Randomized Optimization

Random Search Algorithms

For the experiment, the following local random search algorithms were used.

- **Random Hill Climbing**
  The algorithm starts at a random point and moves to the point that has the best fitness value through iteratively calculating the fitness of the neighboring points. It does a hill climb.
- **Simulated Annealing**
  The algorithm derives its name from the metallurgy process of heating and cooling metals. It takes a parameter T, which defines the temperature. At higher values of T, the algorithm explores random points to escape local optima and at lower values of T, the algorithm exploits the input space to derive the global optima.
- **Genetic Algorithm**
  The algorithm is inspired by the theory of evolution. From the initial population sample, the fittest ones are selected. Crossover happens, and the children go through mutation, which is a random change in the values and are added back into the population. The global optima corresponds to the value or individual with the best fitness.
- **MIMIC**
  The algorithm defines a structure by determining the probability distribution. The top few are further taken forwards and a new probability distribution determined. This is repeated until a global optimum is determined.

Three optimization problems are used to demonstrate the strengths of the individual random search algorithms. They are:

- **Four Peaks**
- **Knapsack [3] [4]**
- **Flip Flop**

The **mlrose-hiive** library is used extensively to solve the random optimization problems. The problem size is limited between 100 and 200.

Four Peaks

The four peaks problem is taken from [2]. Given an N dimensional input vector $\vec{X}$, the four peaks evaluation function is defined as:

$$f(\vec{X}, T) = \max\left[\text{tail}(0, \vec{X}), \text{head}(1, \vec{X})\right] + R(\vec{X}, T)$$

$where$

$$tail(b, \vec{X}) = \ number\ of\ trailing\ b's\ \text{in}\ \vec{X}$$
$$head(b, \vec{X}) = number\ of\ leading\ b's\ in\ \vec{X}$$

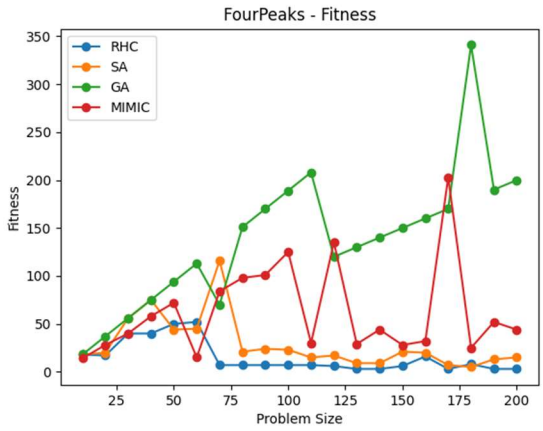
$$R(\vec{X},T) = \begin{cases} N, & if\ tail(0,\vec{X}) > T\ and\ head(1,\vec{X}) > T \\ 0, & otherwise \end{cases}$$

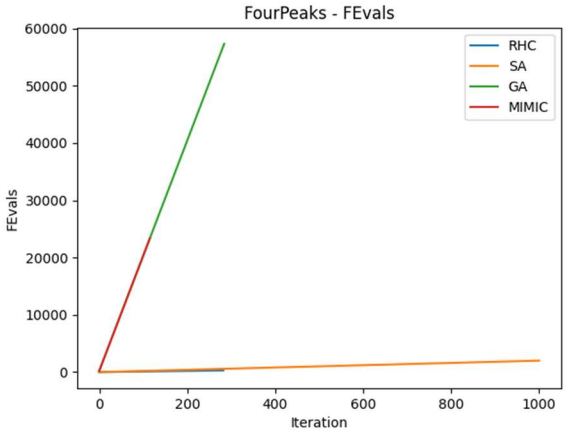
There are two global maxima for the function. There are also two sub-optimal local maxima that occur for a string of all ones or a string of all zeroes. The two global maxima are attained when there are T + 1 leading ones followed by all zeroes or when there are T + 1 trailing zeroes preceded by all ones. For large values of T, the basin of attraction for the sub-optimal local maxima become larger and thus the problem's difficulty increases.

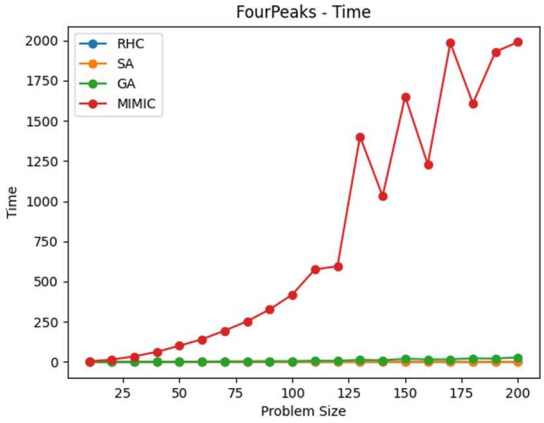
The results for this problem are shown below.

