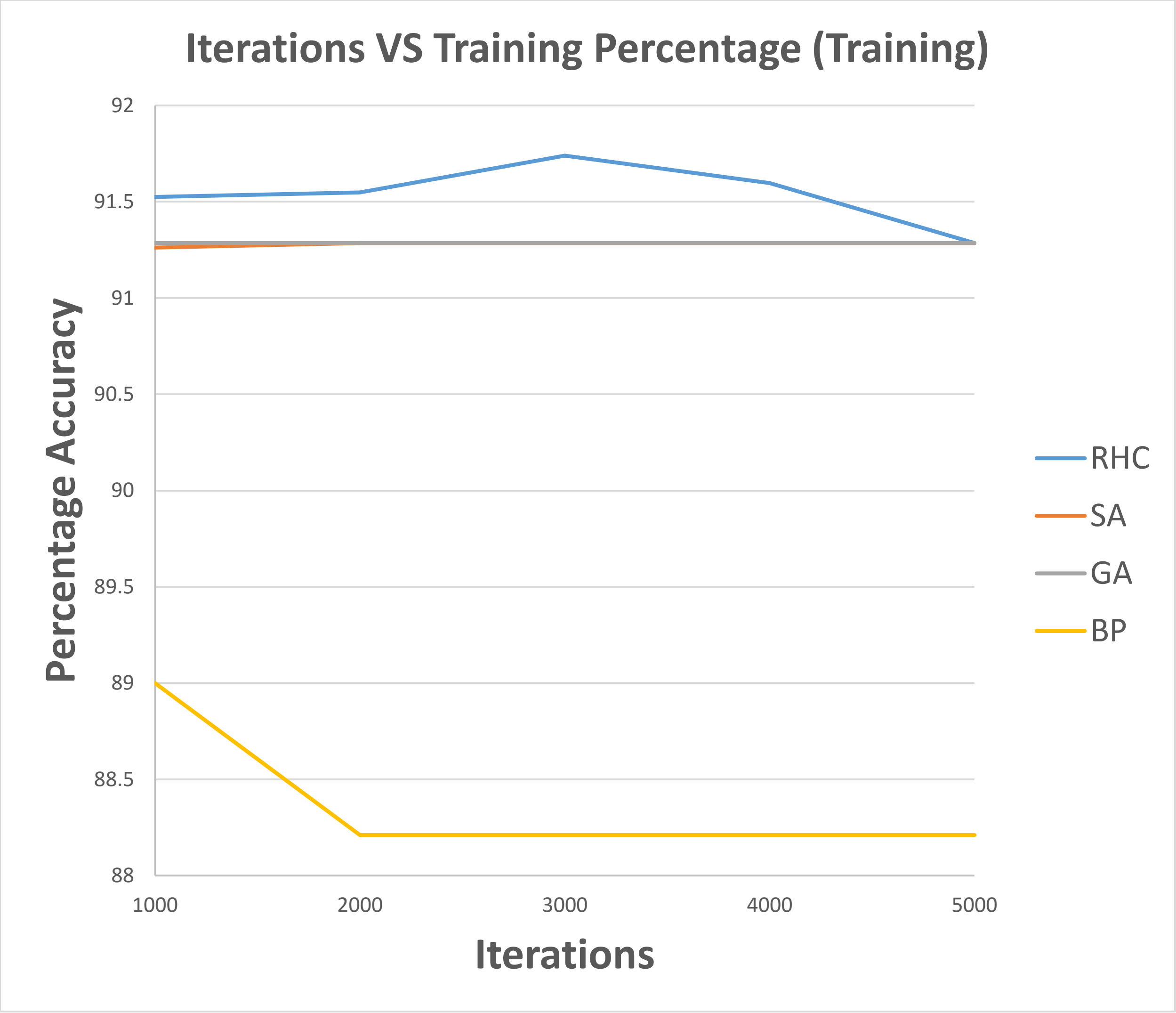
