Machine Learning Assignment 2 - Randomized Search

Keertana Subramani

I. Introduction:

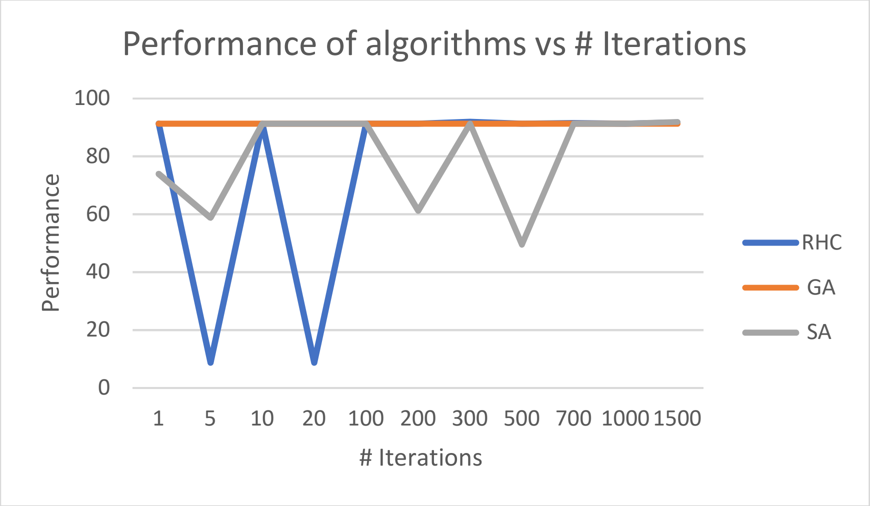
The three datasets/optimization problems I picked are: 1) The Abalone Dataset on which Neural networks were trained to optimize weights 2) The CountOnes optimization problem through which the strengths of Simulated annealing were shown and 3) The Travelling Salesman problem in which Genetic Algorithm outperforms the other two search algorithms. All three are fascinating problems on which the three kinds of optimization/random search algorithms were performed.

II. Analysis

A. Optimizing weights in a Neural Network: Abalone Dataset

The Abalone dataset contains physical measurements of abalone used to predict their age. In this analysis, we implement randomized hill climbing, simulated annealing, and genetic algorithm to find optimal weights to a neural network that is classifying abalone as having either fewer or more than 15 rings (an indicator of age).

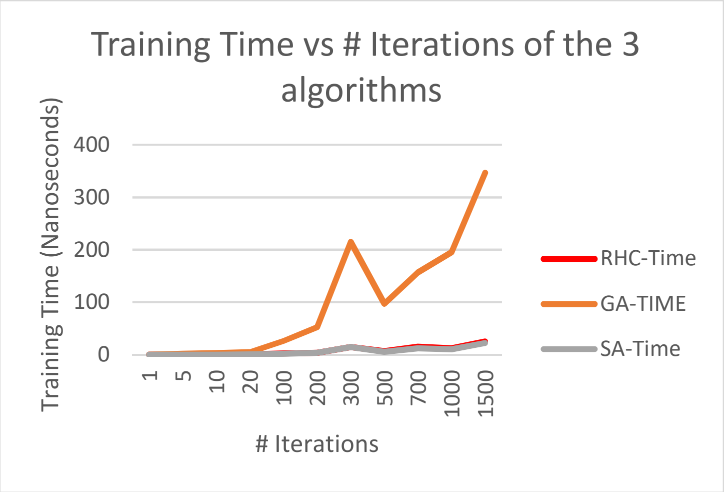
The graph below shows the performance of the 3 random search algorithms with the # of training iterations. The fluctuating performance of RHC for a lower number of iterations could be due to random starts and pure luck – it is possible that with a bad random starting point, the hill climbing algorithm got stuck in a local optima, unable to escape. However, as the number of iterations increases beyond a certain point, the hill climbing algorithm is able to give consistently high performance as it is able to always reach the best possible optimum for neural network weights, given that it has enough iterations and random restarts to overcome any bad luck in some of the restarts that may have led it to a bad local optimum.



