Sagar Laud CS 4641
Assignment 2

# Assignment 2: Randomized Optimization

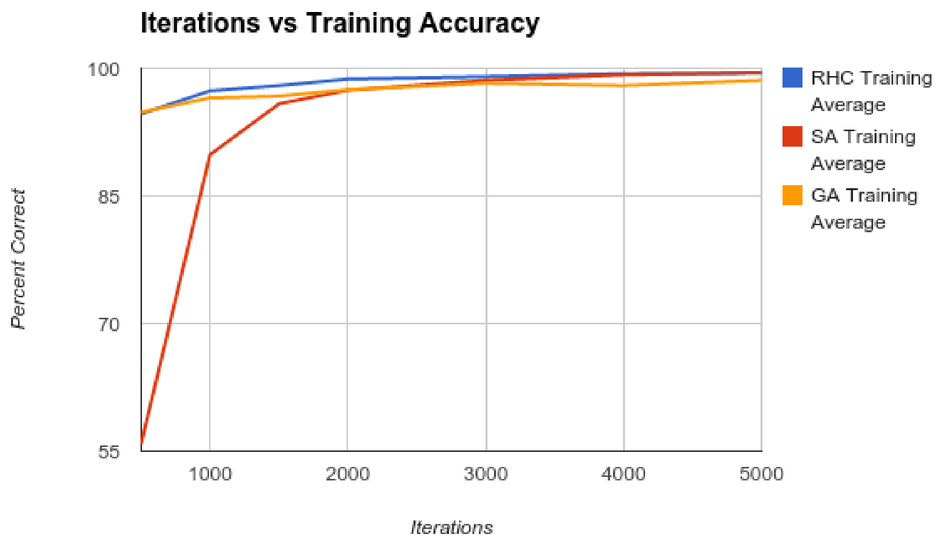
## Part 1:

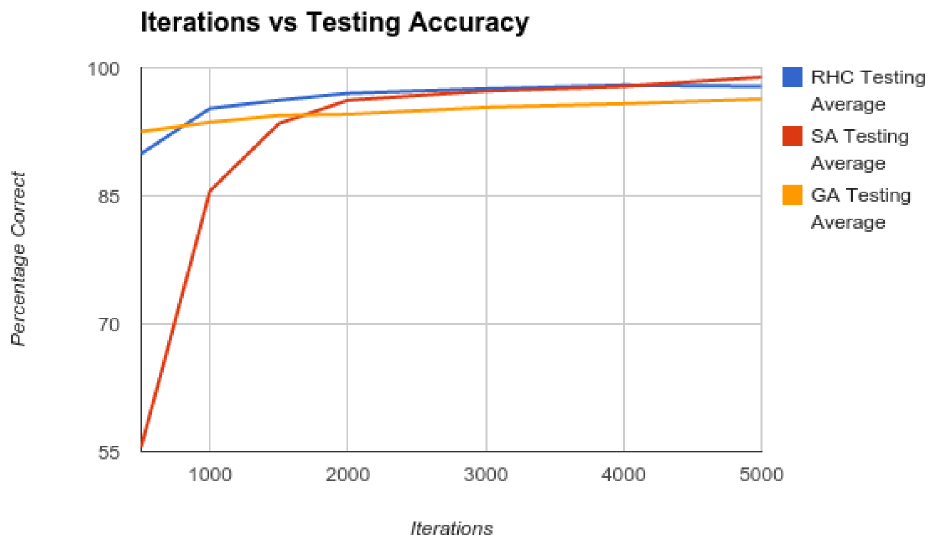
### Dataset:

The dataset used for this project is identical to the voting dataset used in the previous project. The problem itself was already a binary classification problem with discrete values for all of the attributes so there was no reason to modify anything in the dataset or the problem to solve at all.