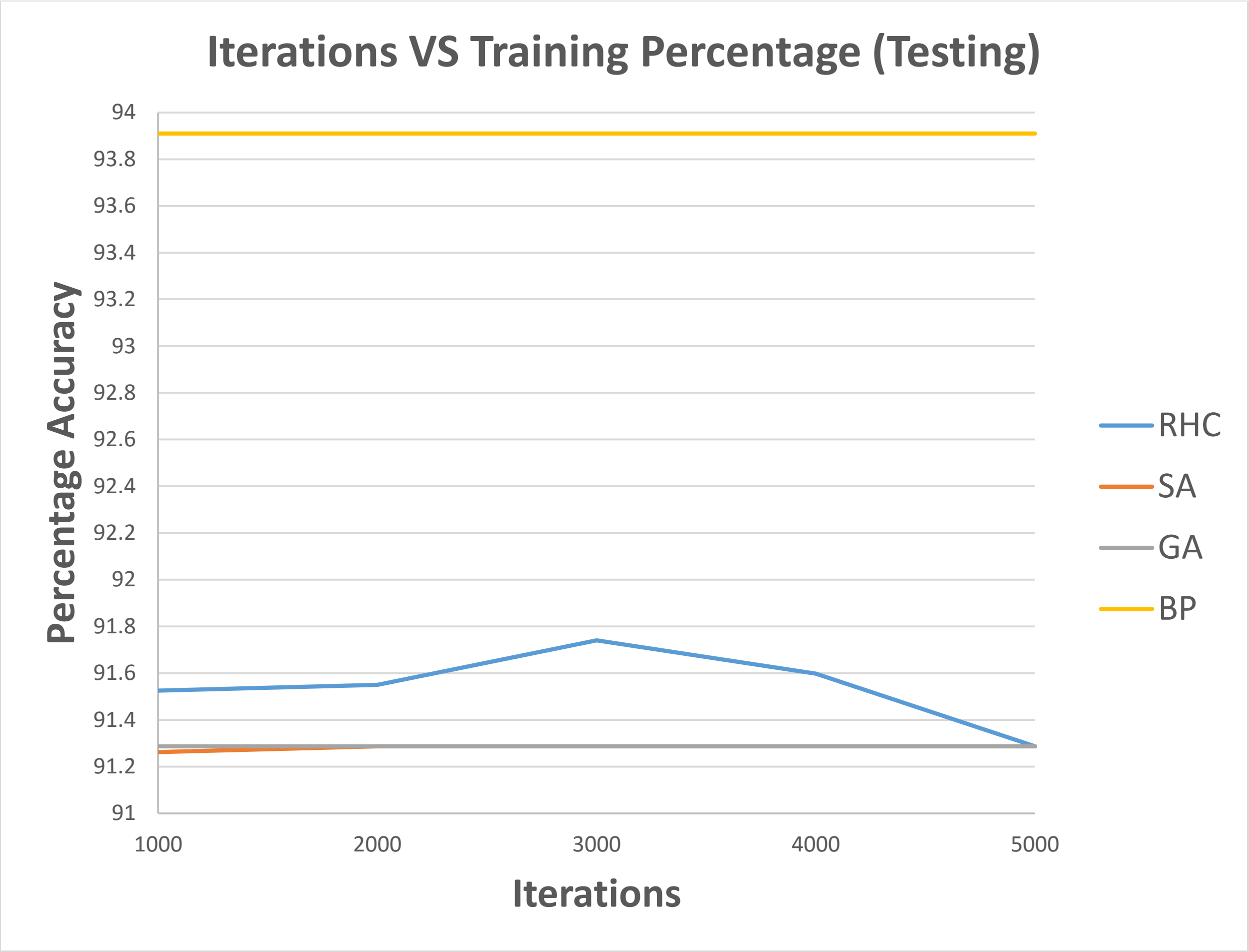
Abhishek Deo
CS 4002