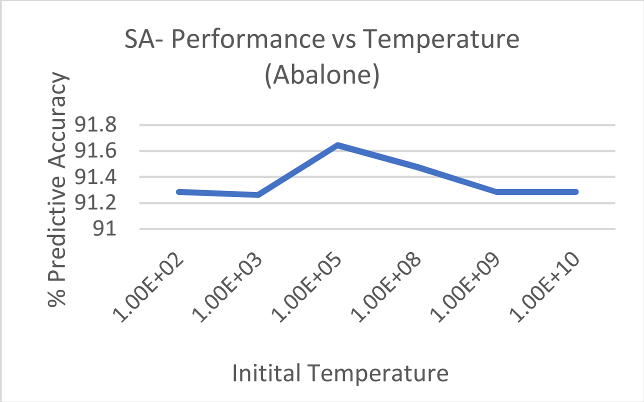
Similarly, for simulated annealing, as the number of training iterations increases, it is able to give high performance since, over many iterations, with enough random restarts, and with a suitable temperature schedule, it is able to gradually move out of any local optima and converge in the direction of the global optimum. In general, Simulated annealing and RHC tend to do well on problems that have a large attractor basin so that regardless of random restarts, they are able to end up at the global (instead of local) maximum.

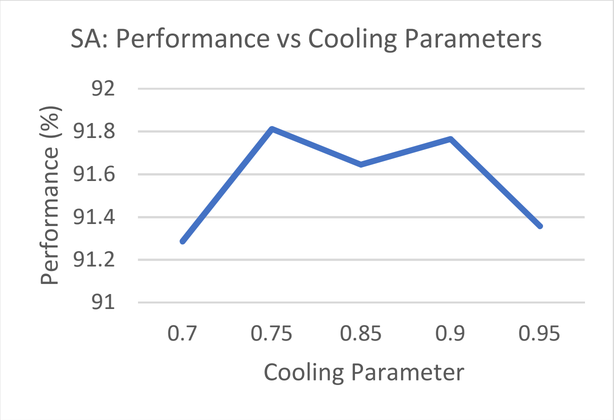
In this dataset, Genetic Algorithm performs consistently well, which could be because it does not depend on random restarts but rather depends on the population size, fitness metric (the objective function to find optimal weights), number of mates and mutations. All of these hyperparameters, and the fact that genetic algorithms repeatedly generate new populations every generation, allows them to explore the search space more exhaustively (At the cost of higher training time), and helps them reach a global optimum.

For all three algorithms, as shown in the graphs, the training time drastically increases as we increase the number of iterations. This is because, with a greater number of iterations, the algorithms have to restart and run through all the steps many times over. Out of the three algorithms, genetic algorithm takes nearly ten times more time than simulated annealing and RHC, both of which take comparable time. Simulated annealing does not take much time since it is only a special version of the RHC problem, with an added probability that allows it to explore worse off solutions close the local optima. The RHC function itself is very a very straight forward algorithm that looks for an optima in the neighborhood of a random start point. The reason the genetic algorithm takes a lot more training time is because it first testing the fitness at each node (weight) by evaluating the neural network performance, then deciding which ones to cross over or mutate, and using selection rules for a similar evaluation for every subsequently generated population in each iteration, which can be very time consuming.



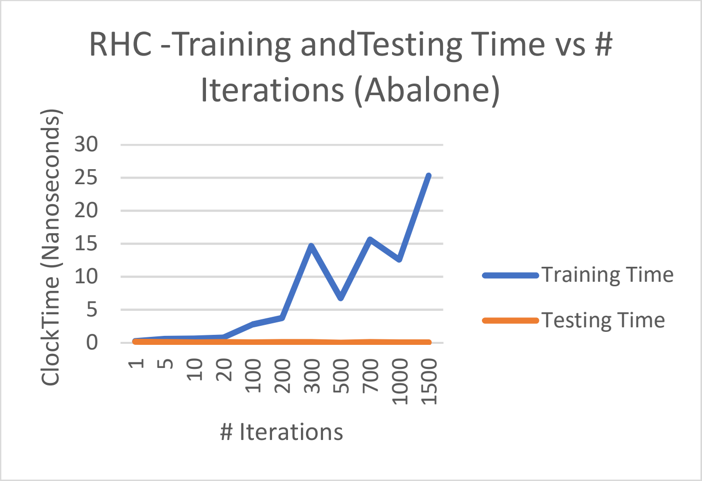
Below, we see that simulated annealing has a decent performance, especially as the number of iterations increases which assures with high probability that even random restarts will be able to explore neighborhoods around local optima to find a global optimum. Algorithm performance peaks at a medium starting temperature (around 10E5) cooling coefficient of around 0.9. This is because a moderate starting temperature provides room for the algorithm to move worse solutions early on, thus allowing it escape potential local optima. A relatively high (but not too high) cooling coefficient ensures that the temperature schedule is scaled down at a pace where the algorithm is allowed exit from local optima for a reasonable span of time before it zones in converges to finding the optima in a narrow search space.

