Randomized Optimization

Wesley Tomjack

[wtomjack3@gatech.edu](mailto:wtomjack3@gatech.edu)

**Abstract:**

This report was generated in order to determine the contrasting effects of different optimization problems on certain randomized optimization algorithms. As well as to ascertain the effects of applying said algorithms to optimize the weights used on each feature in a neural network. The optimization problems that were applied to the algorithms were Count Ones, Four Peaks, and Knapsack. Each of those problems were tested against the following algorithms; Random Hill Climbing, Simulated Annealing, Genetic Algorithm, and MIMIC.

**Neural Network Weight Tuning**

**Simulated Annealing**

**Random Hill Climbing**

**Genetic Algorithm**

**Optimization Problem Comparisons**

**Count Ones**

The count ones problem, other wise known as the One Max is an optimization problem which tries to maximize the value of a vector. Passing in a bit-string with the ultimate goal of maximizing (in the case of this experiment) the number of 1’s in the vector (Schaffer. 1991).

An initial state was randomly generated, and the experiment was run 5 separate times on an increasingly larger bit-string with the expectation that as the string grew larger there would be a noticeable difference between the best state generated by the algorithm. Therefore returning the highest fitness value, which in terms of this application would be considered best or optimal.

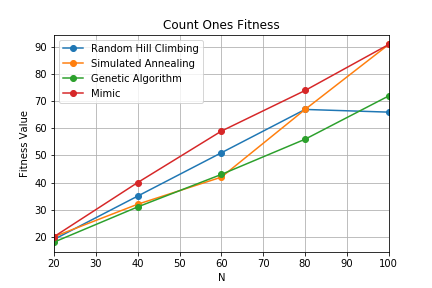
This problem was thought to be an interesting way to explore how algorithms would function in the absence of local optima. They would be able to keep exploring the space without ever getting stuck, giving an interesting view into how under ideal conditions which would do the best.

**Analysis**

Due to the lack of complexity in this optimization problem, it was expected that all of the algorithm’s would do well, with simulated annealing and random hill climbing performing the best. Where best is a representation of both shortest compute time as well as having the highest fitness value (Appendix I) returned. Computationally in terms of wall time it was expected that both simulated annealing as well as random hill climbing should have performed the fastest, as they’re evaluation functions aren’t as complex, while fitness value it was expected that all were quite high.

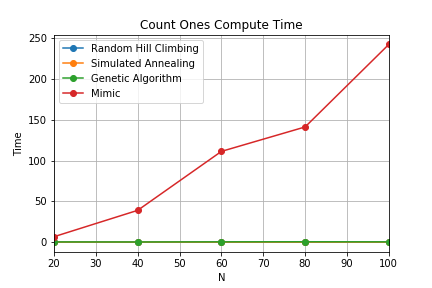
As shown in figure 2.1 below, all algorithms performed well in finding the maximum summation of the vector. With MIMIC, and annealing consistently returning the highest fitness values, for this optimization problem, and genetic algorithms at the lower end of the spectrum. At vector size 60, there is a dip in fitness for Simulated Annealing, this is being attributed to the exploration aspect of annealing to take a worse state over a better one. And since the vector space was small not be able to self correct.

Within algorithms there were parameters that could be tweaked to determine the best results, simulated annealing gave us the ability to set the initial temperature that would be set as well as the decay rate of the temperature. Due to the simplicity of the optimization problem the initial temperature was set very low at about .1, this gave the algorithm the opportunity to maybe explore new states that weren’t as good as the current but that it shouldn’t accept those as the new state.

****

***Figure 1.1: Graphic above depicts the fitness values of each algorithm over a vector space (N) when optimizing the Count Ones problem.***

As seen in Figure 1.2 the compute times for all algorithms were all fractions of a second except for MIMIC. The extremely sharp increase in time is due to the fact that the implementation of the algorithm is not the most efficient in calculating new states, it is a factor that will overwhelm all of the graph comparisons for each of the optimization problems. For the largest vector space each of the other algorithms was under.05 seconds.