Randomized Optimization Assignment Two

Part 1:

- **Datasets**:
  - The dataset used for this assignment is the same training and testing dataset used for assignment 1. I chose to use the solar flare dataset because of the attributes and intrinsic properties. There are 10 total attributes each of which are multivariate rather than binary and there are also 3 potential classes to identify a solar flare instance that are determined by these 10 attributes. (Also Back Propagation results are the same results that occurred in project one with this dataset.)
- **Algorithm Analysis**:





- Both the training and testing datasets for solar flare was run concurrently with all the random optimization problems updating the weights and varying the iteration count. I ran backpropagation individually and also varied the iteration count and updated the weights. This dataset was tested on the ABAGAIL library specifically through the AbaloneTest.java file (Only difference being I used my solar flare dataset instead). Because of the nature of random optimization, I ran the test 5 times for each random optimization algorithm at different iterations and then took an average. For the training set, Simulated Annealing and Genetic Algorithm ran pretty much evenly with each other with the exception of the first iteration count set at 1000. This makes sense due to the factors that influence these algorithms. One being a search space for Simulated Annealing and the other being structure. It is also noteworthy to point out that these algorithms perform well under a discrete set and Neural Networks are continuous. Randomized Hill Climbing showed signs of improvement but after 3000 plus iterations started decreasing in accuracy. This is because of the neighborhood chosen, the poorer it is RHC may run for a long time and never end up anywhere near a maxima. Simulated Annealing can hit a large number of neighborhoods because the weights are continuous. It is possible that it may converge on a bad neighborhood and therefore will be same as RHC at that point. Genetic algorithm had many issues and took the most time to complete because there is no dominant feature. Thus, the crossover function will fail to evolve the genetic strings. Just looking at the graph, as the number of iterations increase, it simply stays consistent and does not improve. Note: I tested Genetic Algorithm at around 30000 iterations just to see if this was the case with this dataset and the result decreased significantly. Also, the dataset does not really favor Genetic Algorithm in the first place due to the number of instances being too small. This also explains why RHC performed well in the beginning because it has a higher chance to succumb to a neighborhood with a high maxima due to a smaller state space. However it isn't the best because accuracy decreases after 3000 iterations at a significant rate. Another thing that affects RHC are the features which are not binary at all. While they provide a reasonable amount of information to determine a class for a solar flare instance, this can prove daunting for RHC to starting falling to a local maxima and getting stuck.