Genetic algorithm does the best as compared to the other three random search algorithms. The crossover and mutation features of the algorithm help it to traverse the sub optimal basins of attraction and then derive the global optimum. The number of function evaluations are considerably higher however the time taken and the high fitness score in general make it the best algorithm to solve this problem.

Mimic takes a long time to determine the best fitness values and is not as good as the genetic algorithm. Random hill climbing and simulated annealing do not do well in determining the global optimum for this problem even though the time taken and function evaluations are minimal. The exploit and explore features of the algorithms are unable to handle the sub optimal basins.







Knapsack

https://en.wikipedia.org/wiki/Knapsack_problem

The knapsack problem is a combinatorial problem. Given a set of items with a weight and value, determine the number of each item to include in a collection so that the total weight is less than or equal to the given limit and the total value is as large as possible.

Given a set of n items numbered from 1 up to n, each with a weight $w_i$and value v_i, along with a maximum weight capacity W,

$$maximize \sum_{i=0}^{n} v_i x_i$$

$$\sum_{i=1}^{n} w_i x_i \leq W \; and \; x_i \epsilon \{0, 1\}$$

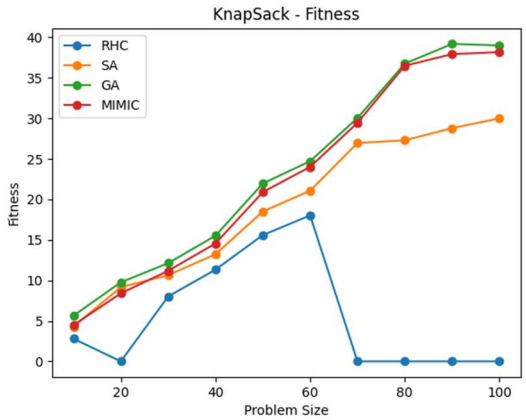
The results for this problem are shown below.

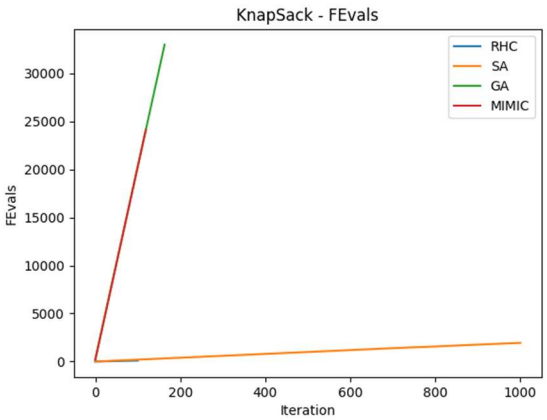
Mimic algorithm does pretty well here. The time taken to derive the global optimum is considerably higher, however, the high fitness and acceptable number of function evaluations make it the one of the better algorithms to solve this problem.

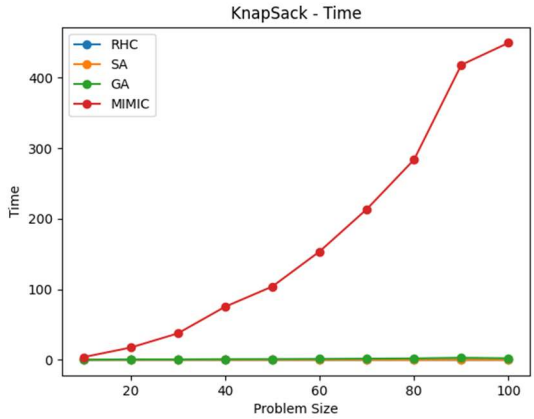
Genetic algorithm also does very well. One can even argue that it is better than mimic as the time taken is way lesser than mimic while determining best fitness values. The function evaluations are higher than mimic.

Random hill climbing and simulated annealing do not do well in determining the global optimum for this problem even though the time taken and function evaluations are minimal. They are unable to traverse and escape the bounds of the local optimums.

This problem does well when the algorithms are able to use past values to move forward. Thus Mimic and Genetic algorithm do well with their respective features of defined structure and crossovers with mutations.