### Analysis of New Algorithms:

For the purposes of this project, a Neural Network was run without backpropagation and was instead run with Randomized Optimization algorithms updating the weights. The Randomized Optimization algorithms which were used included Randomized Hill Climbing, Simulated Annealing, and the Genetic Algorithm. The dataset was run through the ABAGAIL library while varying the amount of iterations allowed for each Neural Network. The results were very interesting. Naturally because of the nature of Randomized Optimization, 5 trials were run for each configuration and the average was taken in order to gain a better understanding of what was occurring. It is worth noting that these algorithms tend to work better in a discrete state space. Since the weights of Neural Networks are continuous, the algorithms didn't work perfectly. Simulated Annealing and Randomized Hill Climbing can work very poorly depending on the neighborhood chosen. If a poor neighborhood is chosen, then Randomized Hill Climbing may iterate for a long time and never end up anywhere near any maximum! Simulated Annealing can bounce around and since the weights are continuous, it can hit a staggering number of neighborhoods. If the neighborhood it ends up converging in is a bad one, it falls victim to the same issue as Randomized Hill Climbing. As for Genetic Algorithms, the neighborhood function becomes a little more complex as finding a neighbor isn't as simple as flipping a bit.



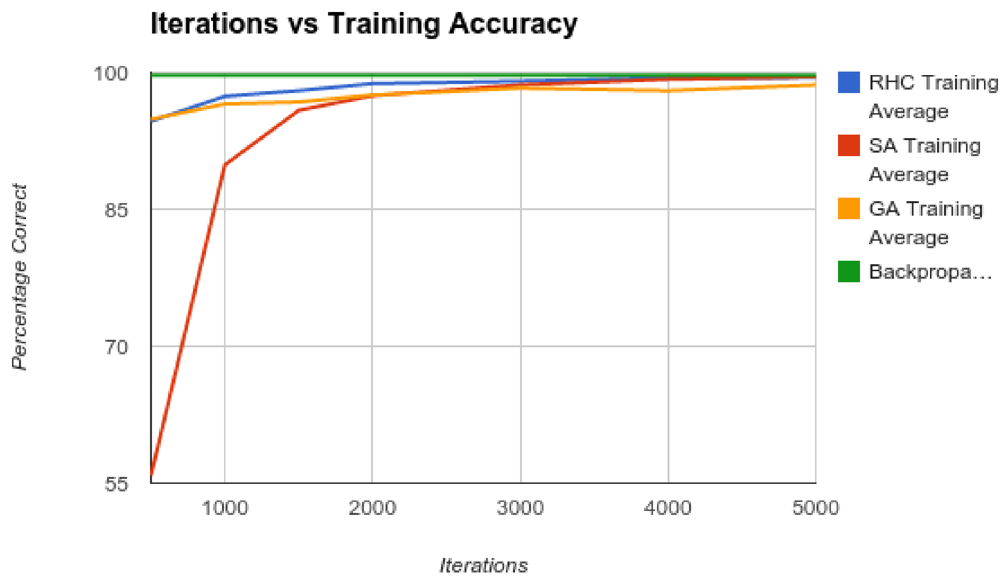


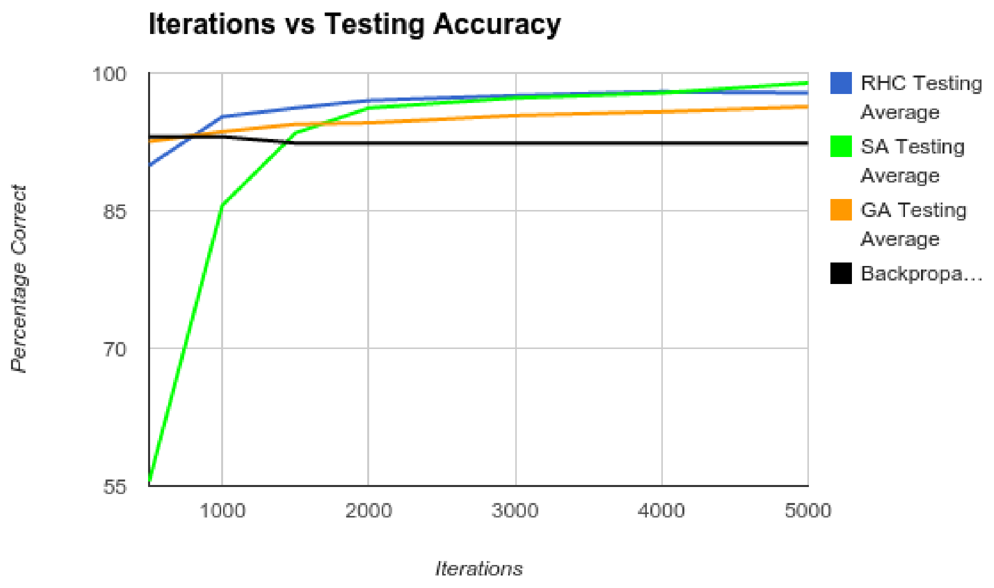
These two graphs display the training and testing curves for the three algorithms in the Neural Network while varying iterations. It is clear that Simulated Annealing did not perform well at all given a very low number of iterations. I believe this is because with such a small number of iterations and with the temperature starting out so high, the algorithm just ends up jumping around and never truly converges to an acceptable maximum. It is clear however that given a reasonable amount of iterations, the Simulated Annealing algorithm performs very well and performs the best out of all three Randomized Optimization algorithms. I believe this to be because of the nature of the dataset. The dataset has quite a number of features, 16 in all excluding the class variable. That is a significant number of features! In addition, none of these features truly dominate others. While each feature has a reasonable information gain as a candidate is likely to vote along party lines and some features may truly be better than others, it is not quite common. There is also the alternative to just not vote which can cause issues as well. All in all, because of the vast number of features, and the fact that no feature truly dominates another, I believe that there are quite a few local maxima. As a result, the Randomized Hill Climbing algorithm had an issue as it has the propensity to get stuck in local maxima. This is not necessarily a terrible thing since all of the features have a reasonable amount of information gain but it does lower the probability of this algorithm finding the optimal value. There isn't really a way to improve Randomized Hill Climbing because of how it works other than to maybe start with a better point using some heuristic. Simulated Annealing however takes Randomized Hill Climbing and lowers the probability of missing the optimal weights. Since we have the ability now to jump around with a varying probability based on temperature, given more time we have a higher probability of finding a better result. This is indeed what ended up happening as by the end, The Neural Network with Simulated Annealing produced a test accuracy of 99% while the maximum accuracy that Randomized Hill Climbing found was 98%. While they are very close, it is evident that Randomized Hill Climbing got stuck in a slightly worse value than Simulated Annealing. The only solution that I can think of to improve the Simulated Annealing algorithm is to tweak the values for starting temperature and cooling rate in an attempt to have it converge to a better maximum.

