**Randomized Optimization Assignment**

Anurag Sahoo  
[asahoo41@gatech.edu](mailto:asahoo41@gatech.edu)

GT ID: 90368005

***Abstract—*** This report deals with the usage of four different random optimization algorithms (Random Hill Climbing, Simulated Annealing, Genetic Algorithm and MIMIC) on three different discrete optimization problems: Flip Flop, Four Peaks and Knapsack. Then Random Hill Climbing, Simulated Annealing and Genetic Algorithm are implemented on determine the Neural Network classifier weights on the Wisconsin Breast Cancer dataset problem.

**Part 1: Analysis of Optimization Problems**

In this part, we will be looking at the different optimization algorithms and assess which performs better in different optimization problems. The four optimization algorithms are:

1. Random Hill Climbing: Random Hill Climbing looks at the neighbors and if the fitness score is higher than that of current position, it makes a move. Once the optima is reached, it makes a random restart to avoid ending up at local optima. Random Hill Climbing is extremely inexpensive, but it may end up in local optima if the attraction basin is smaller.
2. Simulated Annealing: Simulated Annealing works like Random Hill Climbing but with an idea that is sometimes better to explore the space than going uphill. The decision to move is based on the acceptance probability which is a function of difference in fitness score of two points. Simulated Annealing has a better chance of not landing up at local optima and is inexpensive.
3. Genetic Algorithm: It employs randomized, parallel hill climbing algorithm to search a hypothesis that optimizes a fitness function. It operates by iteratively updating a pool of hypothesis, termed as population. A new population is generated by selecting the most fit individuals, and then an offspring is produced via crossover. A small percentage of the population is altered via mutation. As the movement of hypothesis happens abruptly, Genetic Algorithm is less likely to get tapped in local optima. It is computationally expensive though.
4. MIMIC: It stands for Mutual Information Maximizing Input Clustering. Unlike other algorithms, MIMIC uses the structure to help a randomized search through the space. MIMIC tries to find optima by estimating probability densities. MIMIC is also computationally expensive.

For this assignment, I will be using three different discrete optimization problems: Flip Flop, Knapsack and Four Peaks.

**Flip Flop**: Flip Flop is an optimization problem which counts the number of times of bits alternation in a bit string. As the strings are randomized, there is a possibility of local maxima/minima.

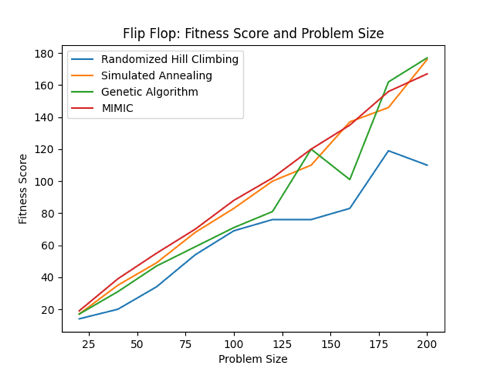
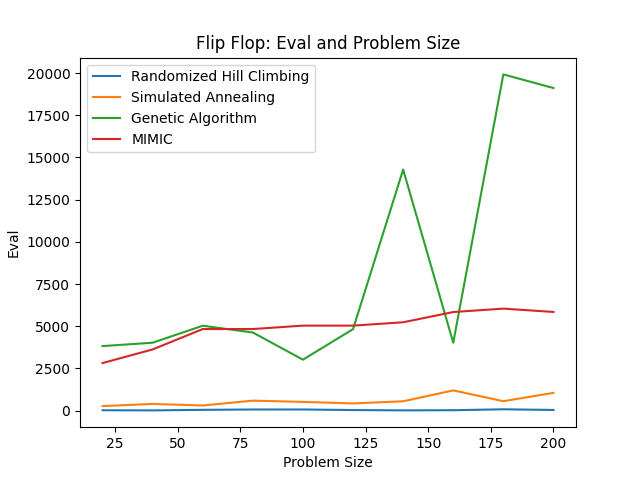
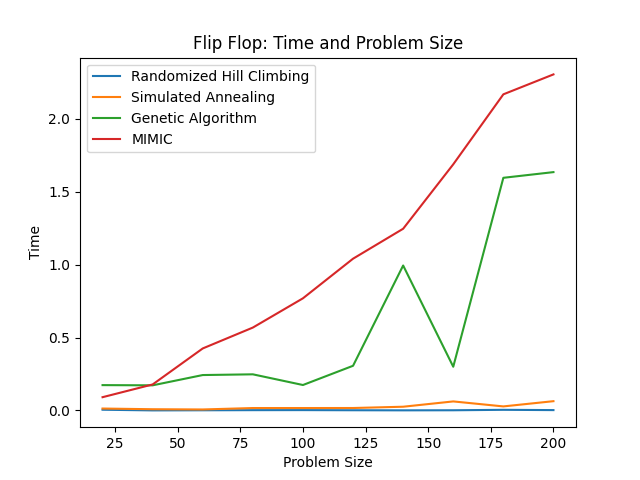
 