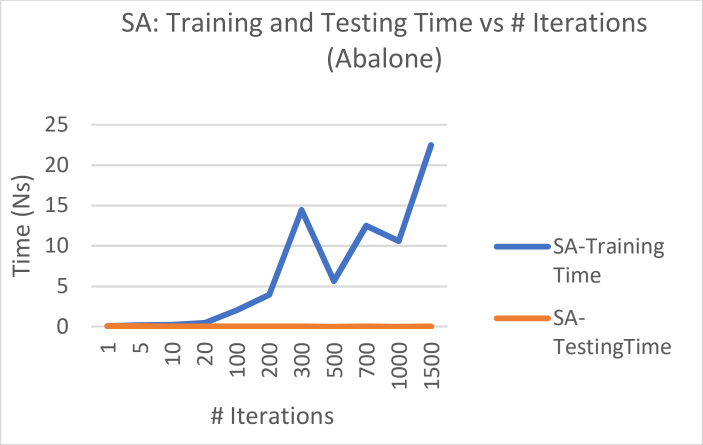


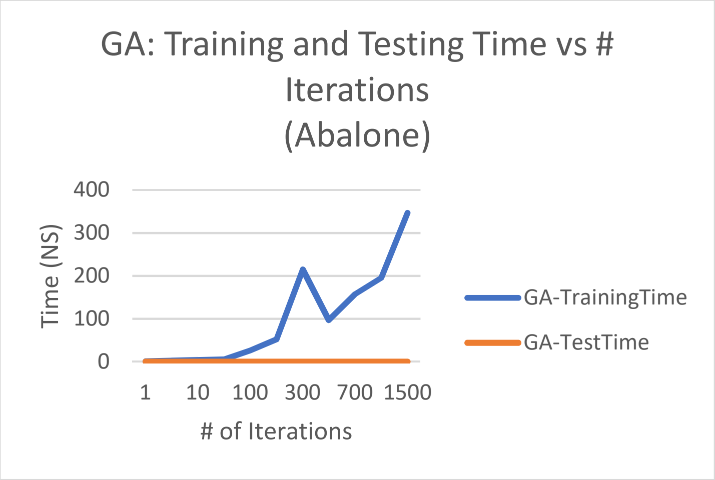


Training vs Testing Time on different Algorithms

(Abalone Dataset)



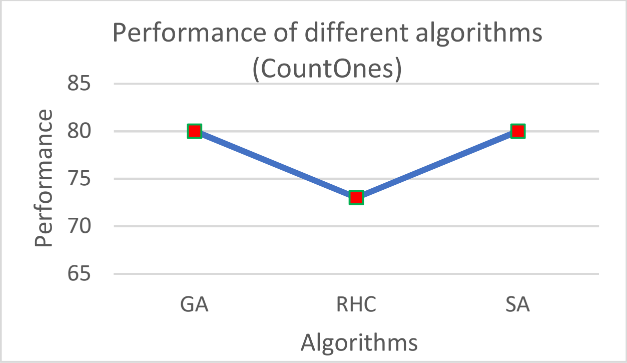


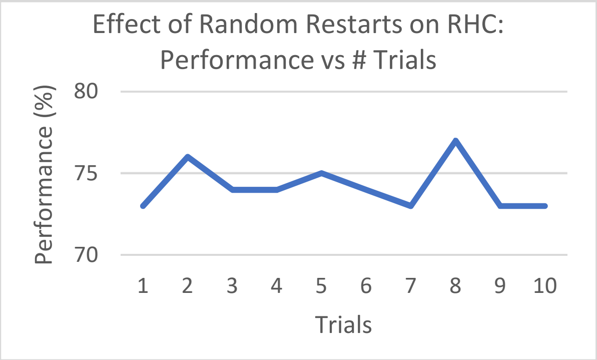


For all 3 algorithms we see a similar trend between training and testing time. While testing time remains very low and roughly constant, the training time goes up with the number of iterations as the algorithm has to restart and go through all its steps for every iteration. Of the three, Genetic Algorithm takes the longest time to run. Performance however, peaks at a certain number of iterations for each algorithm when the number of iterations is high enough to allow it to exhaustively search and optimize over the entire search space – increasing training iterations beyond this doesn't improve performance.