The Genetic Algorithm also had some issues and took a significant amount of time because there is no truly dominant feature. As a result, when it comes to crossover there is a slight problem simply because the genetic strings will theoretically fail to evolve efficiently. The overall results support this. As the number of iterations increases, the Neural Network which uses the Genetic Algorithm does not really get significantly better. In fact, it overall ends up being the worst out of the three algorithms. It ends up training for much longer than any of the other algorithms yet does not actually achieve better results. The results also remain relatively constant throughout which points to converging quickly while not gaining any more information during crossover as time progresses. The only improvement that I can think of for the Genetic Algorithm is to modify the crossover function to perform best for this particular problem.

Another very interesting thing to address is that there doesn't seem to be any evidence of overfitting! This is very interesting as Neural Networks are known for their propensity to overfit. Looking at the graphs below, we see that while the training accuracy always increases for the Neural Networks using Randomized Optimization, the testing accuracy always increases as well. As time progresses, the Neural Networks do not overfit. This is very different than the Backpropagation curves which are plotted on the graph as well. The training accuracy using Backpropagation is virtually 100% however the testing accuracy decreases as the number of iterations increases. While it is not overfitting significantly, it is still overfitting slightly which we see no signs of in our new Neural Networks. I believe the reason for this is because of the presence of multiple local maxima. While training, no algorithm is guaranteed to find the global maximum and I believe that in a lot of cases, it does not find it. As a result, the Neural Network weights are more generalizable because they aren't optimal for the training set. Therefore, they remain more general and as a result they make it significantly more difficult to overfit.

The final thing worth addressing is the amount of time that it took for these algorithms to finish. For comparison purposes, I ran the algorithms for 5000 iterations and measured the training and testing time over the four algorithms. Backpropagation ended up being faster than any of the other algorithms. Using the results from Project 1, for 5000 iterations Weka only took 6 seconds to build the model and test it. In the case of our Randomized Optimization algorithms however, the Neural Networks relying on these took longer. Randomized Hill Climbing clocked in at about 7.5 seconds on average to build the model and test it while Simulated Annealing clocked in at 7.2 seconds on average to build the model and test it. This is not a significant difference from Backpropagation. The Genetic Algorithm however took significantly longer with about 2 minutes and 20 seconds on average to build the model and test it. This is a significant difference, especially when this algorithm produced the worst results out of all three Randomized Optimization algorithms. It is still worth noting however that all of the Randomized Optimization algorithms ended up with better results overall as time progressed in the Testing Set while Backpropagation won out overall in the Training Set, thereby providing further evidence for overfitting in the Neural Network with Backpropagation.





