## [**CS-7641**](https://gatech.instructure.com/courses/224746) **Randomized Optimization**

## 

* **Optimization problems chosen:**

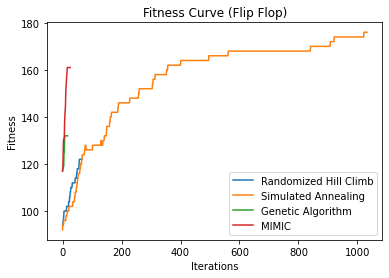
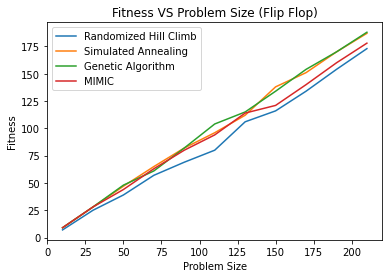
Flip flop to highlight simulated annealing.

Four peaks to highlight genetic algorithm.

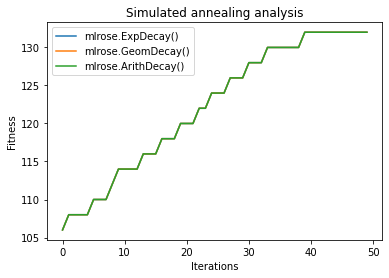
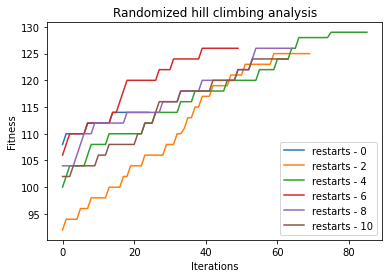
Knapsack to highlight MIMIC.

* **Common steps carried on all optimization problems:**

1. Ran the problems over different problem sizes, to select the best size of problem.
2. Ran the problems for multiple iterations to see when it converges.
3. Carried out hyperparameter optimization for all 4 algorithms to select the best parameters.
4. Calculated the time needed for each algorithm.
5. **FLIP FLOP:**

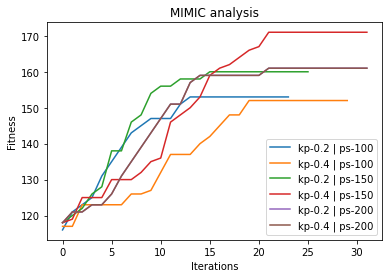
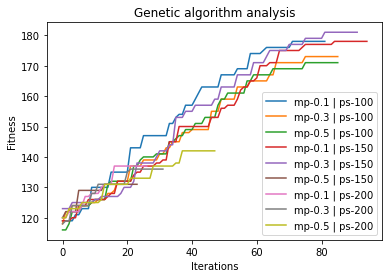


As seen in the above graphs, simulated annealing takes more iterations than others, but has a high fitness score. MIMIC had a good score at very low iterations but then did not improve.



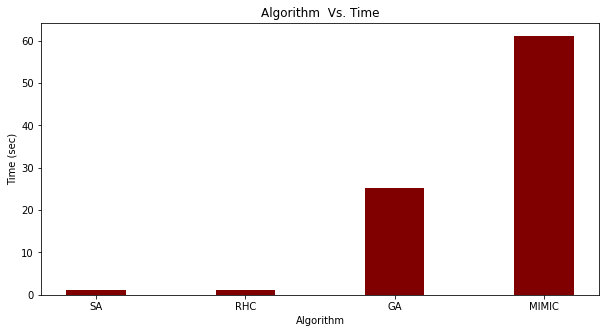
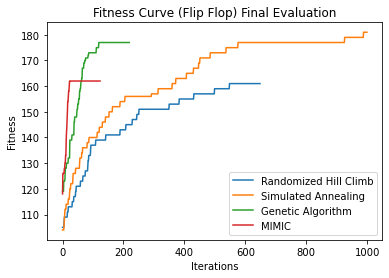
*‘restarts’ parameter tuning - RHC ‘schedule’ parameter tuning - SA*

As clearly seen here, for randomized hill climbing, 4 restarts were the best performing and for simulated annealing, all the 3 schedules performed similar to one another.



*‘Mutation\_prob’ and ‘pop\_size’ tuning - GA ‘keep\_pct’ and ‘pop\_size’ tuning MIMIC*

For the genetic algorithm, mutation\_prob of 0.3 and pop\_size of 150 were the best performing and for MIMIC keep\_pct of 0.4 and pop\_size of 150 was best performing.



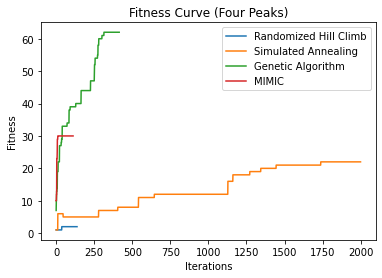
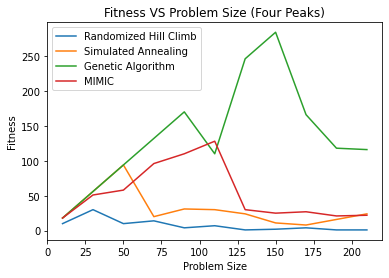
*Final tuned parameters for all algorithms Algorithm and time taken to get best fitness*

After tuning the hyperparameters, simulated annealing took more iterations than others, but had the best fitness score. Also genetic algorithm performed similar to simulated annealing but the time taken for simulated annealing is very much lower than that of genetic algorithm.

