Tyler Bobik
Project Due Date: 10/16/2016
CS 7641
Student ID: 903202939

Project 2: Randomized Optimization

Part 1: Comparing Randomized Optimizers to Back Propagation Neural Networks

For the first part of this assignment I decided to use my Indian pima diabetes dataset to attempt to find good weights for my neural network using, randomized hill climbing (RHC), simulated annealing (SA) and genetic algorithms (GA). Note, I will use these abbreviations throughout this assignment.

I did my analysis to find the best weights by looking at the test accuracy vs training accuracy being trained on the whole training set. This data set was split into a 70% train set and a 30% test set. I thought the Pima Indian Diabetes dataset was interesting because it has some noise in it, which I proved in the last assignment while additionally having relatively a small amount of attributes and instances. I was curious to see if any of these algorithms would be thrown off by it. In order to find what algorithm was suited to find the best weights I thought it would be interesting to compare how each algorithm did with respect to their accuracy, number of training iterations while looking at the time each algorithm took. I also used the same number of hidden layers and values as I did in the previous project.

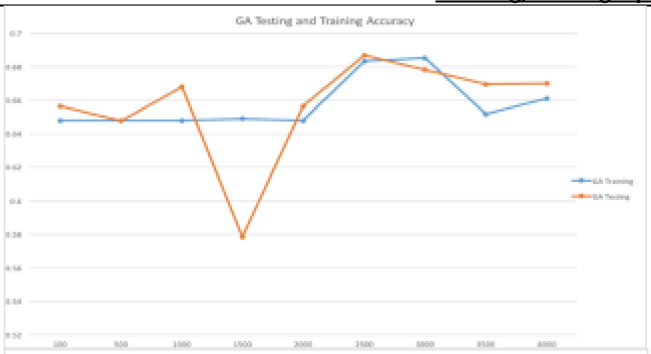
Parameters of Tested Algorithms

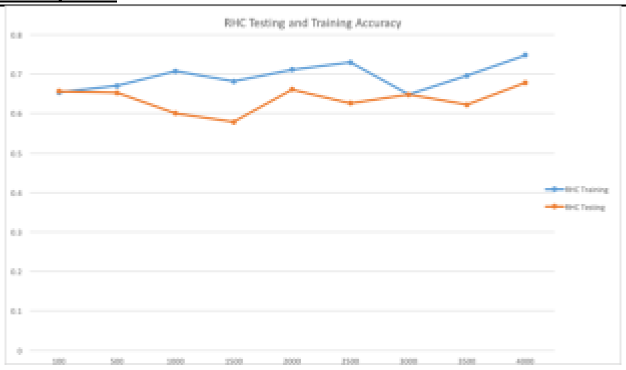
An input layer with 8, hidden layer with 6, hidden layer two with 4 and output layer with one.
To pick the parameters for this test I tried different combinations of each and got the best results using the following parameters.

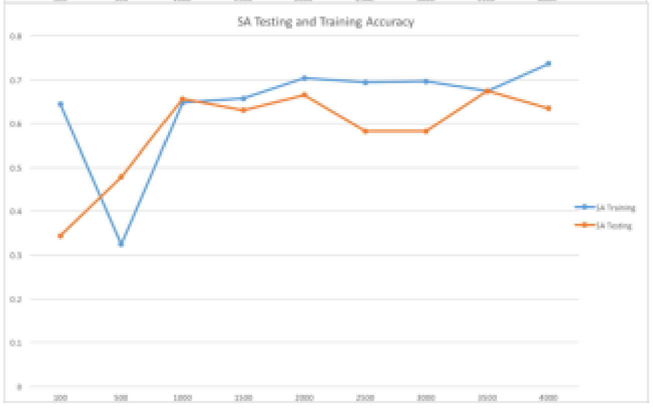
<u>Simulated Annealing</u>: Starting Temperature: 1E11, Cooling Exponent: .95
<u>Genetic Algorithm</u>: Population: 100, Mating Size: 50, Mutation Size: 10

