Hamza Wahhas

Machine Learning Analysis

4/3/2023

Optimized Learning Analysis

For this project, I've chosen two popular optimization problems. I will perform tests on them using the genetic and simulated annealing algorithms, both of which are available for me to use through the Python library mlrose. More information about this library will be provided in the README.txt file.

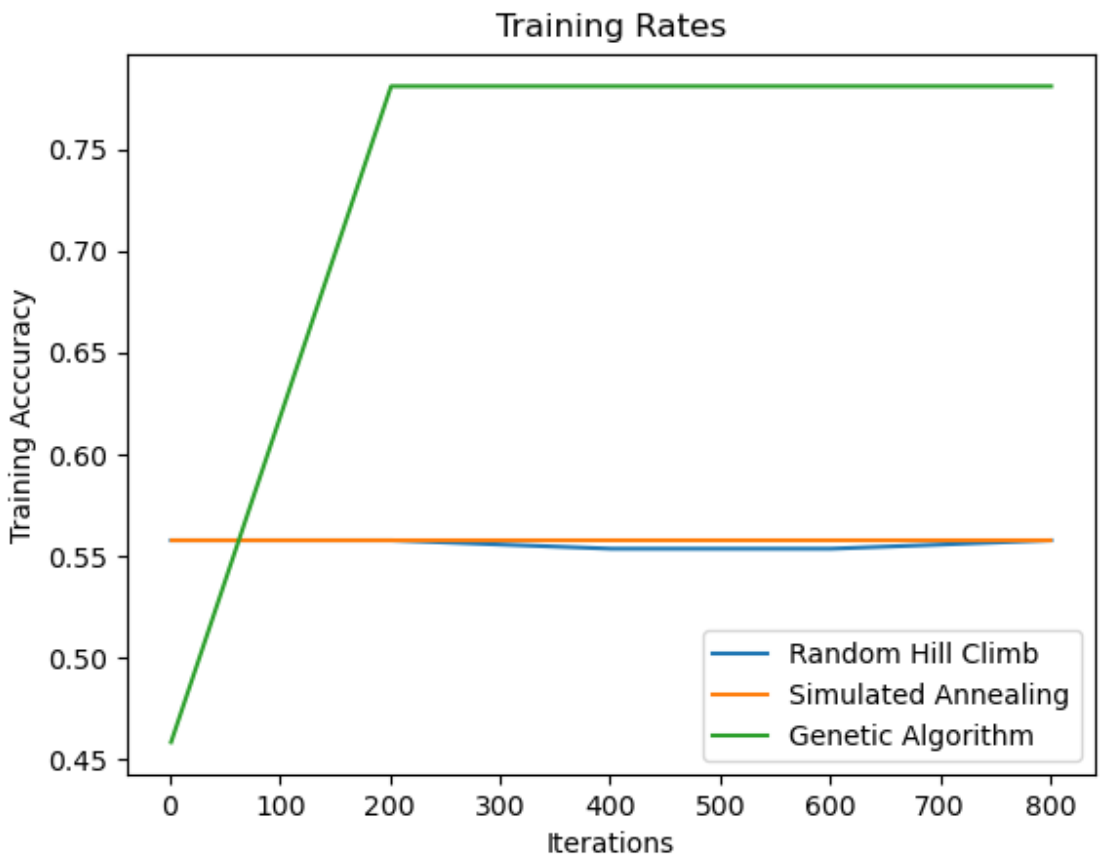
Optimization Problems

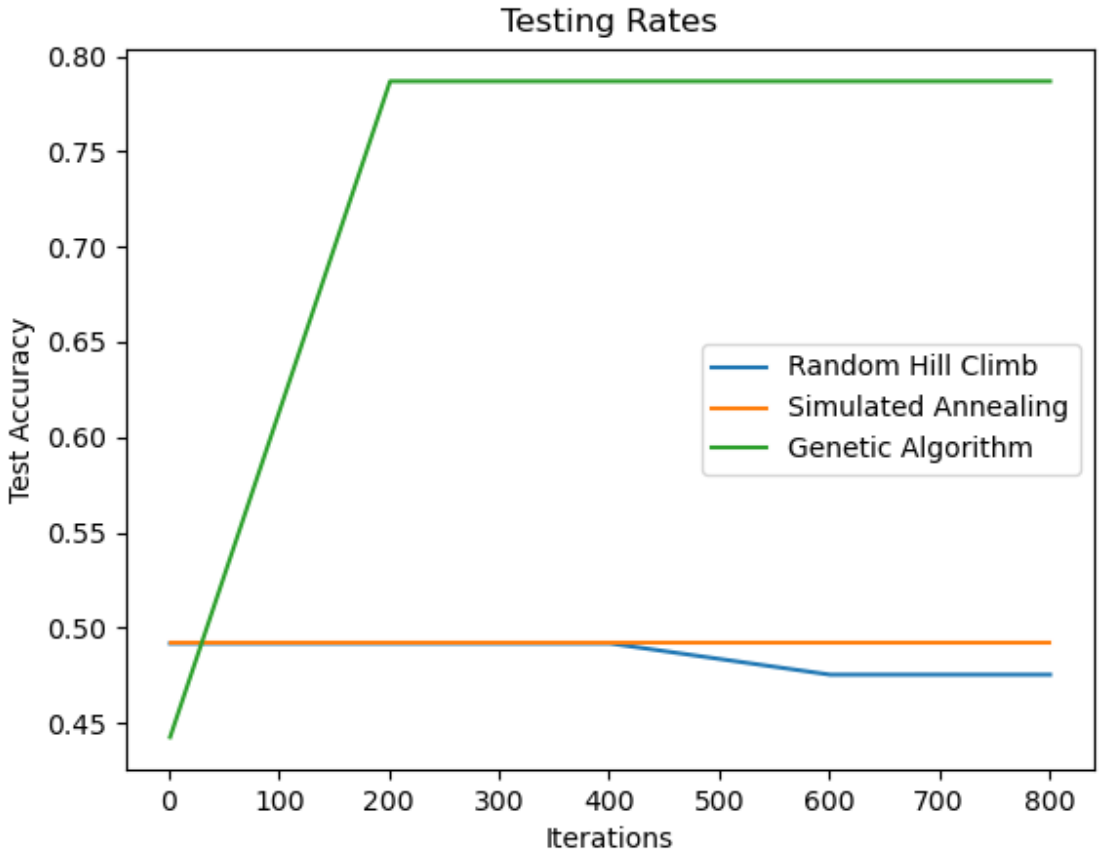
- Traveling Salesperson Problem → The Traveling Salesperson Problem, TSP for short, is a problem based on finding the shortest possible path between a specific list of cities based on the distances between them without visiting the same city twice and returning to the starting city. This problem is a minimization problem, to maximize the number of non-collisions. Much of the code in this algorithm comes from the mlrose website (https://mlrose.readthedocs.io/en/stable/source/tutorial2.html).
- Queens → The 8 Queens problem involves placing 8 queens on an 8x8 chessboard with the condition that no queens are threatening each other. The queen piece follows the traditional rules of chess which allows them to move horizontally, vertically, and diagonally. Because this problem is a simple optimization problem, I expect that all three

algorithms will work well. The code used in this project is also sourced from the mlrose website (https://mlrose.readthedocs.io/en/stable/source/tutorial1.html).