B. CountOnes: Strengths of Simulated Annealing

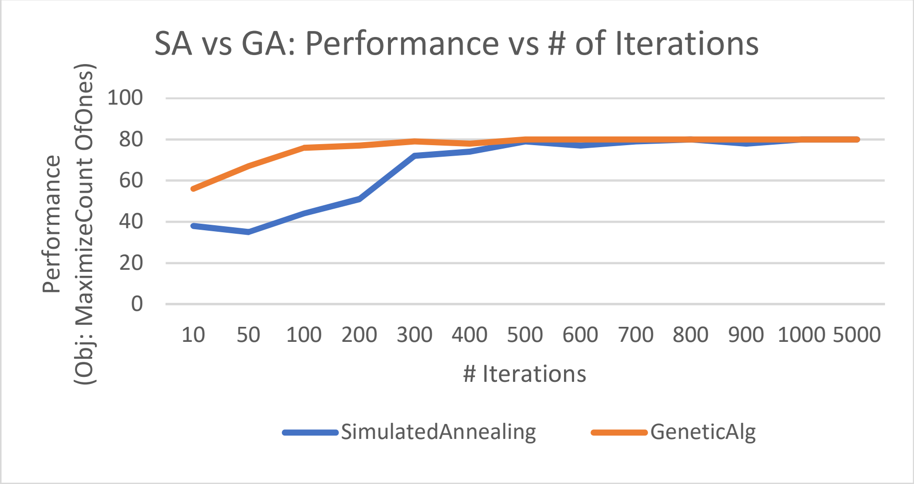
The CountOnes problem is an optimization problem with the objective of maximizing the number of consecutive ones in the dataset. Randomized hill climbing performs the worst in this problem, though it has slightly different rates of performance depending to random restarts each time the algorithm is run, as shown below, over 10 different trials of the RHC algorithm with 300 iterations on this dataset.

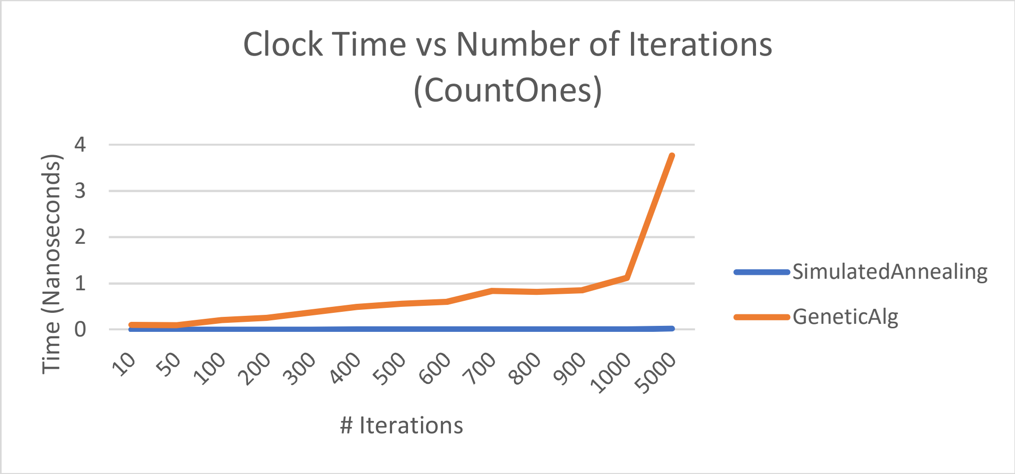




For this optimization problem, we observe that the genetic algorithm performs the best initially, but as the number of iterations increases, is matched in performance by the Simulated annealing algorithm. Initially, with few iterations the SA algorithm has the opportunity to explore a small local search space for optima, but with random restarts at every iteration and increasing number of iterations, is able to expand search to a much larger, and eventually, the entire search space to converge on the global optimum.