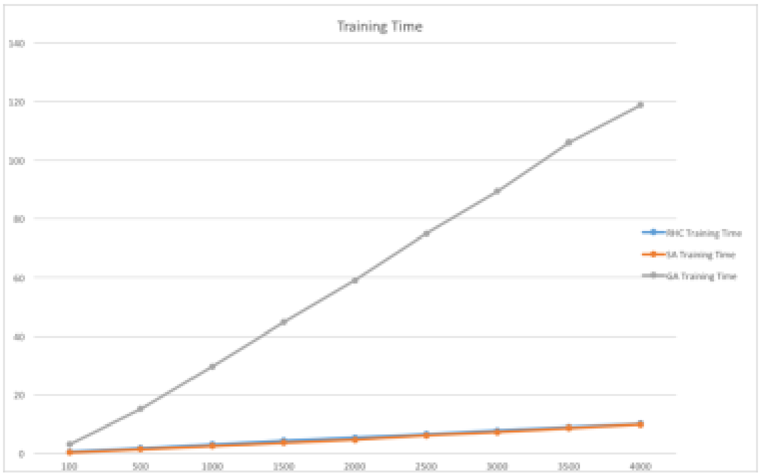
<u>Testing and training accuracy with accuracy (out of 1) on the Y axis and training size on the X axis</u>

<u>Training/Testing Split: 70%/30%</u>









Best results for each algorithm

Algorithm	Highest Accuracy	Iteration	Run Time (Seconds)
BackProp	70.56%	10,000 (Epocs)	142.76
SA	67.41%	3,500	8.76
RHC	67.83%	4,000	10.12
GA	68.02%	2,500	75.68

For this dataset I believe that SA preformed the best which was surprising. I expected that the GA would have performed much better than the rest of the algorithms because it uses local search and crossover to combine populations together. I am guessing the reason that GA did not do that much better was because there were not many subparts of the space that could be independently optimized. We know from lecture that in order for crossover to work that the locality of the bits matter as well as the space must be able to be independently optimized or else you are just randomly mixing things together. My best guess to why this is occurring in my data set is because it has a lot of noise. This supported my initial hypothesis that the noisy data would affect some of the algorithms performance. What makes it ranked as my last choice is the amount of time it takes to train the GA. GA took 75.68 seconds while it took only 8.76 seconds to train with SA and 10.12 with RHC, while only yielding around a .6% improvement in accuracy. Since randomized hill climbing and simulated annealing did decently well I would say that there are not very many local minima's in this dataset for the algorithms to get stuck in because of this I am compelled to say that the noise in this dataset is not very skewed in one way or the other. We know that local minima's are the Achilles heel of these algorithms. In the future I would like to test different starting rates and cooling exponents for SA to how it would affect the time and performance. It was very interesting to see that SA and RHC got very close to the same accuracy to Back Prop while being around 15 times faster. This is a huge improvement; I would be curious to see if these times would stay consistent given a much bigger dataset.

Part 2: Optimization Problems

For this part of the assignment I chose three optimization problems and ran all four search techniques; RHC, SA, GA and MIMIC in order to highlight advantages of SA, GA and MIMIC. When comparing these algorithms, I decided to focus on the accuracy with respect to training iterations and time. To me these were the most interesting and would allow me to get a decent grasp on how each algorithm is preforming.

<u>Optimization Problem 1: Traveling Salesman (Advantages of GA)</u>

<u>Description:</u>

This problem attempts to simulate a city. The optimization problem is to attempt to find the shortest path which visits all points on the city once. It does not care about the direction, just that it hits every node in the city. This is considered an NP-Hard problem. It also should be noted that it is returning the inverse of the solution and this is what we are maximizing, so we are looking for a maximum value, the value that is retuned is 1/distance which gets larger as the distance gets smaller. It should be noted that when returning the results, I took an average of 5 iterations for each.

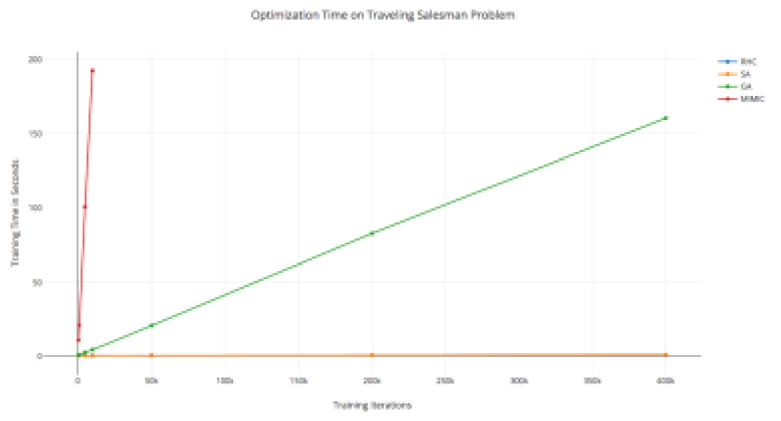
<u>Problem Details:</u>
For this problem I assumed that MIMIC would perform better than all the rest of the algorithms because it could create a very good knowledge structure from this kind of problem. My hypothesis was proven extremely wrong which was very interesting to me. For this problem I decided to test what happens when the input is 50 and 100, also I wanted to see how each algorithm performed when I increased training iterations. Additionally, I decided to plot how long the iterations took in terms of seconds to help discover which algorithm is best suited for this problem. The time is important because for instance we know that MIMIC run orders of magnitude fewer iterations while simulated annealing will take fewer iterations.