## Part 2:

Problems:

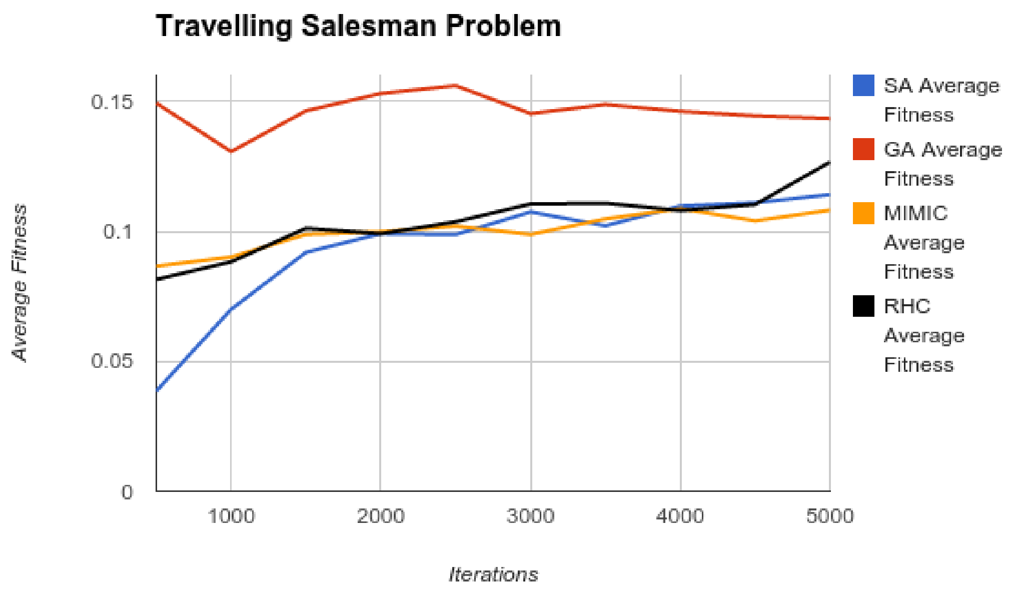
The problems chosen for this part were chosen for a reason. The Travelling Salesman Problem was chosen with the Genetic Algorithm in mind. If a good crossover function is used, it allows it to find good children from it's parents. Because this problem can be broken up into several subproblems and then evolution can be used to find a good solution, this problem seems ideal for this algorithm. This combined with a good crossover function will hopefully lead to good results. MIMIC and Simulated Annealing would have issues with this because of the lack of a strong set of dependencies between points and the sheer size of the space respectively. The Knapsack Problem was then chosen with MIMIC in mind. The reasoning for this is because MIMIC exploits the set of dependencies between points which I feel lends itself well to the Knapsack Problem, as all of the combinations are dependent on each other. Simulated Annealing should do well here as should the Genetic Algorithm though I think the ability to find dependencies will make MIMIC superior here. Finally, the Continuous Peaks problem was chosen with Simulated Annealing in mind. The relatively small domain combined with the presence of a large amount of local maxima is ideal for an algorithm such as Simulated Annealing. It's ability to converge to a good maximum given a smaller dataset combined with many maxima makes it ideal for this problem. It will take a reasonable amount of iterations for it to work properly however.

Travelling Salesman Problem:

This is a classic problem that many have worked on for a long time. The problem essentially asks: Given a list of points and distances between each pair of points, what is the shortest path that visits each point exactly once and returns to the original starting point?

Normally, the best solution to TSP would have the smallest value.  However, we seek to maximize our fitness value and as a result, we will define our fitness as 1/cost so that higher cost paths will result in a lower fitness.