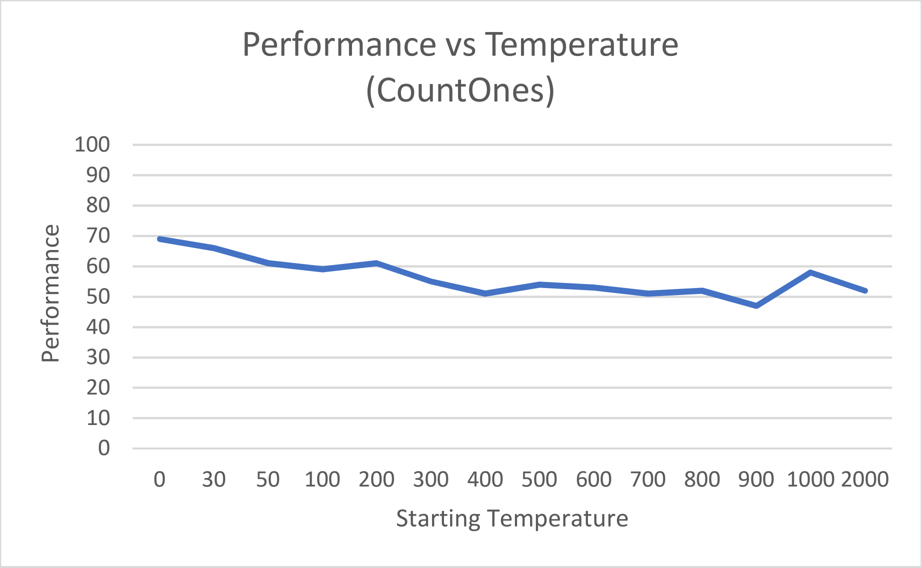
Simulated Annealing is best suited to this optimization problem because while it performs equally well compared to a genetic algorithm with a high enough number of iterations, its training time is much, much lower than that of a genetic algorithm. In general, on large datasets, simulated annealing can find us a much quicker global optimum than genetic algorithm while giving equally high performance.



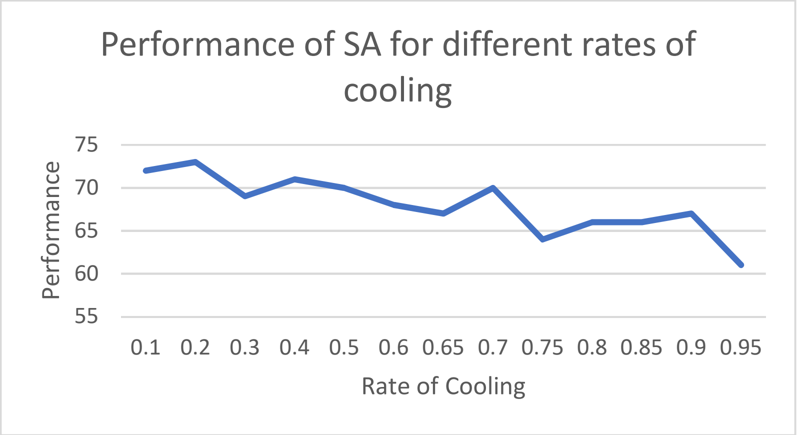


Simulated Annealing: We can understand the strengths of Simulated Annealing using the Count ones problem. Simulated annealing is a probabilistic technique for approximating the global optimum of a given function that is similar to hill climbing but has a probability (that progressively decreases according to its initial temperature and cooling parameter) of moving from a local maxima to a worse solution in hopes of finding a global optimum (if one exists that is better than the current optimum).

For the graph below SA is run over 200 iterations and 0.95 as cooling coefficient with varying initial temperatures. Very high performance at a temperature 0 could be a chance result where random restart picked a point that resulted in a comparatively high local optimum. As the temperature increases performance peaks a moderately high temperature (about 200), which allows for potential changes in directions towards worse solutions, but not for long enough that the algorithm turns random. Higher temperatures result in worse performance as the algorithm becomes more and more random, as it allows a move towards worse solutions more frequently and for longer than a run with a lower initial temperature. In contrast, very low starting temperatures very rarely allow the algorithm to escape a local optimum, essentially turning the Simulated annealing solution to a hill climbing-like problem.