****

***Figure 1.2: The compute times of the algorithms when calculating the best state.***

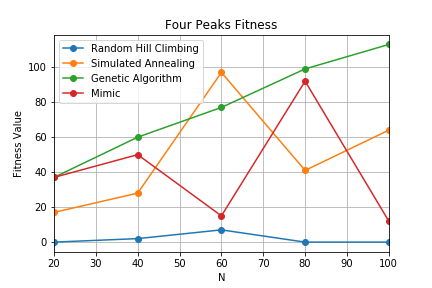
**Four Peaks**

Four peaks is an optimization problem that generates to global maxima, over an N dimensional space. The maxima is achieved when T (Threshold Percentage of N) + 1 leads with >= the amount of 1’s followed by all 0’s or vice versa. A random state was passed in as the initial and an increasing dimensional space size. For figure 2.1 below the Threshold percentage was set relatively low .07, since the higher it got the more random the results appeared. Understandably that is due to the increase in the difficulty of the problem since a higher percentage mean’s a larger basin of attraction between optima’s.

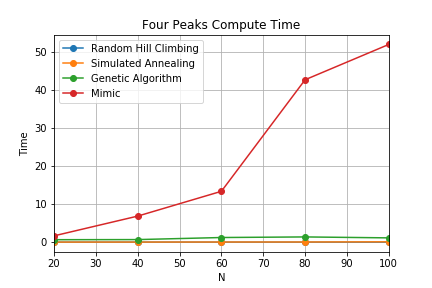
This is an interesting problem, as it has some characteristics that make it troublesome for both Random Hill Climbing as well as Simulated Annealing since it generates large basins of attractions (Appendix II) around the maxima (both local and global). This basin should in some cases catch RHC and SA and keep them from finding the local optima.

**Analysis**

As seen in Figure 2.1, the genetic algorithm function performed significantly better on almost all tests, with the exception of simulated annealing when the dimensional space was of size 60. This will be chalked up to simulated annealing exploring the entire space and luckily landing on the peak which in fact was the true global optima. The temperature parameter for simulated annealing was set very high, since there was a fear that it could get stuck in a local optima and not be able to escape. As far as genetic algorithm parameters were adjusted to increase the population size as well as the maximum number of steps allowed at each step to find the best state. Although this did increase the computation time of the algorithm (Figure 2.2) the fitness value of GA algorithm saw the best results of the 4.

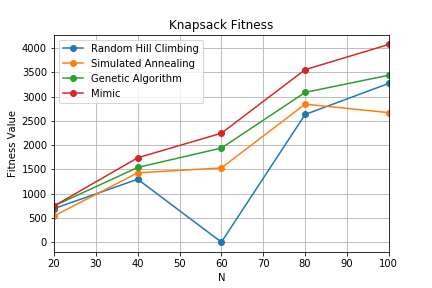
****

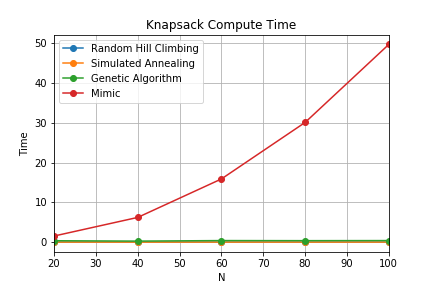
***Figure 2.1: The fitness values for each algorithm over an increasing N-Space***

******

***Figure 2.2: The compute time of each of the algorithms across an increasing N-Dimensional Space***

**Knapsack**

****

****

**Comparisons**

**Appendix**

1. **Fitness Value:** The fitness value, in terms of the mlrose library is the value of the optimally returned state when used in the fitness function.
2. **Basin of Attraction**: A set of points leading to an optima, either local or global.

**Bibliography**

Hayes, G. (2019). Mlrose: Machine Learning, Randomized Optimization and Search package for Python. <https://github.com/gkhayes/mlrose>. Accessed: 10/04/2019

Schaffer, J.D. (1991). On Crossover as an Evolutionary Viable Strategy. Pg. 64. <https://tracer.lcc.uma.es/problems/onemax/files>. Accessed: 10/9/19