Fig 1: Flip flop optimization problem: fitness score, time and function eval vs problem size

I ran the experiment of assessing the fitness score, time taken, and function evaluation for the Flip Flop optimization problem with varying problem sizes. From the fitness score curve, Simulated Annealing seems to be doing well at different problem sizes. However, looking at the time and function evaluation, it is evident that Simulated Annealing beats MIMIC as the former takes less time and less function evaluations. Simulated Annealing, because of it’s ability to explore the space than just moving uphill, is able to evade the local optima. Hence, it is able to perform as good as MIMIC. However, MIMIC is computationally expensive. Hence, Simulated Annealing would be preferred for this optimization problem.

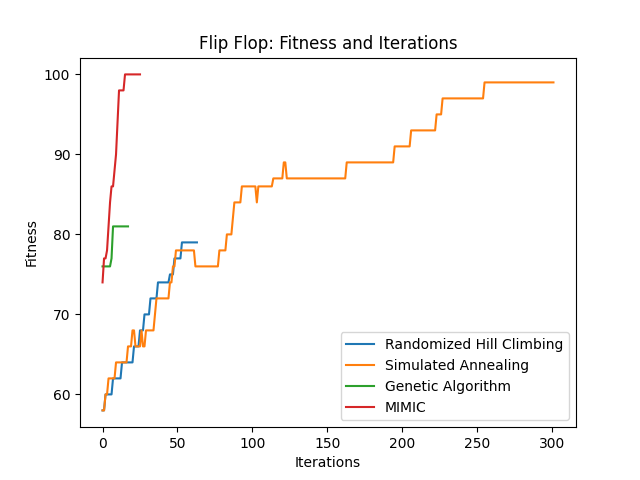
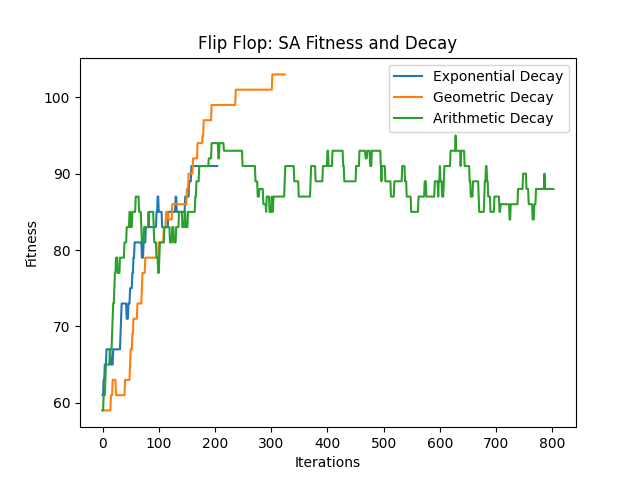


Fig 2: Flip Flop: Fitness Score vs Iterations

From the above chart, it is evident that both MIMIC and Genetic Algorithm takes very less iterations to converge but Genetic Algorithm is converging at a very low score. Simulated Annealing reaches a good fitness score steadily with increasing iterations. Random Hill Climbing performs poorly.