Training iterations: [500,1000,500,10000,50000, 200000, 400000]

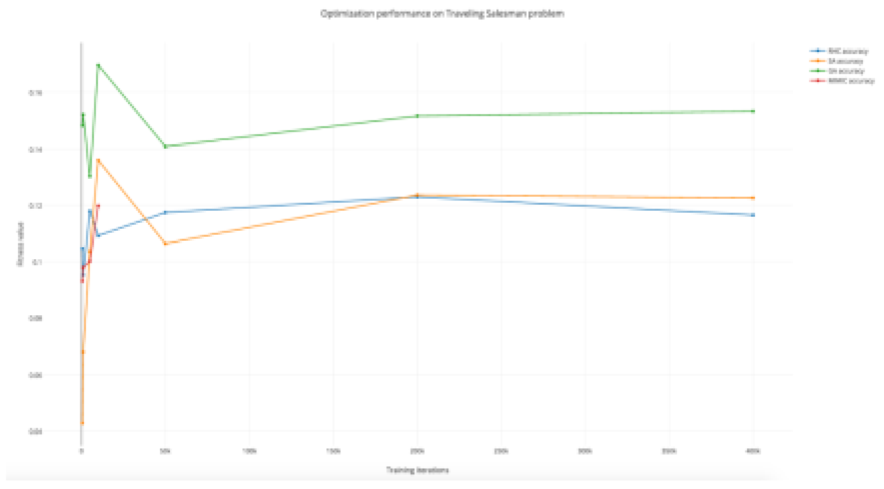
<u>Simulated Annealing</u>: Starting Temperature: 1E12, Cooling Exponent: .95
<u>Genetic Algorithm</u>: Population: 200, Mating Size: 150, Mutation Size: 20
<u>MIMIC:</u> Samples per iteration: 200 to keep: 100