As a baseline, Randomized Hill Climbing actually didn't perform the worst on average. In fact, it performed better than Simulated Annealing on average.  However, three algorithms out of the four ended up performing almost identically over the course of the iterations.  And while Simulated Annealing was the fastest algorithm overall, and didn't produce the worst average, visually it performs about the same as two other algorithms.  Only one algorithm truly won.  The Genetic Algorithm ended up working the best for this optimization problem.  I believe that the reason for this is because the problem itself is very easily broken up into smaller subproblems.  In addition, it appears to have used a good crossover function because of how the results panned out.  As a result, I posit that Genetic Algorithms worked best for this problem simply because the Genetic Algorithm has the ability to find an optimal solution to these subproblems and evolve to the point that it finds an optimal solution for the entire path while using an ideal crossover function.  Simulated Annealing also had a very tough time with this problem because of the sheer size of the space.  There are n! possible paths!  Even given a reasonable amount of time, it just seems very difficult for the algorithm to make the proper jumps to end up at an optimal solution.  Perhaps changing the starting temperature and the rate of cooling would allow it to find better solutions quicker and end up beating MIMIC by more.  MIMIC did not fare any better than Simulated Annealing.  MIMIC works well due to it's ability to find an underlying structure in the problem but it seems that in the end, it did not make a significant difference.  The underlying dependencies which were found by the algorithm just do not seem to help it, implying that these dependencies are not really significant.  It also took far longer than any of the algorithms because it's fitness function is very expensive compared to the other two algorithms.



| Average TSP Statistics | Average Fitness | Average Training Time |
|---|---|---|
| Randomized Hill Climbing | 0.1039613539 | 0.01012415128 seconds |
| Simulated Annealing | 0.09423976114 | 0.01139221006 seconds |
| Genetic Algorithm | 0.1463280479 | 4.307688061 seconds |
| MIMIC | 0.1001464352 | 160.6907774 seconds |

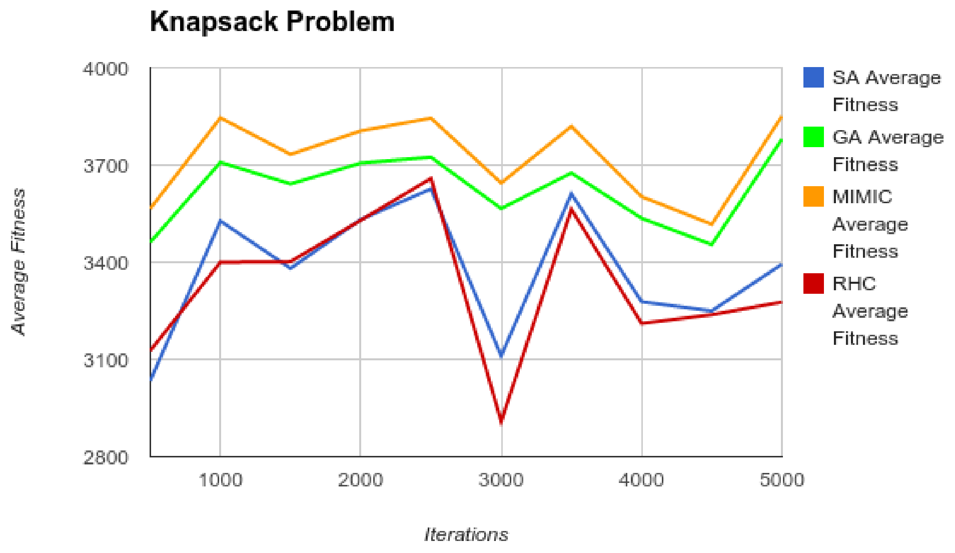
The table above also shows us some interesting information. By taking the overall average of all data collected, we get the overall average for the results. We see that the overall average fitness of the Genetic Algorithm is indeed far superior to that of MIMIC or Simulated Annealing. It's also interesting to see that Simulated Annealing was only slightly behind MIMIC even though it started out with horrible fitness values. Tweaking the parameters mentioned above may even result in a better average solution than MIMIC. The training time is another thing to take note of. The Genetic Algorithm took more time than Simulated Annealing and far less than MIMIC and ended with the best overall solution. However, Simulated Annealing took a fraction of second on average while MIMIC took almost 3 minutes total. There is something to be said for an algorithm which works this quickly and produces results this close to MIMIC. With that amount of time, we could run several thousand more iterations of Simulated Annealing and end up with far better values than MIMIC could do in the same amount of time.

## Knapsack Problem:

This is a classic combinatorics problem. The question this problem asks is: If I have a number of items each with a size and a value, and a fixed-size container, what combination of elements should I use so that I maximize the value within my container?