Chart, line chart

Description automatically generated

Chart, line chart

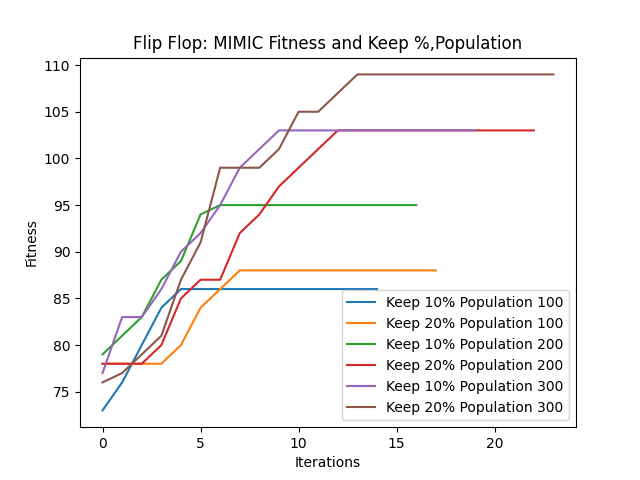
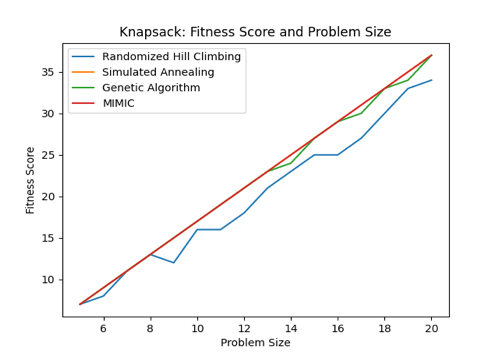
Description automatically generated

Fig 3: Flip Flop: Varying hyper-parameters

For each of the algorithms, I tried varying the hyper-parameters and the charts were generated. For Simulated Annealing, I varied the schedule with different types of decay: Geometric, Exponential and Arithmetic with the default value of decay. Geometric Decay of Temperature reaches the global optima, whereas the Exponential Decay and Arithmetic Decay gets stuck in local optima. The Arithmetic Decay reduces the temperature extremely slow and Exponential Decay reduces the temperature vigorously. So, both might miss the global optima. The Geometric Decay reduces the Temperature in a decent speed and might converge. For Random Hill Climbing, I varied it with number of restarts. The restarts help in evading the local optima. Low restarts value might get tapped in local optima, whereas extremely high value of restarts may miss the attraction basin. For the Flip Flop optimization problem, restarts of 8 achieves higher fitness score. The population (hypothesis) size and mutation probability were varied for Genetic Algorithm. Higher the population size and mutation probability, more the variability within the hypothesis. Low value of population and mutation probability may not reach the attraction basin because of its slow change. Mutation probability of 20% and population size of 200 gives better fitness for this problem. For MIMIC algorithm, I varied the population (hypothesis) size and keep percentage (what proportion of samples to retain at each iteration of algorithm). MIMIC gives the best fitness with population size of 300 and a keep percent of 20%.

**Knapsack problem**: Knapsack is a combinatorial optimization problem in which, given a set of items with weights and values, we need to determine the number of each item to include in a bag to maximize the value keeping in mind the constraint that the total weight should not exceed that of bag.

**N Queens problem**: N Queens is an optimization problem of placing N Queens in a NxN chess board such that no two queens threaten each other.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated with medium confidence

Fig 4: N Queens: Fitness score, Time, Function Eval vs Problem Size

Fitness score, Time and Function eval curve was plotted for varying problem sizes of Knapsack optimization problem. MIMIC seems to be beating others consistently at different problem sizes. However, it does take more time and function evaluations comparatively.