<u>Test 1: Input = 50</u>

Time/Iterations

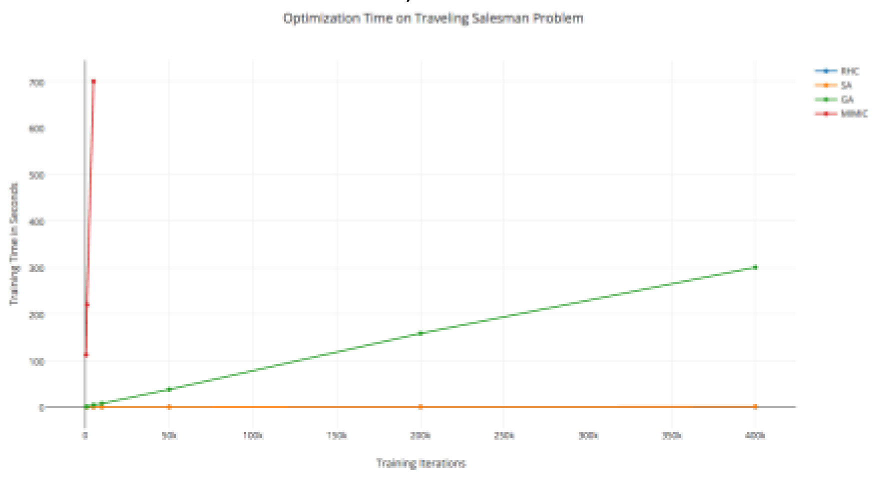


Fitness/Iterations

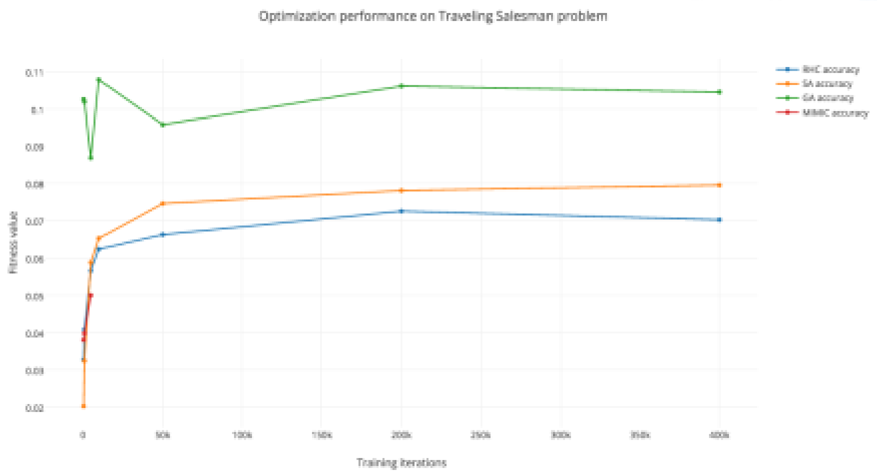


Test 2: Input = 100

Time/Iterations



Fitness/Iterations



Analysis:

It is clear that GA outperformed SA, MIMIC and RHC. This is demonstrated in both tests. For this problem I decided to change what I consider best. For this problem I consider best by looking at the time it takes for each algorithm to reach the highest fitness value.

In **test one**, GA took 10,000 training iterations to find the fastest route .17 in 2.7 seconds. The second best was SA which also took 10,000 iterations but only got to a maximum route value of .135 and it took .052 seconds to reach that. The third was MIMIC which was severely limited because of run time, in 50,000 iterations it only got to a value in .12 which took 190 seconds.

In **test two**, GA took 10,000 training iterations to find the fastest route .109 in 8.3 seconds. The second best was SA which also took 10,000 iterations but only got to a maximum route value of .079 and it took .058 seconds to reach that. The third was MIMIC which was severely limited because of run time, in 10,000 iterations it only got to a value in .05 which took 700 seconds.

These results really surprised me, I did not expect GA to outperform MIMIC and SA. I thought MIMIC would do a lot better because MIMIC is known to do well with structure. My hypothesis why it did not do well is because it did not have enough time to run, but because of time constraints this is a big reason why MIMC would not be considered as a top contender for this problem. We know MIMIC runs fewer iterations more magnitude. This is because MIMIC is drawing from a distribution where values are above median performance then representing a new distribution, if this takes a lot of samples it can take a lot of time. On the other hand, SA is much faster than MIMIC but it also fails to find the fastest route, I believe this is because SA needs more iterations in order to wonder around all the valleys in this problem so that it can bridge the gulf of the local optima's. I would be interested in the future of testing lower temperature values and make sure it is decreasing slowly for this to see if this is really the case. The reason for this is because we know that decreasing temperature values slowly allows SA to not get stuck in the low valleys.

I thought that it was really interesting that GA outperforms all of these algorithms. I believe this is because GA has a part in its algorithm where it takes the top half of the population in terms of their scores and declares them to be the most fit. It then pairs points up and lets them produce offspring with crossover and we let that offspring replace one of the least fit individuals in the population. For this problem I think that GA gets a leg up on the rest of the algorithms in the first set where it selects the best top half. I believe that the other algorithms are selecting points that are on the bottom half of the population that are not as optimal. This is how GA is avoiding doing that, this makes it faster than MIMIC as well.

<u>Optimization Problem 2: Continuous Peaks (Advantages of SA)</u>

<u>Description:</u> This problem is a variation of the 4 peaks and 6 peaks problem which was created in order to see what algorithms perform well in finding the highest peak (global maxima) vs suboptimal peaks (local maxima's). For this problem there are many peaks and dips that are not very evenly distributed as well as being separated by plateaus. Characteristics such as these create lots of local maxima's. These problems are known to cause problems for algorithms that tend to get stuck in local maxima's such as GA. This problem was interesting to me because I wanted to see if this expected result would hold up in practice.