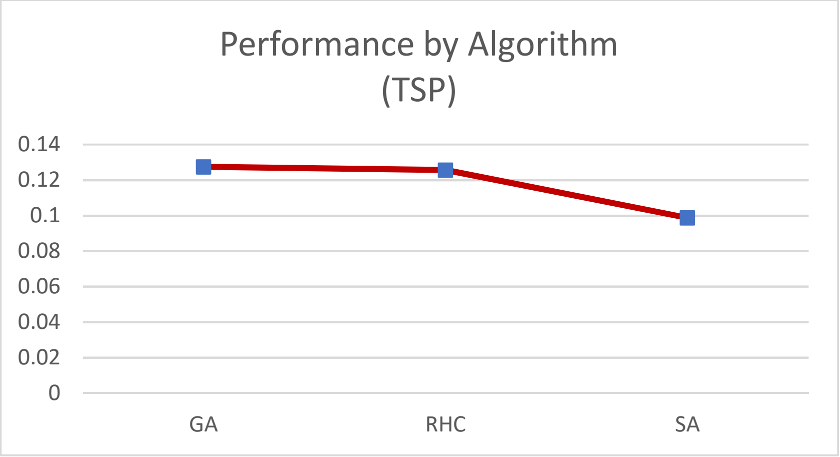


As we increase the rate of cooling, the algorithm cools down quicker or converges to a local optimum quicker, with a rapidly decreasing probability of escaping local optimums to find a global optimum, thus its performance decreases, as shown below. At very high cooling rates, the temperature schedule is forced to rapidly decrease, thus drastically decreasing chances for the algorithm to escape local optima and explore (worse-off) neighbors in the search space, in order to ascend the gradient to another, better optimum. This is why SA's performance declines at very high rates of cooling.