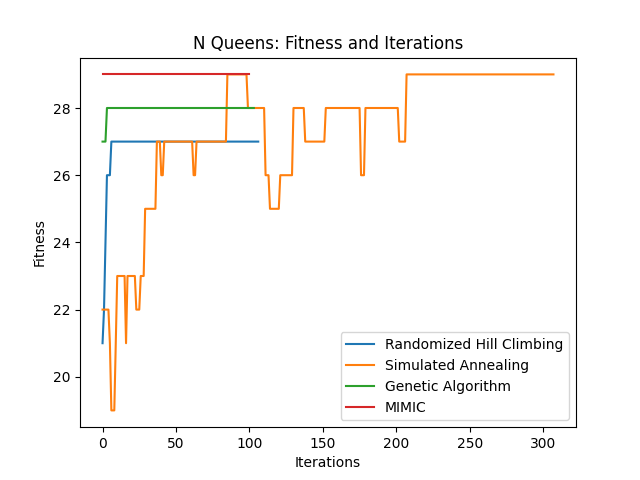


Fig 5: N Queens: Fitness Score vs Iterations

The above chart was generated for 16 Queens problem. Clearly, MIMIC and Simulated Annealing are achieving a good fitness score and possibly getting the global optima. Random Hill Climbing and Genetic Algorithm might have been stuck in local optima, hence having lower fitness scores.

Chart, line chart

Description automatically generatedChart

Description automatically generated

Chart

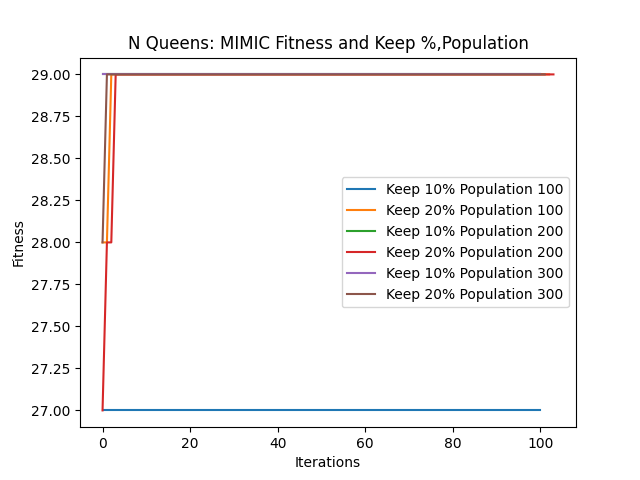
Description automatically generated with medium confidence

Fig 6: N Queens: Varying hyper-parameters

I tried varying different hyper-parameters for each of the algorithm. For Simulated Annealing, Arithmetic Decay might be getting tapped in local optima regions because of the slow decay in Temperature, hence we see lot of variability even after many iterations. On the other hand, Exponential and Geometric Decay is taking fewer iterations to reach the optima. Geometric Decay achieves optima tad bit faster than Exponential Decay. For Random Hill Climbing, restart of 10 achieves the optima fastest indicating that the algorithm needs higher number of restarts to reach the attraction basin. I varied the mutation probability and population size for Genetic Algorithm. Higher the population size and mutation probability, more the variability within the hypothesis, but extremely high value might make the movement more abrupt and might miss the attraction basin. A mutation probability of 10% and population size of 200 works well for this problem. For MIMIC, I varied both keep % and population size. It turns out that it achieves global optima for almost all configurations. However, a low value of keep % (10%) and population size of 100 might be tapped in local optima.

**Four Peaks**: Four Peaks is a simple toy problem and is good for testing algorithms. The fitness consists of counting the number of zeros at the start and ones at the end and returning the maximum. It is a problem with two local optima with wide basins of attraction, and two sharp global optima at the edges.