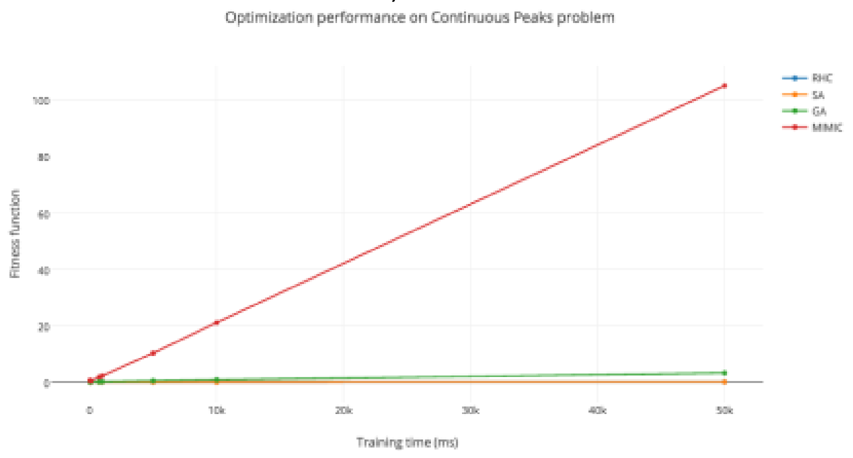
<u>Problem Details:</u>
When comparing these algorithms, I decided to focus on the accuracy with respect to training iterations and time. To me these were the most interesting and would allow me to get a decent grasp on how each algorithm is preforming. It should be noted that I did not adjust the parameters of each algorithm, I did some preliminary tests but decided to save these for a later date because to display results for each would cause the length of this paper to be far too large. I also tested the algorithms on two different input values, the first input I set N=60 and T=N/10. The second input N=100 and T=N/10.

For the training iterations I decided to use the values: [50, 100, 750,1000,5000,10000,50000]

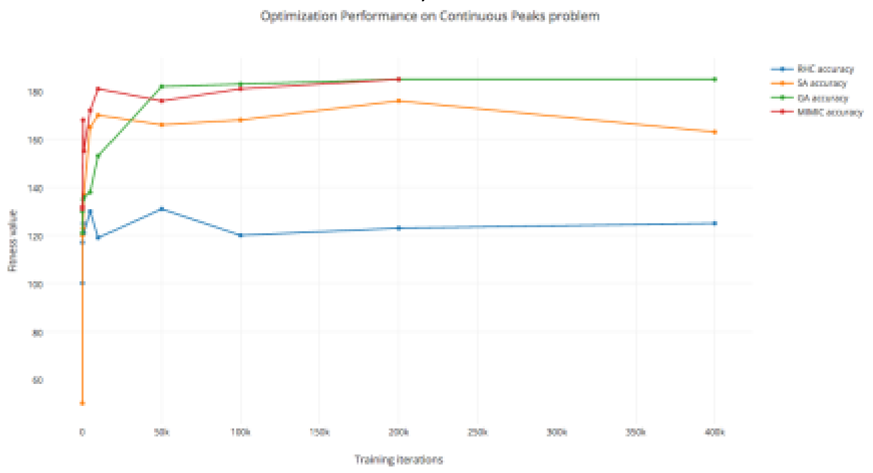
<u>Simulated Annealing</u>: Starting Temperature: 1E11, Cooling Exponent: .95
<u>Genetic Algorithm</u>: Population: 200, Mating Size: 100, Mutation Size: 10
<u>MIMIC:</u> Samples per iteration: <u>200,</u> to keep: 20