Fitting Neural Networks Using Optimization Algorithms

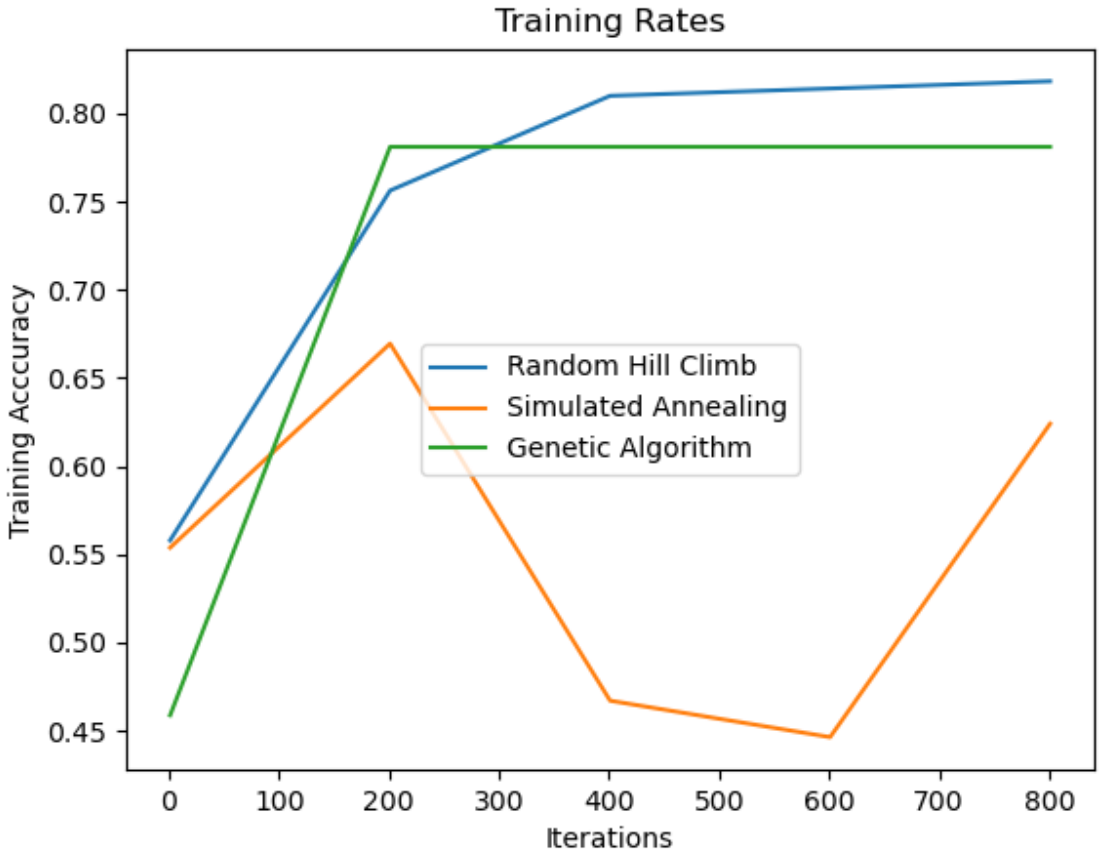
Training and Testing Rates

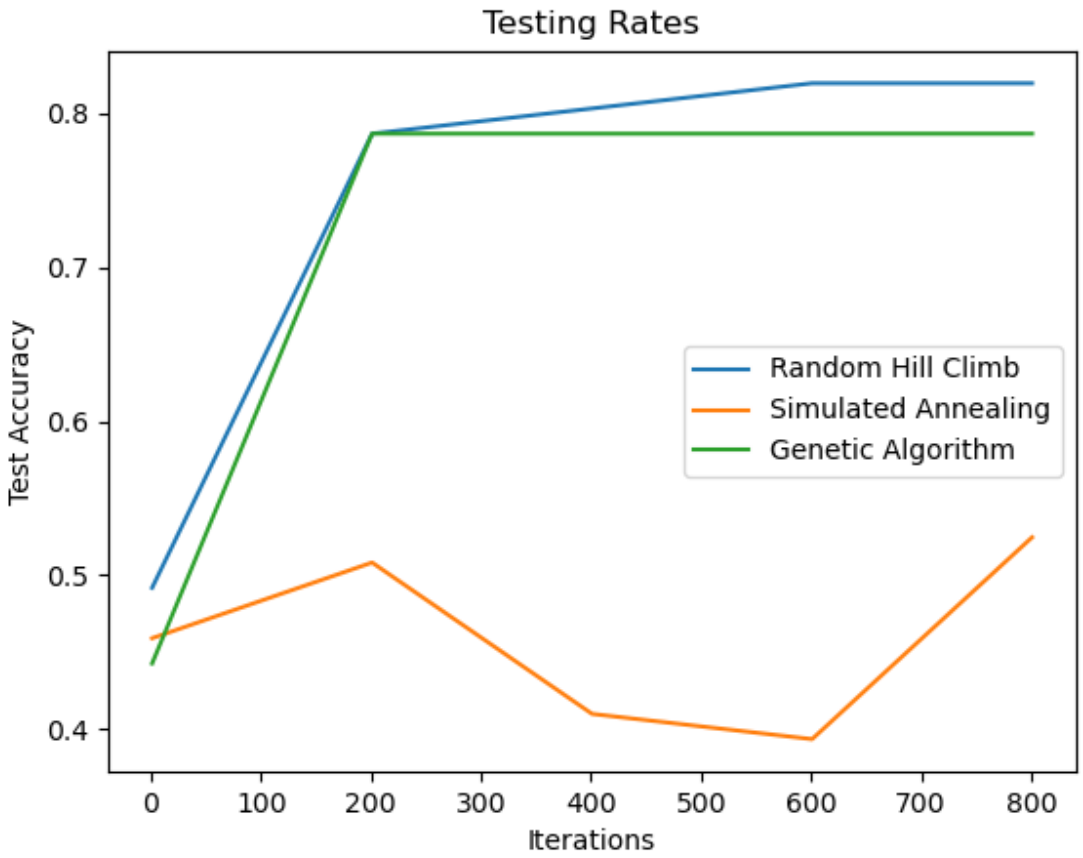




The data set used for this project is the heart disease dataset which takes attributes most commonly found in patients with heart disease along with whether they have the illness or not. When it comes to training, we can use the data to predict whether they have heart disease or not.

I began using the default .0001 learning rate for the neural networks which resulted in a low accuracy rate. I upped the learning rate to .1 which resulted in higher results.


Training Rates
Training Acccuracy
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
100
200
300
400
500
600
700
800
Iterations
Random Hill Climb
Simulated Annealing
Genetic Algorithm



These are the results using the learning rate of .1, which can be seen as having a higher training and testing accuracy when using simulated annealing and random hill climb. Note that the genetic algorithm didn't change which leads me to believe that genetic algorithms will achieve the same training accuracy regardless of the learning rate.

The weights were set at 2 due to all weights that were greater than or equal to 2 being the same accuracy and everything below 2 had a lower training accuracy. The neural networks were run 5 times each at 1000 iterations with values being tracked every 200 iterations. Because the training and testing rates are very similar, it means that the algorithms did not cause overfitting. Typically

if the training accuracy was very high with the testing rates staying low, there's most likely overfitting that is present.

What confused me was how badly the simulated annealing compared to the other two algorithms. The most plausible reason is that it got stuck in the local minimum and never had the chance to get out of that minimum. I expected the genetic algorithm to perform well given the nature of the algorithm to perform well on datasets with little noise like this heart disease set.

Traveling Salesperson Problem

Times allotted & Fitness

Algorithms	Fitness	Time
Simulated Annealing	27.802933495334084	0.0029993057250976562
Genetic Algorithm	28.382100514362897	0.8699824810028076

From these results, we can see that both simulated annealing and genetic algorithms both perform well with the traveling salesperson problem. However, one big difference which should be noticed is the difference in time. The genetic algorithms performed far slower than the simulating annealing algorithm due to how genetic algorithms are structured.