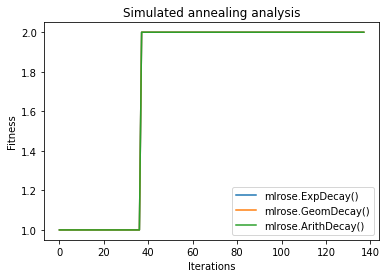
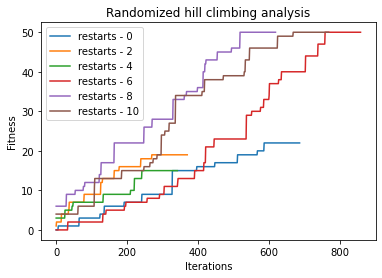
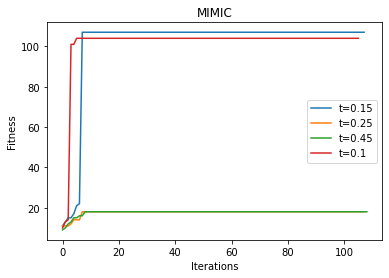
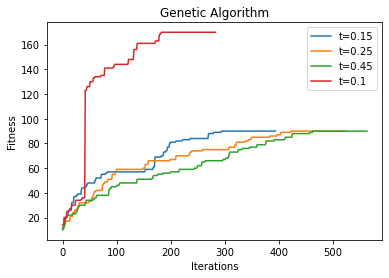
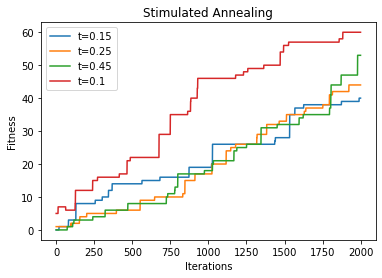
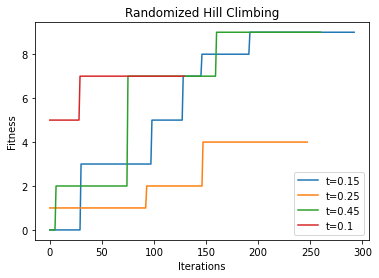
1. **FOUR PEAKS:**

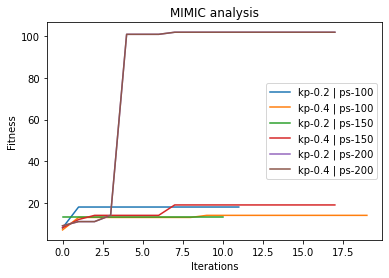
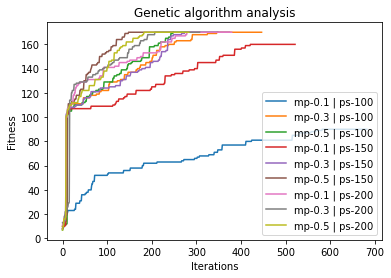
I chose a problem size of 90 so all algorithms have a smaller size and the code will run faster.

Also, for the number of iterations the genetic algorithm outperforms all the other algorithms with huge differences.

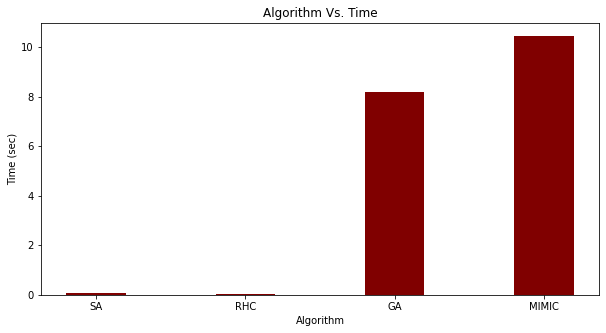
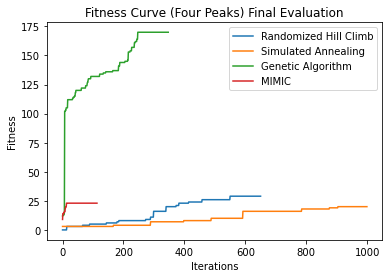
****

I also tuned the **t\_pct** parameter of the four peak fitness function. The results are below.