Test 1: N=60, T=N/10

Time/Iterations

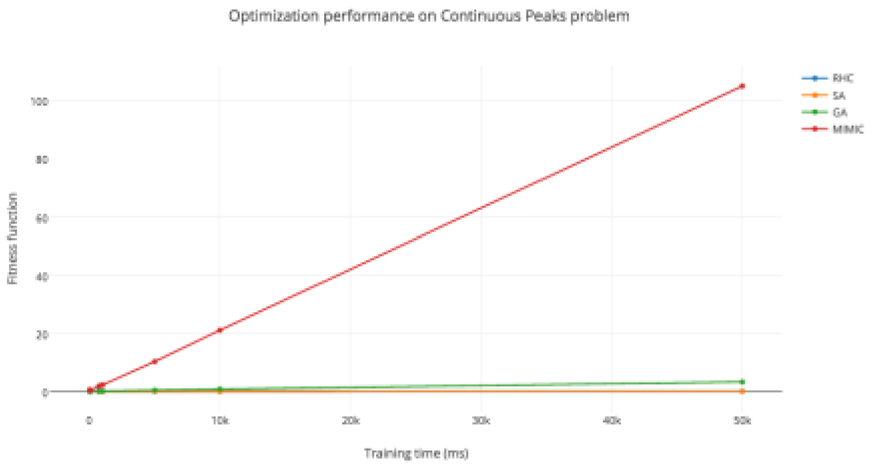


Fitness/Iterations

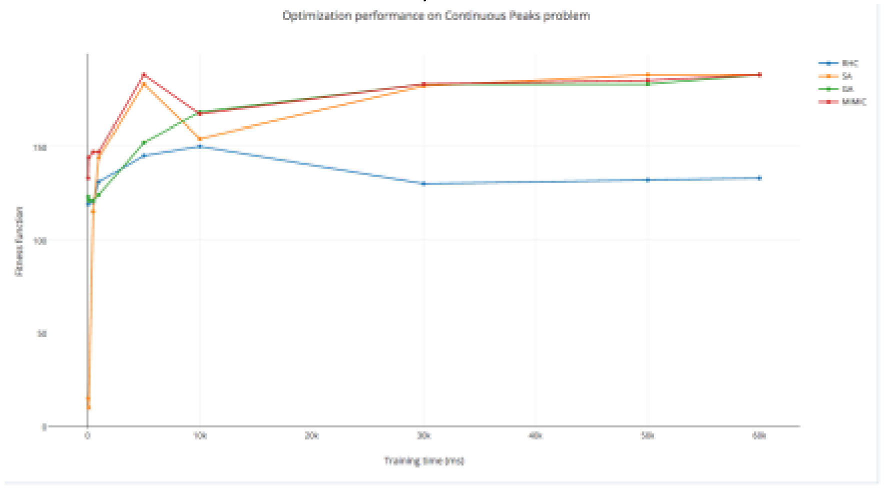


Test 2: N=100, T=N/10

Time/Iterations



Fitness/Iterations



Analysis:

Here you can see that in both tests that my hypothesis that GA would not preform very well is confirmed, it clearly has problems with the local optimum. What was extremely surprising to me was that SA ended up outperforming MIMIC. This is supported by the fact that when comparing times to global optima convergence SA did the best.

In **test one**, MIMIC took 5000 training iterations to converge which took 10 seconds, GA took 50,000 iterations which took 3.2 seconds and SA did the best by finding converging in 5,000 iterations while taking only .009 seconds.

In **test two**, MIMIC took 5000 training iterations to converge which took 23.4 seconds, GA took 30,000 iterations which took 2.59 seconds and SA did the best by finding converging in 5,000 iterations while taking only .01 seconds.

Given this information, it is clear that SA is much better suited for this kind of problem. This problem highlights the advantages of SA because we know that SA is best suited for problems where there are lots of local optima's, which this problem contains. MIMIC did the worse of SA and GA, my assumption on why MIMIC did not do as good in terms of converging speed is because, MIMIC is better suited for problems with structure and this problem did not have much. It would be interesting for further study what would happen if I ran a grid search to find the best parameters for each algorithm tested to see if that would impact the results significantly. I will be leaving this for more exploration at a later time

Although SA did end up being the best choice given this problem, it is worth noting how close GA did. I believe this is because GA does a good job of finding the most fit individuals but still ends up getting stuck in a local optimum when N=60. Further research to see if I adjusted the mutation and crossover rates would prevent this from happening.

Optimization Problem 3: Max K-Coloring (Advantages of MIMIC)

Description:

A graph is K-Colorable if it is possible to assign one of K colors to each of the nodes of the graph such that no adjacent nodes have the same color. Determining whether a graph is K-Colorable is known to be NP-Complete.

Problem Details and Results:

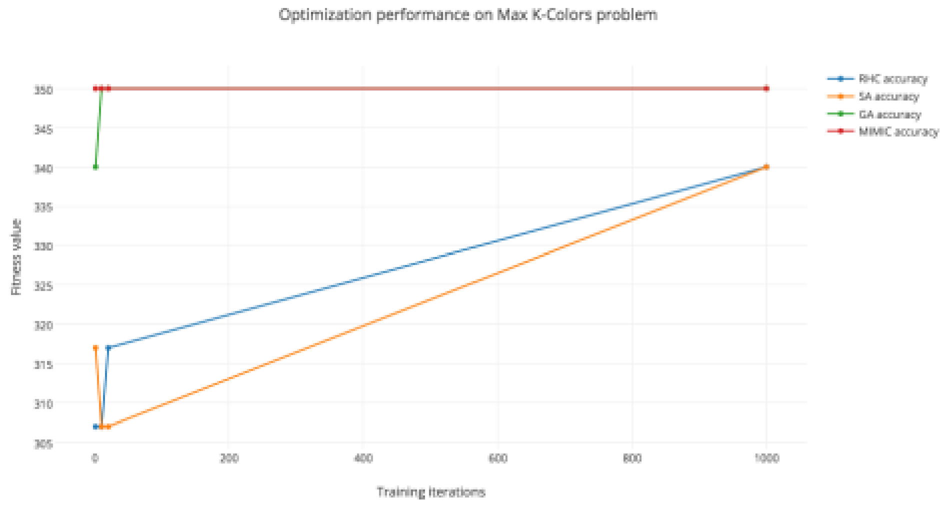
For this problem I tested the number of training iterations when compared to the time it took to find the Max-K color combination. I also tested the algorithms on two different input values for the first input I decided to test the number of vertices set to 50 and for the second test the number of vertices set to 200. These tests were run leaving the number of adjacent nodes per vertex constant at 4 and possible colors constant at 8.

For the training iterations I decided to use the values:
For vertices=50: [1,10,1000]
For vertices=100: [1,10,500,1000]
Note: I explain why I only chose 3 values for the training iterations below.

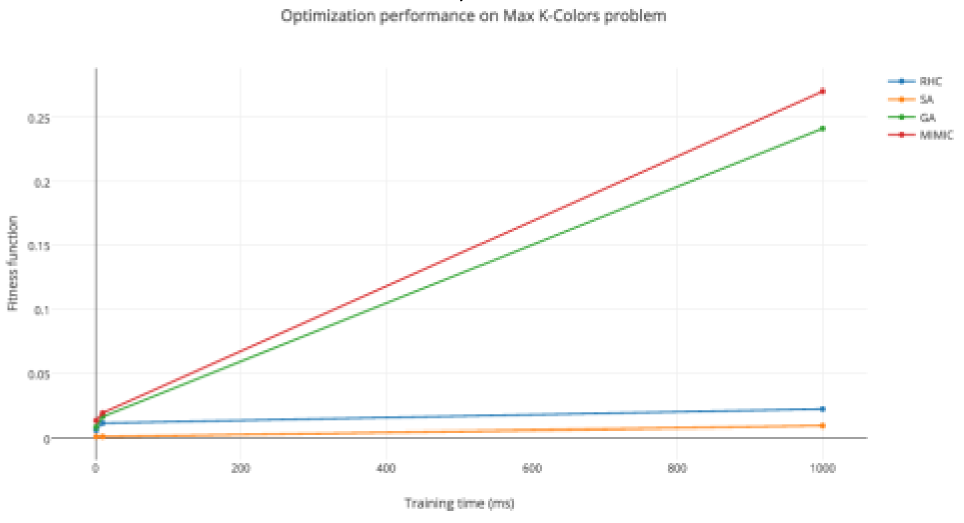
Simulated Annealing: Starting Temperature: 1E12, Cooling Exponent: .1
Genetic Algorithm: Population: 200, Mating Size: 10, Mutation Size: 60
MIMIC: Samples per iteration:40, to keep: 20