For the Four Peaks optimization problem, I implemented different algorithms with varying problem sizes. Genetic Algorithm seems to be the clear winner at different problem sizes. Although, the Genetic Algorithm takes more time and function evaluations to reach optima.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Fig 7: Four Peaks: Fitness Score, Time, Function Eval vs Problem Size

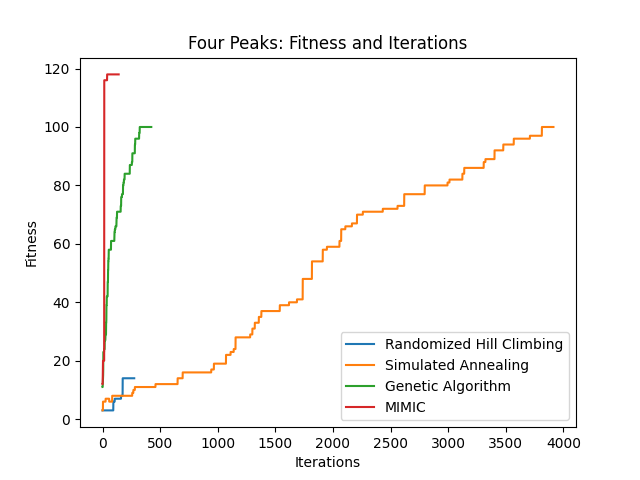


Fig 8: Four Peaks: Fitness Score vs Iterations

Genetic Algorithm and MIMIC reaches the optima quickly in fewer iterations. MIMIC’s fitness score is observed to be higher. Simulated Annealing also achieves a good fitness score, but it takes more iterations.

Chart

Description automatically generatedChart, histogram, box and whisker chart

Description automatically generated

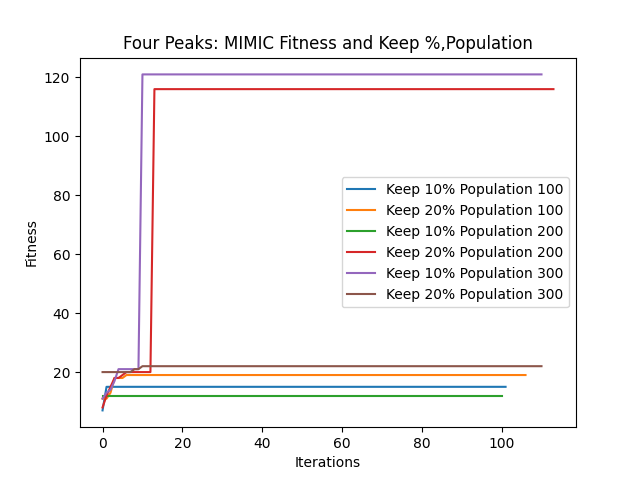
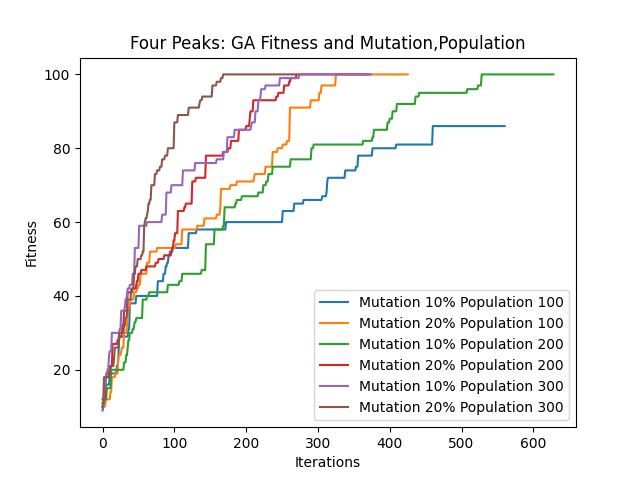


Fig 9: Four Peaks: Varying hyper-parameters

Different hyper-parameters for each of the algorithm were varied and analyzed. For Simulated Annealing, all the decays are behaving in tandem. However, Geometric Decay reaches optima in slightly lower iterations. For Random Hill Climbing, restart of 8 and 10 achieves the optima faster signifying that the algorithm needs higher number of restarts to reach the attraction basin. Mutation probability and Population size are varied to assess the fitness score. Higher population size and mutation probability increases the variability, and hence might achieve global optima. Mutation probability of 20% and population size of 300 achieves higher fitness score with lower iterations. For MIMIC, higher population size of 300 and a keep % of 10% gives the best fitness score indicating higher variability with a good population size may achieve global optima.