**** *‘restarts’ parameter tuning - RHC ‘schedule’ parameter tuning - SA*

****

*‘Mutation\_prob’ and ‘pop\_size’ tuning - GA ‘keep\_pct’ and ‘pop\_size’ tuning MIMIC*

****

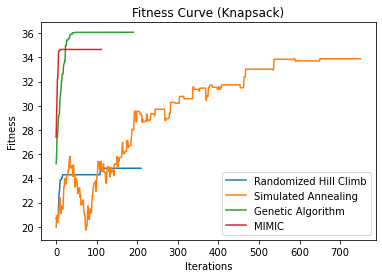
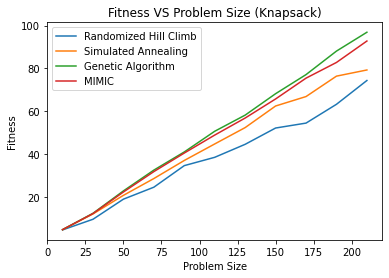
*Final tuned parameters for all algorithms Algorithm and time taken to get best fitness*

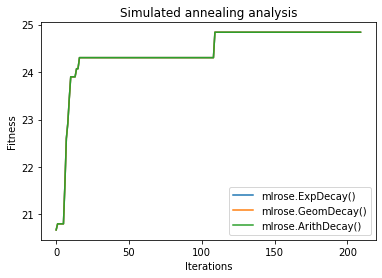
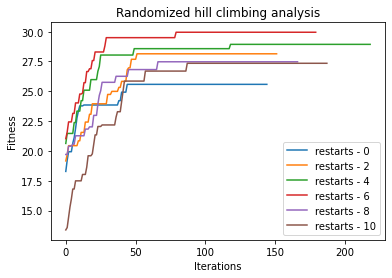
After tuning the hyperparameters, for all 4 algorithms, it is very clear that the genetic algorithm outperformed the rest of the algorithm and also found the best fitness in less iterations. Although the time taken by genetic algorithm is higher than SA and RHC, as the fitness is more in less iterations, we can say it's the best performing.

Genetic algorithm (GA) performs well on Four Peaks as it exploits the problem space using crossovers. Randomized hill climbing is not efficient even after using restarts where it gets stuck in local maxima. Also, GA requires less iterations

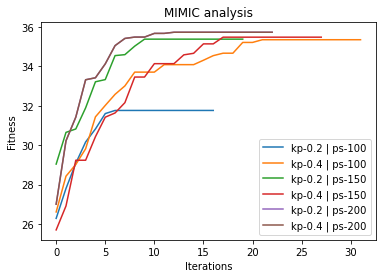
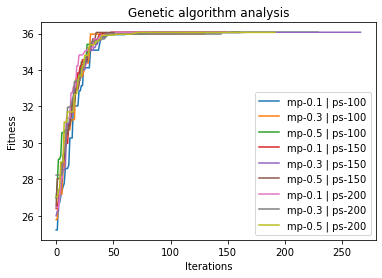
1. **KNAPSACK:**

For knapsack I chose the problem size as 75. Initially without hyperparameter optimization, genetic algorithm outperformed MIMIC

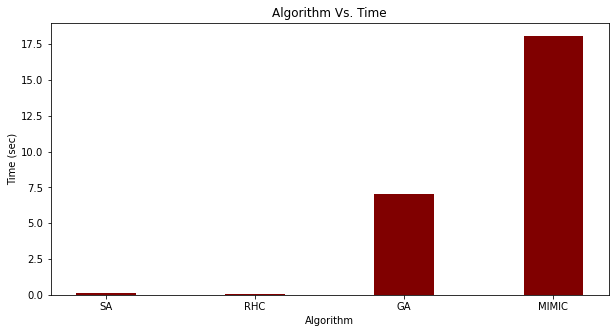
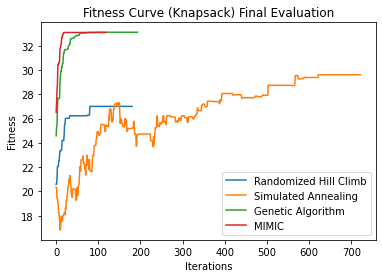
****

****

*‘restarts’ parameter tuning - RHC ‘schedule’ parameter tuning - SA*

****

*‘Mutation\_prob’ and ‘pop\_size’ tuning - GA ‘keep\_pct’ and ‘pop\_size’ tuning MIMIC*

****

*Final tuned parameters for all algorithms Algorithm and time taken to get best fitness*

After tuning the hyperparameters multiple times, MIMIC and genetic algorithm were performing very similar in terms of fitness and MIMIC was taking more time than that of GA. But MIMIC in less iterations has best fitness and GA after more iterations has similar fitness to that of MIMIC.

Genetic algorithm and MIMIC perform very similarly on Knapsack. But as MIMIC is more efficient as in it communicates between iterations and thus finds best fitness before GA.

* **References**

1. <https://mlrose.readthedocs.io/en/stable/source/algorithms.html>
2. <https://mlrose.readthedocs.io/en/stable/source/intro.html>
3. <https://mlrose.readthedocs.io/en/stable/source/fitness.html>
4. <https://github.com/gkhayes/mlrose>
5. https://github.com/gkhayes/mlrose/blob/master/mlrose/algorithms.py