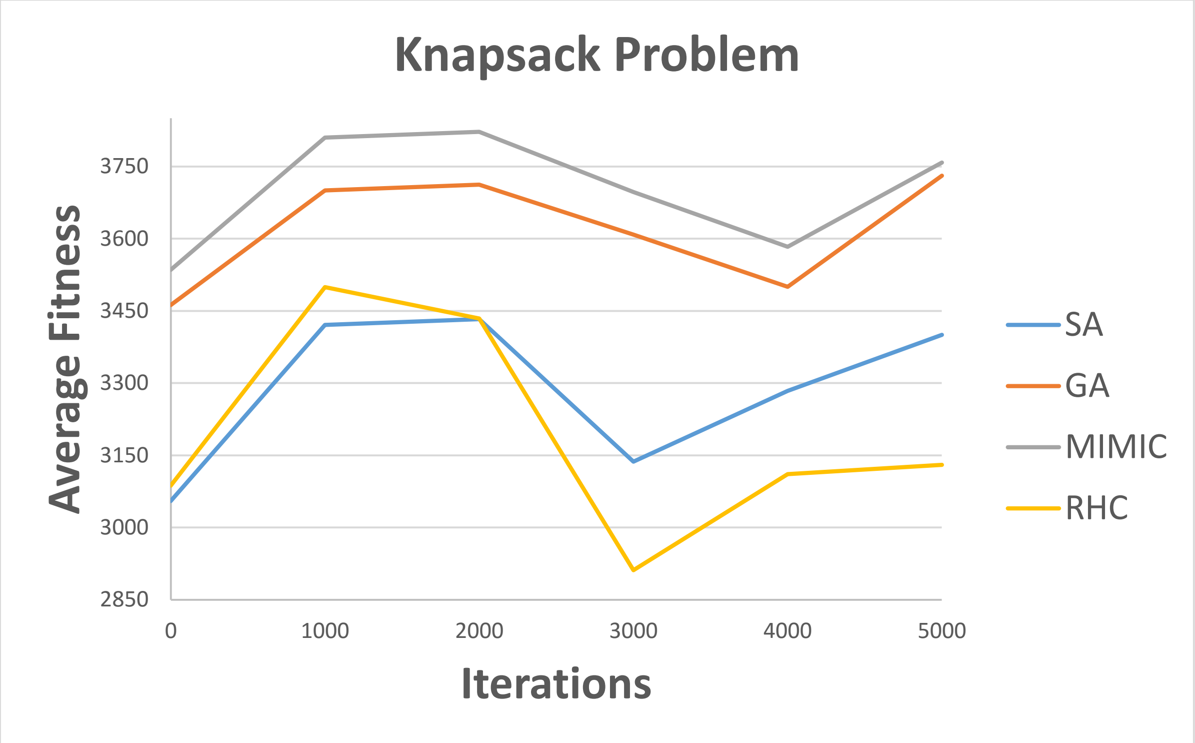
  - For the training set, the BP line is significantly lower than all the random optimization algorithms. It remains consistent after the first 1000 iterations and then converges quickly at 2000 iterations plus. I expected this because of the design of the dataset which features 10 multi-variable attributes trying to classify an instance in one of three classes. The biggest issue was the size of the dataset which did not have enough instances. However, just by seeing the minor changes in the random optimization algorithm it is clear that eventually the accuracy will drop below BP after an x amount of iterations. For the testing set BP already produces more accurate results compared the random optimization algorithms and looking at the flow of the lines, it seems they will never beat BP.
  - A big thing to point out is that there are signs of overfitting only in Randomized Hill Climbing. It happens on both the training and testing set after 3000 iterations. As mentioned above, this is inevitable because of how RHC succumbs to a local maxima. Especially with this dataset and its size plus the way it's constructed. There are not many local maxima for this dataset which means the higher the iteration count the worst for RHC.
  - Time is also another factor in play to help analyze the results. (Note: They are added in the text files provided and are in succession order meaning the first section is for 1000 iterations, second for 2000, etc.) Based on the times, it seems Back Propagation took the fastest time. Among the Random Optimization algorithms, Genetic Algorithms took the longest time. Simulated Annealing took the fastest time but RHC was a close second at a difference of around .8 seconds at each iteration count. Genetic Algorithm however took way too much time at around 205 plus seconds and increasing. However the accuracy was the same as Simulated Annealing. Between Genetic Algorithm and Simulated Annealing it seems Simulated Annealing is better for this dataset because it produces the same accuracy at a much shorter time.

Part 2:

- **Problems**:
  - The problems selected for this section were chosen to complement the Genetic Algorithm, MIMIC, and Simulated Annealing. To start off, for MIMIC, I choose the Knapsack Problem for two primary reasons. First, the Knapsack problem exploits the set of dependencies between points. Second, this problem is a limited by the constraint of resource allocation as well as the fact that the combinations

are dependent on each other. Both Simulated Annealing and Genetic Algorithm perform relatively decent on this problem as well but I believe it will show the strength of MIMIC the most because of the ability to find dependencies between points. For Simulated Annealing, I chose the Continuous Peaks problem because it directly affects Simulated Annealing in the sense of its purpose. There is generally a small domain and large population of local maxima which is ideal for Simulated Annealing. Because this problem converges to a satisfactory maximum with a small dataset that contains many local maxima fairly quickly through Simulated Annealing. Finally, for Genetic Algorithm I chose the Traveling Salesman Problem because this problem is essentially several small subsets of problems. Genetic Algorithm is at its strongest in these situations through the process of evolution which can be used to find a solution. This is under the circumstance that a good crossover function is used. MIMIC and Simulated Annealing will suffer on this problem because of the space size.

