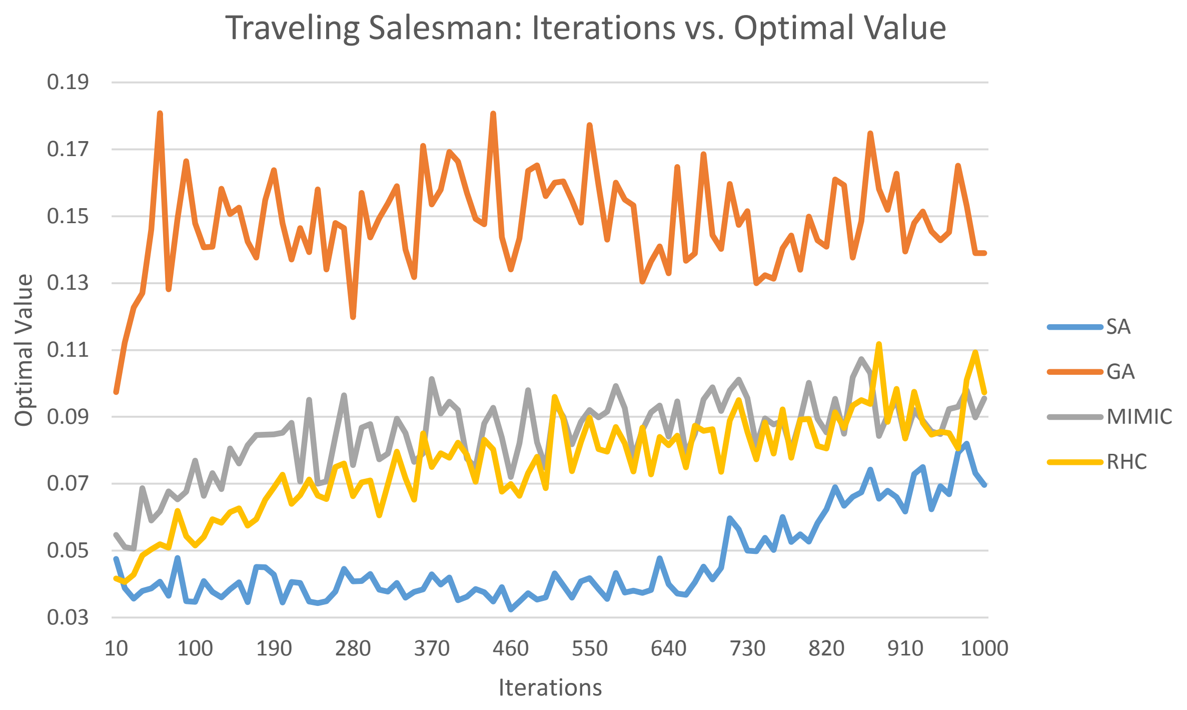
Now think about how Genetic Algorithms would be applied to N-Queens. It would take two board configurations and cross them to create a new board configuration. Occasionally, the new board configuration is mutated. The problem with Genetic Algorithms with respect to N-Queens is that given two board configurations that each have their own respective collisions, there is no guarantee that the board configuration obtained after crossing would have fewer collisions. Just because two independent boards have low collisions doesn't mean that the combined boards have even lower collisions because Genetic Algorithms assumes that the locations of the queens is what makes the configuration good. However, it is not the location of a single queen that makes a board configuration good, it is the combined locations of all of the queens together that determines the collisions. On the other hand, MIMIC tries to get rid of the underlying assumption that Genetic Algorithm makes by saying that a board configuration with a low number of collisions could be due to the problem itself, not by the locations of the individual queens (which is true in the case of N-Queens). That is why MIMIC performs better than Genetic Algorithms, but not as well as Randomized Hill Climbing and Simulated Annealing.

Algorithm	Time (seconds)
Randomized Hill Climbing	0.018194935
Simulated Annealing	0.019905565
Genetic Algorithms	0.362589392
MIMIC	0.360400062

The table to the right shows the runtime of each of the algorithms for 1000 iterations. Randomized Hill Climbing and Simulated Annealing have a comparable runtime, while Genetic Algorithms and MIMIC take about 20 times longer to run and is even less accurate. Therefore, Hill Climbing based algorithms and especially Simulated Annealing would be the best randomized optimization algorithm for the N-Queens problem.