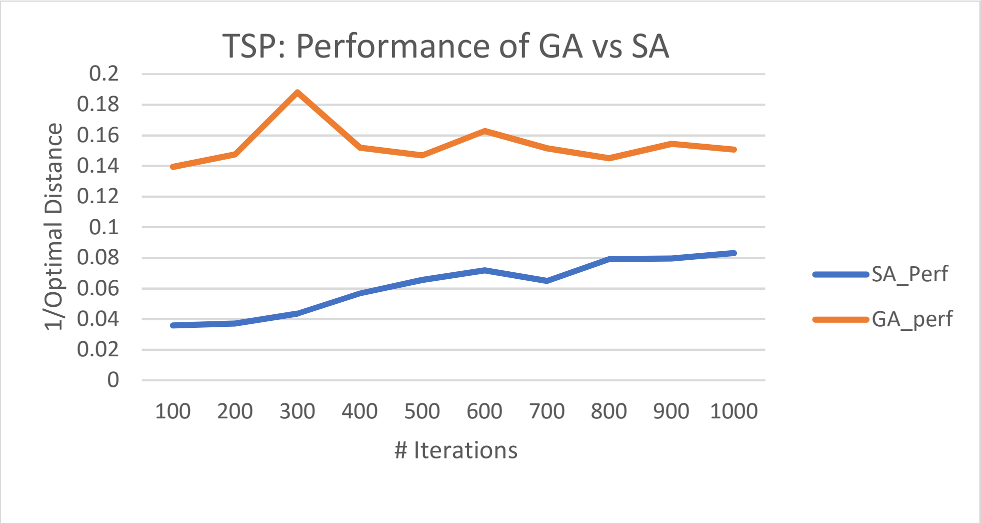


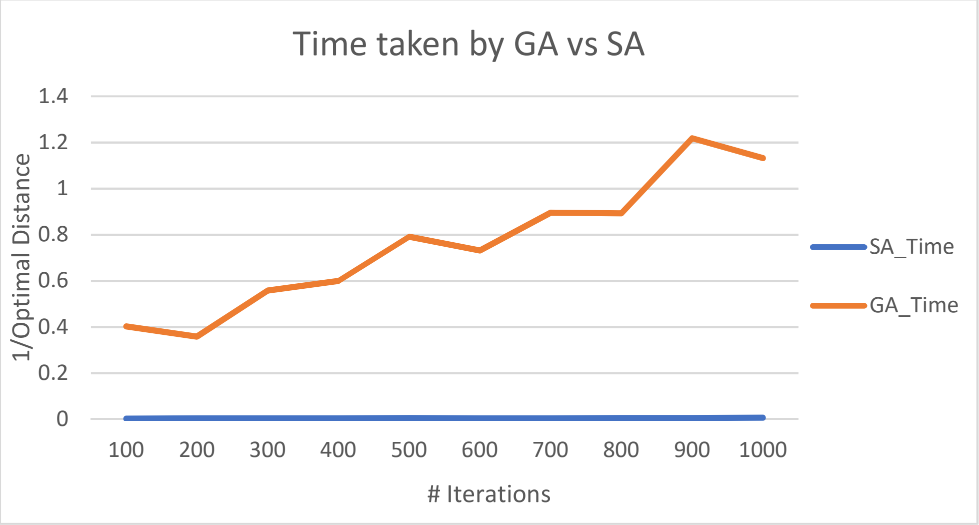
C. Traveling Salesman Problem: Strengths of Genetic Algorithm

Given a list of cities and the distances between each pair of cities, the traveling salesman problem seeks to find the shortest possible route that visits each city and returns to the origin city. The graph below shows the performance of each search algorithm in finding the optimal route, where performance = 1/shortest distance found. We see that Genetic algorithm performs best and Simulated Annealing the least well.



This can be because, since the search space is usually very large (and increases exponentially with the number of cities), randomized hill climbing has a high chance of getting stuck at a local optimum and simulated annealing might also not be able to move far enough from local optima to reach a globally optimal solution, and so may get stuck after reaching a worse-off solution at the neighborhood of a local optima, which worsens the algorithm's performance, especially when the number of iterations is low. Genetic algorithm, that evaluates the fitness of the current "population", accordingly mates and mutates the current population to make new generations, can solve this problem much more efficiently to reach the global optimum, since it doesn't depend on gradient based hill climbing or random restarts.

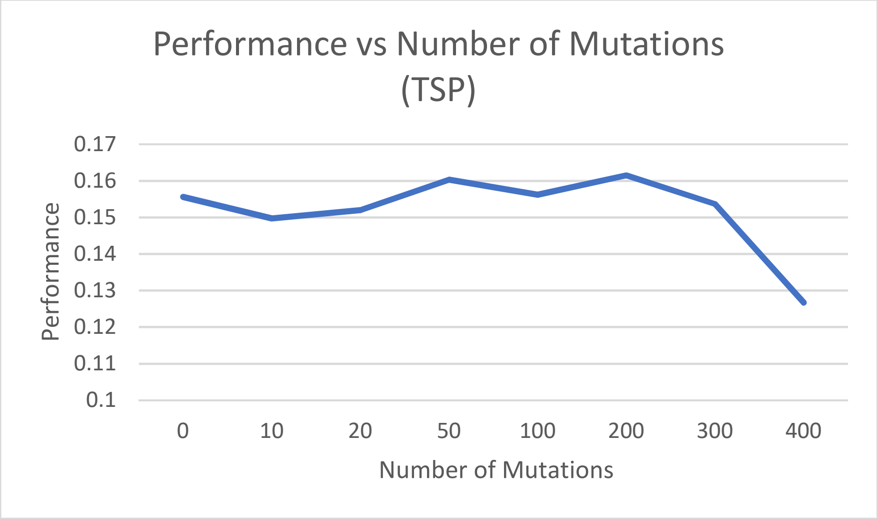




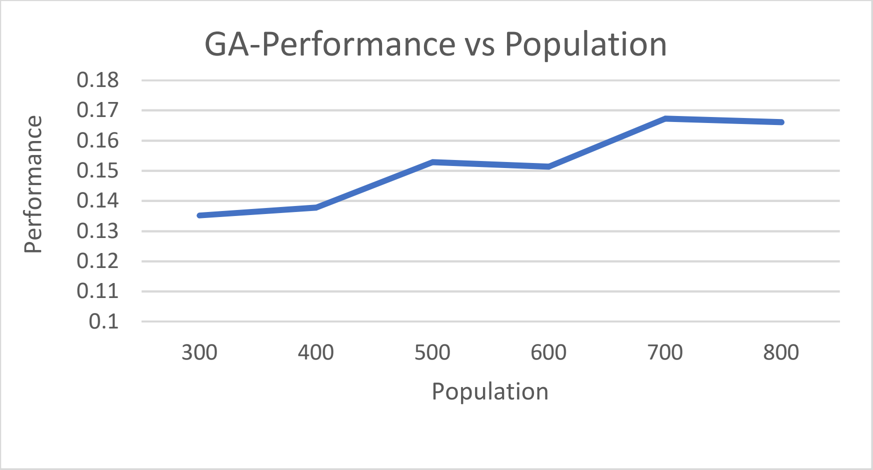
As we see above, even though genetic algorithm takes a lot more training time as the # of iteration increases, its performance is also consistently and vastly better on the Traveling Salesman problem which makes it the best algorithm to evaluate the optimal route. Let us explore why.