- **Knapsack Problem**:

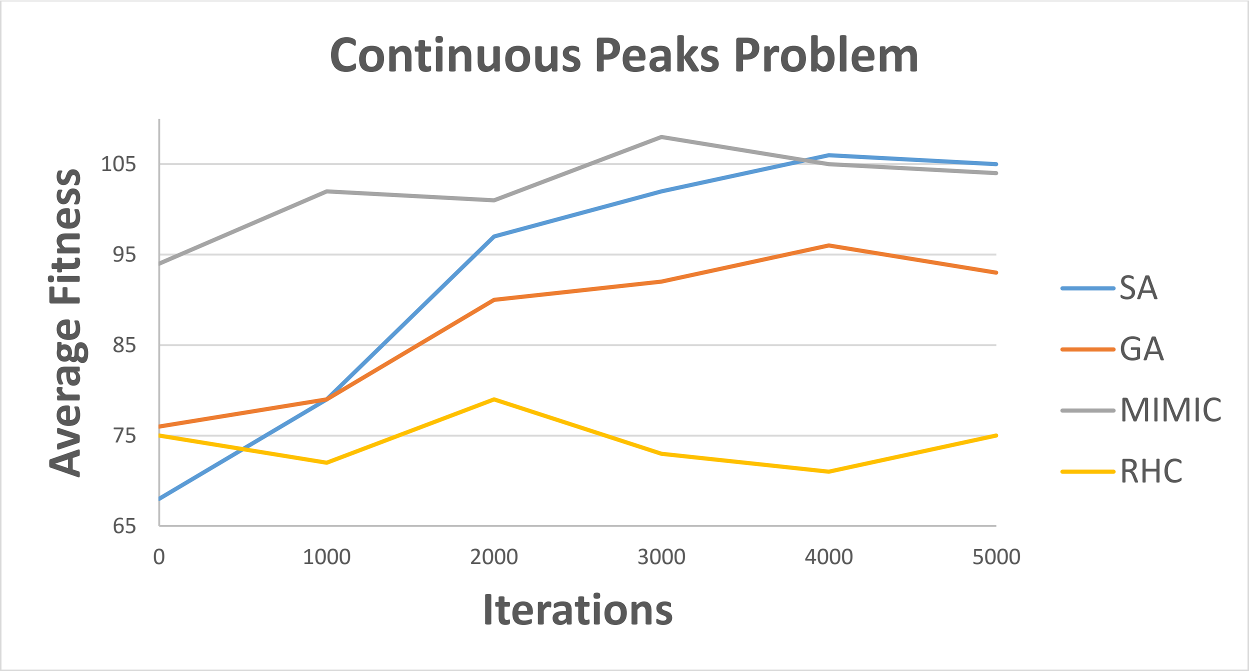


Algorithm	Average Fitness	Average Time
Simulated Annealing	3288.333	0.00437 seconds
Genetic Algorithm	3618.833	0.61248 seconds

MIMIC	3700.833	16.2317 seconds
Random Hill Climbing	3195.333	0.00289 seconds

- **Conceptual Analysis**:
    - The Knapsack problem is looking for a solution to the question: Given a set of items, each with a weight and a value, what combination of items will produce the maximum total value under the constraint that the weight is less than or equal to a given limit in the container? MIMIC is more suited for this problem because it has a structure as well as this problem. MIMIC ran the quickest on this problem as well compared to both Simulated Annealing and Genetic Algorithm. It had a consistent maximum fitness through the set of iterations. Each step in the Knapsack problem affects every subsequent step which gives the problem a structure which the MIMIC algorithm can "flow" on due to its nature. Genetic Algorithm is usually stuck a lower maximum compared to MIMIC because it breaks down the structure and then uses evolution to find a solution. Another factor that affects Genetic Algorithm is the crossover function. Simulated Annealing suffered the greatest simply due to the search space. It does converge, but generally on a local maximum and eventually it acts as Randomized Hill Climbing as the number of iterations increase.
- **Analysis Based on Results**:
    - MIMIC has the largest average fitness vs training time values. As stated before it remains consistent throughout the iterations. Genetic Algorithm ran much quicker than MIMIC but there are instances where it never reached its maximum fitness within the iteration count. Simulated Annealing training time was small but never reached a maximum fitness value. Overall, Genetic Algorithm is quicker but results come late. MIMIC may be slower than Genetic Algorithm but not by much and results come very early. Regardless of how quick Simulated Annealing is it doesn't produce proper results due to acting like Random Hill Climbing.