Genetic Algorithms: Traveling Salesman

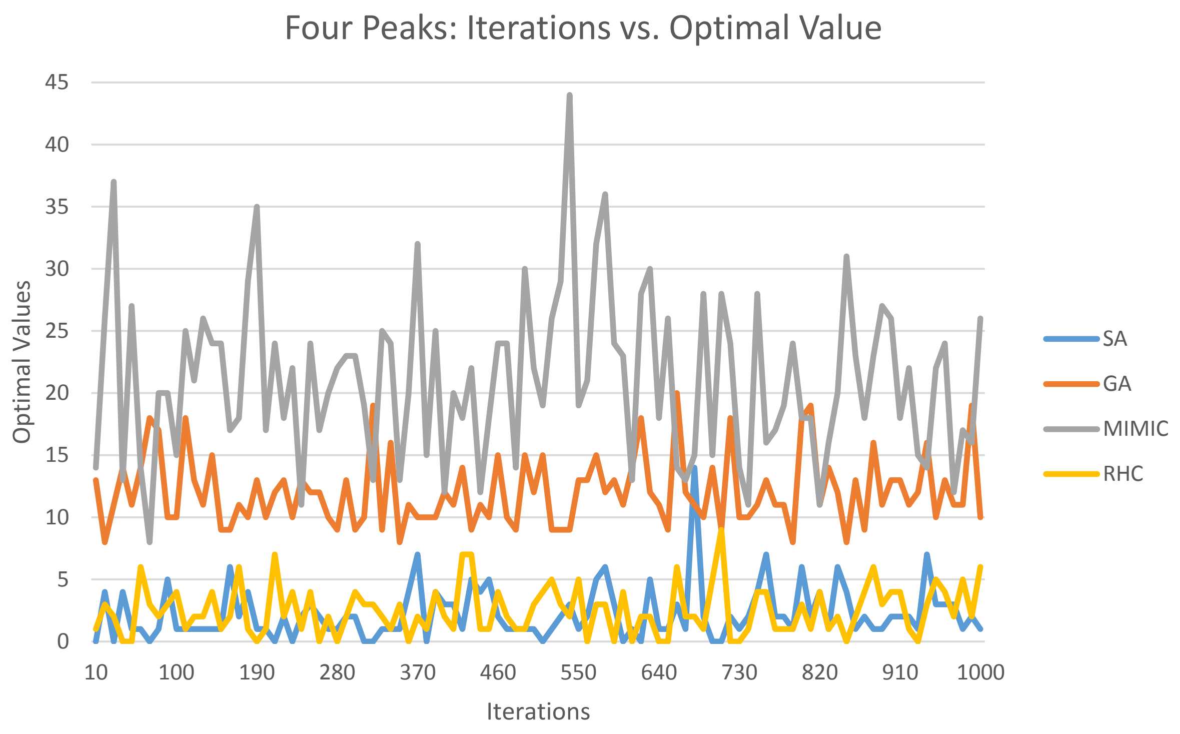
For the traveling salesman problem, you are given a list of cities and distances between each pair of cities and you have to find the shortest possible path that visits each city exactly one time and then returns to the original city.



For this problem, Genetic Algorithm clearly outperforms the other three algorithms. Simulated Annealing does not perform well because it only evaluates one neighbor at a time and the search space is just too large to find a global optimum since there are so many path combinations. Following similar logic, randomized hill climbing does not perform that well either. MIMIC performs better than Randomized Hill Climbing, but still worse than Genetic Algorithms. This is MIMIC evaluates the solution based on the probability of a true maximum being in a certain area and the fitness of the routes is not necessarily similar to routes similar to themselves. Since there are many local maximums given the large search space, MIMIC would require many more iterations to find the global maximum. Genetic Algorithms performs the best for this problem because it looks at all of the solutions and only moves on with those that are slightly altered to the candidates that performed the best in the previous iteration. In every iteration, solutions with shorter paths are the ones that are selected to continue for the next iteration. Even though Genetic Algorithms took 21.4 seconds, it is still better to use it since the optimal value is so much higher even when comparing the runtime of 0.0322 seconds for Randomized Hill Climbing, 0.033 seconds for Simulated Annealing, and especially the 1024.5 seconds for MIMIC.

MIMIC: Four Peaks

Four peaks is a function that that takes a bitstring of length, N, and a trigger position, T. Its maximum is at bitstrings that have contiguous 0s or 1s up to the trigger point, and then the complementary bit contiguously until the end of the string. It has a total of four local maxima. The absolute maximum is with a value of 2N – T – 1.