As a baseline, Randomized Hill Climbing performed the worst overall. It was certainly the fastest overall but this did not make it a reasonable solution. Overall, MIMIC ended up with the best solution and that too, it did it very quickly. At 1000 iterations it more or less matched it's maximum fitness throughout the entire set of 5000 iterations. I believe that MIMIC is superior in this case because it takes into account the underlying "structure" of the problem. In the Knapsack Problem, each step is dependent on the previous step which was taken. The knapsack at each step is filled depending on what has been done in the previous steps. Because MIMIC has the ability to find and exploit this structure, it performs better than the regular Genetic Algorithm. As a result, MIMIC is able to find a better solution while the Genetic Algorithm gets stuck in a slightly lower maximum relative to MIMIC. That being said, the Genetic Algorithm does do well overall. This is due to the nature of the Genetic Algorithm, in that it works by splitting this large problem up into smaller subproblems and evolving to find an optimal solution, which is a fairly intuitive solution. The crossover function still worked well too, but it did not work as well as it did in the Travelling Salesman Problem. So if I were to make a change to this algorithm to improve results, I would attempt to modify the crossover function to make it work better for this space.

Simulated Annealing fails due to how large the search space is. Even with a reasonable cooling rate and having it take it's time with iterations, the space is too large for it. While it converges on a maximum, it is most likely a local maximum that is not really very close to the global maximum. By the time it stops jumping around and begins to favor converging over taking a risk, with the amount of maxima and the size of the space, it will likely end up at a local maximum and as a result, this algorithm is not ideal for this problem. All of these factors force it to perform about the same as Randomized Hill Climbing.



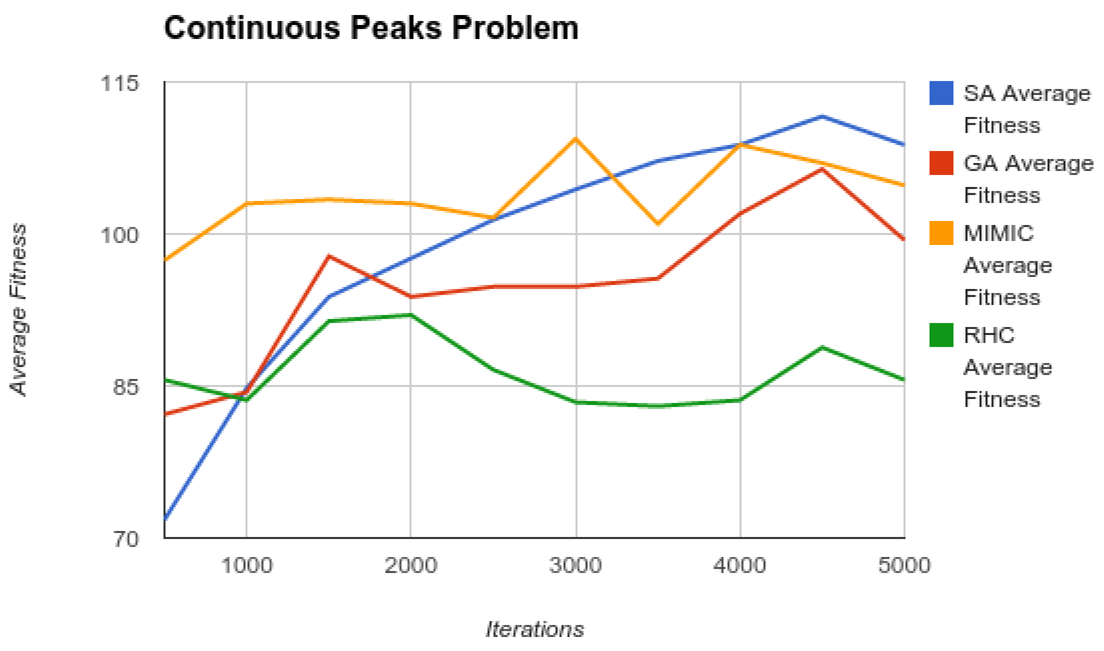
| Average Knapsack Statistics | Average Fitness | Average Training Time |
|---|---|---|
| Randomized Hill Climbing | 3331.552966 | 0.0014239112 seconds |
| Simulated Annealing | 3374.487412 | 0.00267236504 seconds |
| Genetic Algorithm | 3625.307124 | 0.7445345531 seconds |
| MIMIC | 3722.925884 | 18.66035398 seconds |