- **Continuous Peaks Problem**:



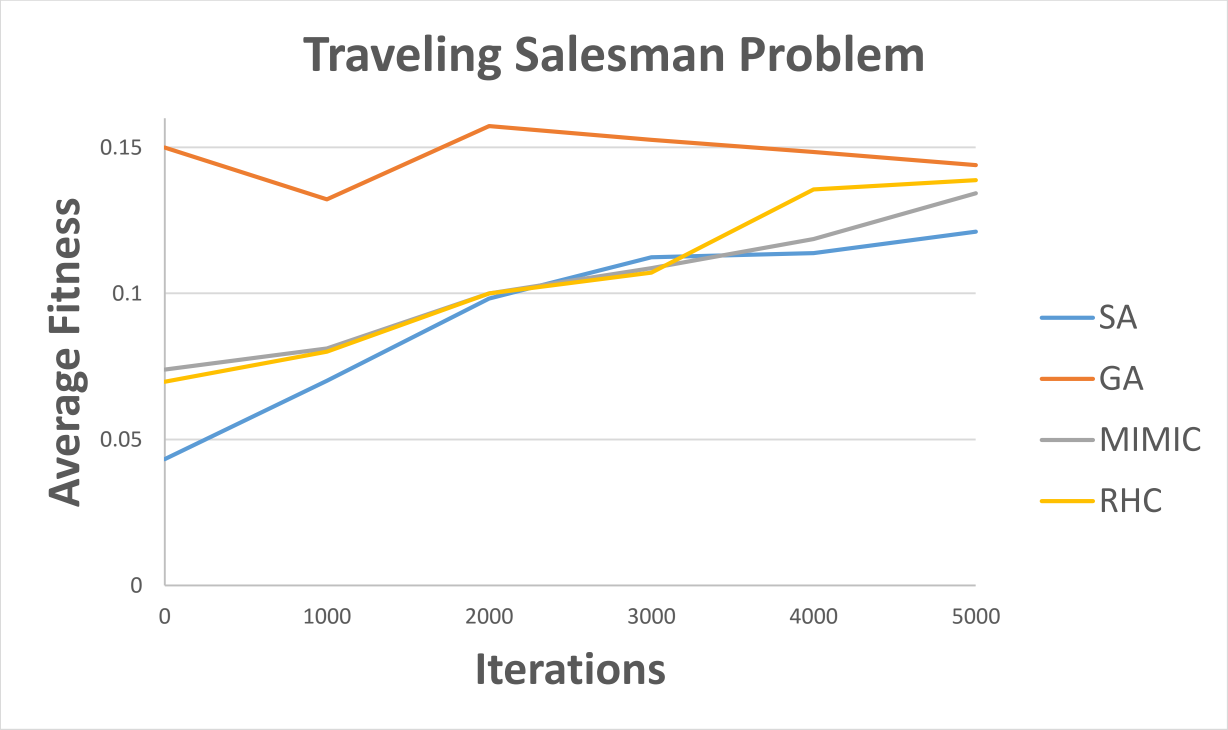
Algorithm	Average Fitness	Average Time
Simulated Annealing	92.83	0.00421 seconds
Genetic Algorithm	87.67	0.5332 seconds
MIMIC	102.33	18.1173 seconds
Random Hill Climbing	74.17	0.002011 seconds