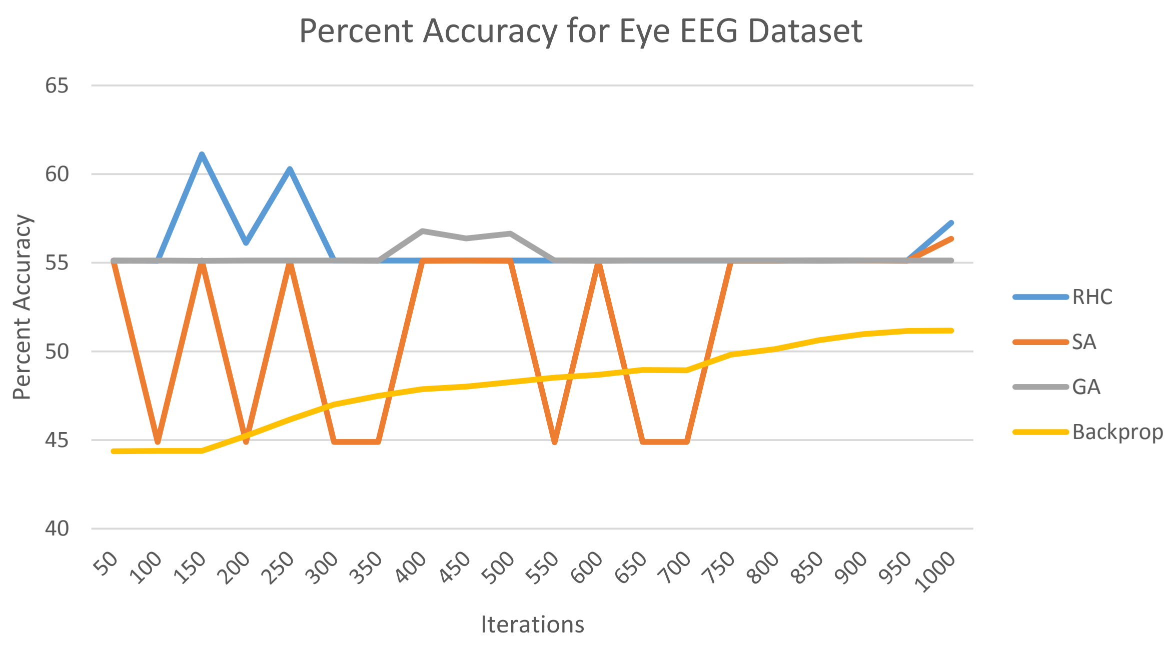


As the name implies, there are only four peaks in the Four Peaks problem. MIMIC excels at problems that do not have many local maxima, so it would make sense that MIMIC would outperform all of the other algorithms in this case. This is because at every iteration, MIMIC reduces the area that it has to look at and focuses on the best area that it can find. Since there are only four maxima, MIMIC can eliminate suboptimal maxima quickly and find the global optima with fewer iterations than the other three algorithms. The only downside is that MIMIC takes significantly longer for each iteration, as it does in each of the other problems. As with N-Queens, Genetic Algorithms incorrectly assumes that the cross of two high scoring candidates would result in a more correct offspring. Randomized Hill Climbing and Simulated Annealing do not perform well on Four Peaks because they can easily get stuck in the other three local maxima and even with random restarts, get stuck in another local maxima. Essentially, when either Randomized Hill Climbing or Simulated Annealing make a wrong turn, they are stuck and cannot get out.

Algorithm	Time (seconds)
Randomized Hill Climbing	0.00003647
Simulated Annealing	0.00003194
Genetic Algorithms	0.004829
MIMIC	0.8229

Conclusion

From this assignment, I have learned about how to use randomized optimization algorithms to find weights for neural networks and how they compare to backpropogation.



The Eye EEG dataset that I used in the Supervised Learning assignment does not perform better with randomized optimization algorithms than it does for backpropogation. I think this is because the underlying distribution of the Eye EEG dataset is too complex for random algorithms to effectively determine the weights and an actual nonrandom, structured algorithm is needed to find effective weights in a timely manner. Given the upward trend of the backpropogation with the Eye EEG dataset, it would be interesting to see the accuracy of Randomized Hill Climbing, Simulated Annealing, and Genetic Algorithms compared to backpropogation with more iterations.

Each of the randomized optimization algorithms have their strengths and weaknesses and excel in certain types of problems. MIMIC seems to do very well in problems that have fewer local maxima. However, each iteration of MIMIC takes much longer to run than for the other three algorithms. Genetic Algorithms do well on problems where it is acceptable to assume that if two candidates that do well on a problem, the cross of those two candidates will result in an offspring that does even better. Genetic Algorithms are faster than MIMIC, but still slower than Randomized Hill Climbing and Simulated Annealing. Randomized Hill Climbing and Simulated Annealing do well on problems where there is large enough attractor basin for the global maximum that randomly restarting will lead you to the global maximum and not one of the local maxima.