Test 1: # of Vertices = 50

Time/Iterations

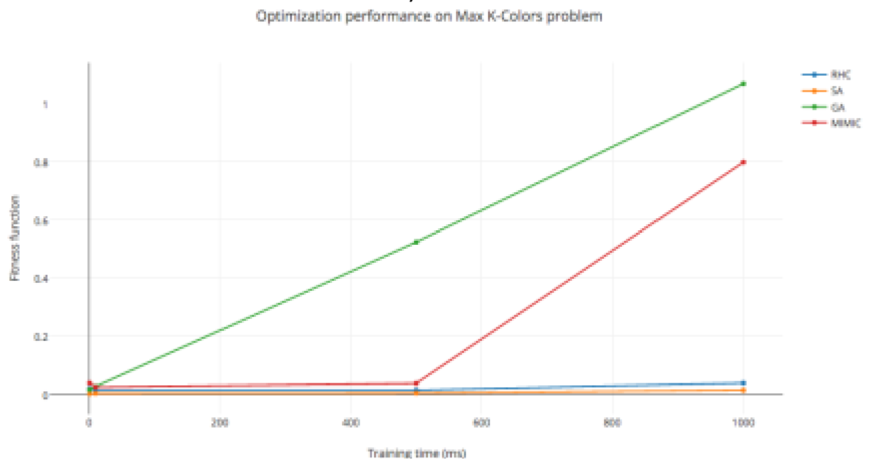


Fitness/Iterations

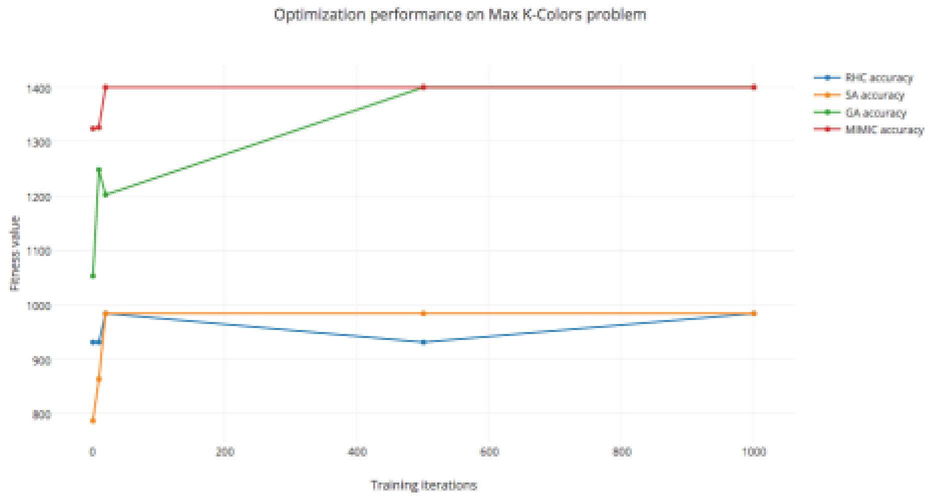


Test 2: # of Vertices = 100

Time/Iterations



Fitness/Iterations



Analysis:

I stopped the graph at 1000 training iterations because even when I increased the training iterations to 200,000, SA and RHC never found the max-K color combination for either test. This was actually very interesting as it clearly shows the advantages of MIMIC and GA while also demonstrating the disadvantages of SA and RHC because they never even converge.

When comparing the two you can see the number of vertices impacting performance on all algorithms.

With N= 40 you can see that GA and MIMIC converge on a max K color extremely fast while it takes a bit more time for N=100. MIMIC outperforms GA in both tests.

In **test one,** MIMIC converges on the max-K color in only 1 iterations taking .013 seconds, while GA converges on the max-K color in 10 iterations taking .016 seconds.

In **test two,** MIMIC converges on the max-K color in 20 iterations taking .035 seconds, while GA converges on the max-K color in 500 iterations taking 1.06 seconds.

This problem also demonstrates the advantages of GA over SA because it was able to find a max-k color when SA was not able to at all. My best educated guess for this is because with GA, any crossover point is representative of some of the backbone of the graphs used. While SA does not use any crossover in order to get information on the population.

This problem more importantly highlights MIMIC's strength of being able to capture all of the structural regularity within the inputs. Additionally, MIMIC demonstrates advantages when the computational cost function is expensive so that lots of information can be collected from each iteration to generate better search results. MIMIC transfers this information stemming from the cost function iteration to iteration. I would be very interested in doing a grid search of the parameters to see how much it would change my results, but I will hold this off for a later time.

Conclusion

In conclusion, we see that each algorithm has its strengths and weaknesses. This should be duly noted when wanting to apply them to different datasets because each will have its own unique characteristics. When analyzing my Pima Indian dataset I discovered that SA, RHC and GA have similar accuracy when comparing them to backprop. What was very interesting was that the time it took for all three was much less than backprop. When comparing the three algorithms we tested, SA really stuck out to me because it was able to get within .5% of GA while being more the 9 times faster which is very significant.

We discovered that GA has an advantage over all the other algorithms is the traveling salesman problem because it selects the top half of the population that has the highest fitness, avoiding points that the other algorithms end up hitting reducing their performance significantly.
We then saw SA has an advantage when looking for global maxima's when there are lots of local minima's, as shown in the continuous peaks problem.
Finally, we saw that when looking for a graph that has a max-k color that MIMIC has a strong advantage followed by GA. MIMIC does well with this problem because it is able to capture all of the structural regularity within the inputs.

All of the results I found were very interesting and really demonstrated the granularity of these randomized optimizers. You should really have a good knowledge of the domain that you are analyzing before choosing which one you want to work with.