- **Conceptual Analysis**:
  - This problem's goal is to the find the number of continuous bit strings with a reward for having either a greater number of continuous bits set to 0 or set to 1 called "T". All three main algorithms performed better than Random Hill climbing after the first 1/3 of the iterations. Simulated Annealing performed the best for this problem. This is mainly due to the fact that the search space for this problem isn't fairly large and that random jumping on a small search space performs better than on a large space. MIMIC did relatively well on this problem as well by finding the structure of the problem, however it converged too quickly compared to Simulated Annealing. The maxima was found in the first few hundred iterations and remained consistent for the rest. This is the case for around the first 4 iterations sets (up to 4000 iterations). So for this section in terms

of time and ability, MIMIC seemed to perform better than both Simulated Annealing and Genetic Algorithm. However, once we hit 4000 iterations plus, Simulated Annealing outperformed MIMIC. This implies that MIMIC is good up to this point relative to Simulated Annealing. This is because that the higher then number of iterations, the more chances Simulated Annealing has to jump around randomly and find more maxima which favors the algorithm. Aside from Random Hill Climbing, Genetic Algorithm suffered the worst among the three because the nature of evolution does not favor many local maxima. In addition, the crossover function does not know how to properly account for a reward in this case causing subpar performance and sticking with one local maxima.

- **Analysis Based On Data**:
  - Based on the average time results, MIMIC outperformed Simulated Annealing as well as the other two algorithms. But at around 4000 iterations and more it is clearly shown that Simulated Annealing does better than MIMIC in regards to Average Fitness at the cost of time. So while MIMIC does solve this problem quicker it is only up to a certain point. Quicker also does not mean accurate because Simulated Annealing does significantly better the more iterations you set and returns a better accurate result. (In this case however MIMIC average fitness did beat Simulated Annealing average fitness.)

- **Traveling Salesman Problem**:



Algorithm	Average Fitness	Average Time

Random Hill Climbing	0.14741	0.10312 seconds
Simulated Annealing	0.09313	0.03711 seconds
Genetic Algorithm	0.10520	5.10342 seconds
MIMIC	0.10286	148.53192 seconds

- **Conceptual Analysis**:
  - TSP is famous problem that basically asks given a set of points and distances between pairs of points called weights, what I the shortest path that visits each point once and returns to the starting point? For our case, our fitness is defines as 1/c where c is the cost illustrating that the higher cost paths will show a lower fitness value. As mentioned during the problem statements, this problem is best suited for Genetic Algorithm. Intuitively speaking, this is because this problem tends to be very large in size and is best solved through a dynamic approach (i.e break the problem into subparts and then combine after solving the smaller problems). Genetic Algorithm works the same way because the crossover functions breaks the problem into subparts, finds an optimal solutions for the portions, and evolves to the point where it finds an optimal solution for the whole problem. This can be best described as “the sum of its parts”. MIMIC performed fairly well because it was able to find the structure of the underlying dependencies in the problem. But it isn’t the best because it fails in regards to time because the fitness function is very costly compared to the other algorithms. For Simulated Annealing, one of the biggest weaknesses which is search space size affected the performance of the algorithm. Especially for TSP, there are n factorial amount of paths and even with a high amount of iterations, it will simply take too long for Simulated Annealing to find a proper solution.
- **Analysis Based on Data**:
  - Just by looking the lines on the graph, you can see that three of the four algorithms performed very similar to each other. The odd one out being Genetic Algorithm which performed better than the other three algorithms. You can also notice that in terms of average fitness, Random Hill Climbing performed better than even Simulated Annealing. Which is expected considering this problem is a terrible matchup for Simulated Annealing. Another thing to point out is that Genetic Algorithm and especially MIMIC average time was significantly much higher than Random Hill Climbing and Simulated Annealing. However the results between Genetic Algorithm and MIMIC for average fitness were fairly close. So while MIMIC does fine in terms of average fitness, it almost takes by a factor of 30 or 40 times more time for MIMIC to complete and return a similar fitness value.