Genetic Algorithms: Genetic Algorithms are a class of stochastic search strategies modeled after evolutionary mechanisms and use several hyperparameters such as population size, number of mates, and mutations to select the most "fit" individuals in the population , cross them or mutate them, to generate new populations whose fitness (evaluated by the objective function) are also subsequently maximized. This random search algorithm does especially well on the travelling salesman problem which has a large search space.

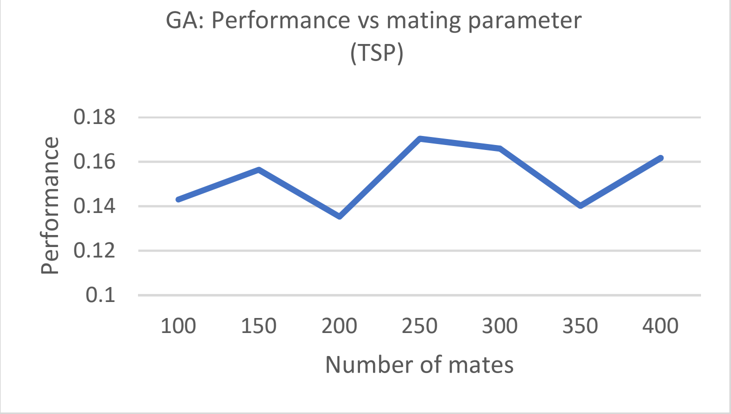
First, as the number of mutations increases, performance first improves, reaches a peak and then decreases. This is because mutation increases the probability of random features being added or changed to the existing population. Low amounts of mutation can help the problem explore new search spaces for optima, while very high levels of mutation can make the optimization process random and decrease performance.



As the size of the population increases, so does the potential search space for the algorithm, and the probability that the fitness functions will select for, and yield a solution closer to the global optimum. Larger the initial population, larger will be the size of the mating pool and the selected “fit” individuals to choose the best ones from.



As mating increases, performance fluctuates with many local maximas. With mating, Every two parents selected from the pool will generate two offspring (children). However, children will have the same (recombined) characteristics that are subset of its parents’, including their drawbacks, without any new addition of features outside of the chosen “fit” subset of parents. This may result in worse performance if some important (optimal) features were somehow lost in the selection/mating process, or if the number of iterations is too small. Small levels of mutation can help improve performance in such cases to bring in new features and potentially overcome parents’ drawbacks.



III. Conclusion

In conclusion, from optimizing weights for a neural network in the first classification problem (The Abalone dataset) using RHC, SA and GA algorithms, we identified that GA takes the highest training and testing time as it has multiple steps at every iteration. Performance of both SA and RHC increase as the number of iterations increases. We then exercised the strengths of Simulated Annealing as a search algorithm with high accuracy and a very quick training time in the CountOnes Optimization problem. It does especially well as the special case of the simple randomized hill climbing algorithm with a finite probability of being able to explore worse-off solutions in the neighborhood of local optima, which, with increased iterations, helps it potentially reach a global optimum. Finally, we explored the strengths and weaknesses of Genetic Algorithms on the travelling salesman problem where it outperforms other algorithms due to its versatility, and complexity such as fitness-based selection, mating and mutation, that allows it to search extensively without getting stuck at local optima or navigating to potentially worse solutions.

IV. References

1. https://en.wikipedia.org/wiki/Travelling_salesman_problem

2. https://en.wikipedia.org/wiki/Simulated_annealing

3. http://www.obitko.com/tutorials/genetic-algorithms/parameters.php