It should be noted that the genetic algorithm worked slightly better for this problem because of its ability to work through large spaces quite well. Genetic algorithms also can avoid being trapped in local minima, which is not the case for simulated annealing.

Overall, the choice between these two algorithms may depend on the specific problem at hand, in this case, genetic algorithms.

8 Queens

Times allotted & FItness

Algorithms	Fitness	Time
Simulated Annealing	28.0	<1ms
Genetic Algorithm	23.0	0.11850857734680176

Both algorithms performed well in contrast to each other just as the case with TSP. I didn't expect simulated annealing to perform as well, due to how crucial it is to find the best solution when with simulated annealing, there is a chance that you can get stuck and end up with a solution that isn't the cheapest.

When it comes to time simulated wins the speed race by a large margin although the genetic algorithm is quite slower than the previous one for TSP which is most likely from the search space difference between the two problems. With 8 queens, there are 92 possible solutions while for TSP the solutions can exponentially grow with the number of cities considered, in this case, 8 where the number of possible solutions is ((7*6*5*4*3*2*1)/2 = 2520. This also carries onto simulated annealing which performs faster with 8 queens than with TSP due to the large search space slowing the process down.

Conclusion

In this project, I tested two popular optimization problems - the Traveling Salesperson Problem and the 8 Queens problem - using genetic and simulated annealing algorithms available through the Python library mlrose. I also fit neural networks using optimization algorithms and experimented with different learning rates to achieve higher accuracy rates on the heart disease dataset.

The results showed that both genetic and simulated annealing algorithms performed well in solving the optimization problems, but the choice between them may depend on the specific problem at hand. The genetic algorithm worked slightly better for the TSP problem due to its ability to work through large spaces, while simulated annealing performed better for the 8 Queens problem.

Overall, the project demonstrated the effectiveness of these optimization algorithms and the importance of considering the specific problem and search space when choosing an algorithm.