Looking at the average fitness vs training time, MIMIC has by far the highest average. And that too, it remains very consistent throughout all of the iterations. While Simulated Annealing is fast, it never came close to being able to do what the Genetic Algorithm or MIMIC did. The Genetic Algorithm is still significantly faster than MIMIC but it is worth noting that in 1000 iterations, MIMIC did reach its maximum fitness while the Genetic Algorithm never once topped that mark. As a result, MIMIC is the overall best algorithm for the Knapsack Problem because while it takes longer on average, we do not actually have to wait for the full set of iterations for it to hit its maximum while we have to wait for the Genetic Algorithm (evidenced by its maximum fitness occurring at 5000 iterations). MIMIC may still be a little slower even with cutting down iterations, but that is a small price to pay for a better value when it is a matter of a few seconds.

Continuous Peaks Problem:

The third problem which was chosen was the Continuous Peaks Problem. This problem looks to maximize the number of contiguous 1s and 0s anywhere in a bitstring with a reward for having greater than T contiguous bits set to 0 or for having greater than T contiguous bits set to 1.

In terms of comparing to the baseline, all of the algorithms performed better than Randomized Hill Climbing after the first few hundred iterations. It's clear that it got stuck in local maxima in every case and while it is the fastest of all of the algorithms, it's really meaningless considering that it doesn't find anywhere near the overall maximum. Of the remaining algorithms, Simulated Annealing performed best for this problem. The reasoning for this is because of the nature of the search space. The search space isn't astronomically large. The size of the space is such that even with random jumps, it is still possible to do better while in a very large search space, this would lead to jumping around and never really finding anything acceptable. In addition, this search space is bound to have a lot of local maxima as the consecutive 0s or 1s can be located anywhere within the string. If an algorithm does not try to find a better maximum when it finds something which is reasonable good, it may end up very far from the optimal value. Additionally, the nature of the space seems to be something that MIMIC did well on but not necessarily the best at. It seemed to find some form of structure in the underlying problem very quickly because of how quickly it converged. Whatever it found was found very quickly in the early iterations and from there on out, there wasn't much of a deviation from the original trend. However, this was not powerful enough of a structure for MIMIC to win overall as it was eventually overtaken by Simulated Annealing. Initially, MIMIC is the clear winner and as time progresses, it does remain relatively constant. However, Simulated Annealing ends up overtaking MIMIC at around 3500 iterations and from there, MIMIC never does better than Simulated Annealing. I believe that having an ample amount of time combined with the ability to jump around and find many maxima gives Simulated Annealing the edge here. The Genetic Algorithm had trouble with this primarily because of the presence of a lot of local maxima. The presence of these maxima along with the notion of a "reward" seem to be something that a Genetic Algorithm would have trouble with during evolution resulting in it finding a local maxima and more or less staying there. The crossover function does not seem to have done a stellar job here either so to improve results, I would start with modifying the crossover function to handle points in this space better.. The Genetic Algorithm never really does better than either of the other two relevant algorithms. There is a brief spike but it then drops down to where it remained for the majority of the iterations, implying that while it did find a better solution, that better solution was nothing more than a slightly better local maximum and the true optimum was never found.



| Average Continuous Peaks Statistics | Average Fitness | Average Training Time |
|---|---|---|
| Randomized Hill Climbing | 86.36 | 0.00240813206 seconds |
| Simulated Annealing | 99.02 | 0.00300166392 seconds |
| Genetic Algorithm | 95.12 | 0.393809921 seconds |
| MIMIC | 103.94 | 15.02626037 seconds |

The overall time taken compared to average results is the most intriguing part of this. In this regard, Simulated Annealing is actually in second place behind MIMIC. Looking at the graph however, we see that this is simply because Simulated Annealing needs a reasonable amount of iterations with the current starting temperature and cooling rate. Even with values which start well below the value for MIMIC, it is capable of getting an average very close to that of MIMIC. While MIMIC has the highest average out of all of them, the biggest thing to note here is the amount of time which MIMIC took vs the amount of time that Simulated Annealing took. MIMIC's times are astronomical compared to that of Simulated Annealing. By this logic, we could allow Simulated Annealing to continue for several thousand more iterations before it takes the same amount of time as MIMIC. As a result, considering the averages are very close, yet one takes thousands of times longer than the other and Simulated Annealing ends up with the overall highest fitness in 5000 iterations, I believe that Simulated Annealing is the best algorithm of the three for this problem.