Flip Flop

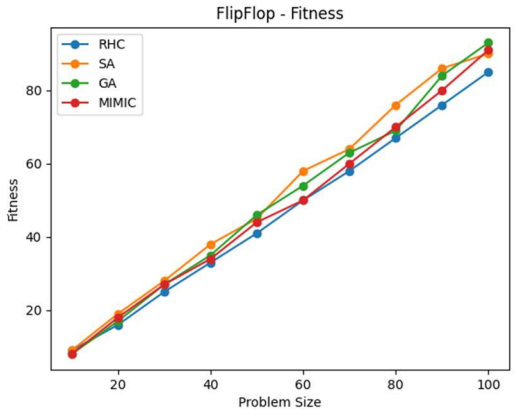
The flip flop problem counts the number of times of bits alternation in a bit string – from a number to any other number in the next digit is counted as 1. The maximum fitness will correspond to a bit string that consists entirely of alternating digits.

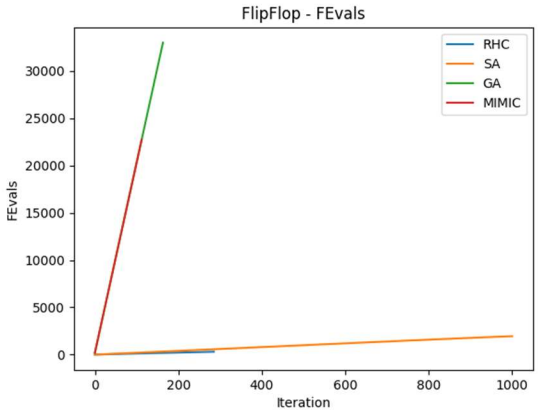
The results for this problem are shown below.

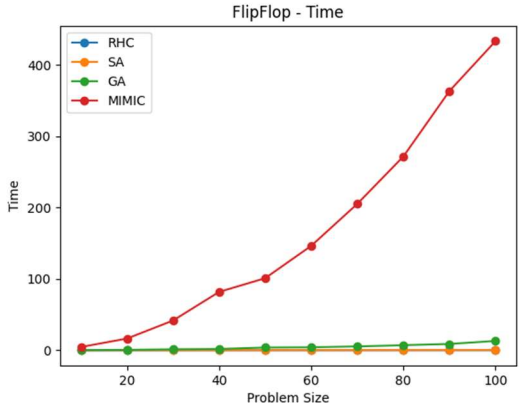
Simulated annealing algorithm does the best as compared to the other three random search algorithms. The number of function evaluations, the time taken and the high fitness score in general make it the best algorithm to solve this problem.

Mimic takes a long time to determine the best fitness values. The genetic algorithm has a higher number of function evaluations.

Random hill climbing performs close to simulated annealing. However, simulated annealing has a higher fitness value.





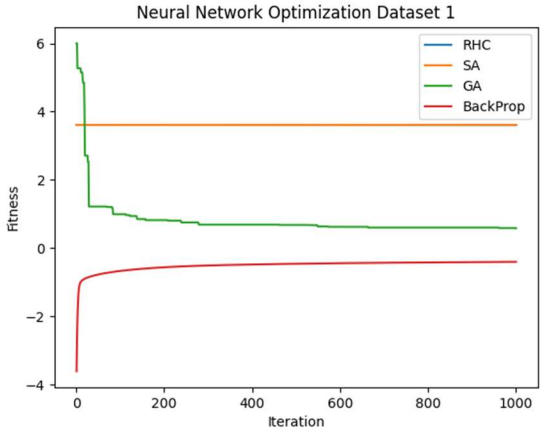


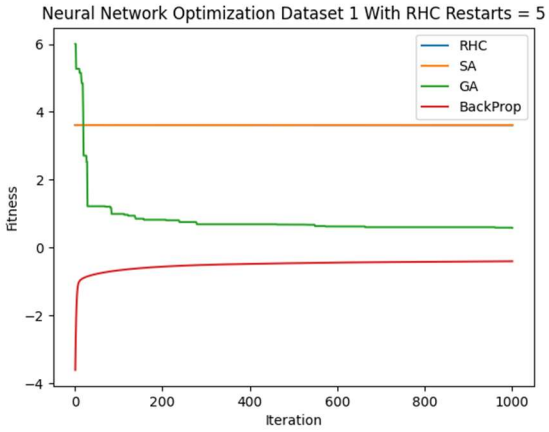
Neural Network Weight Tuning

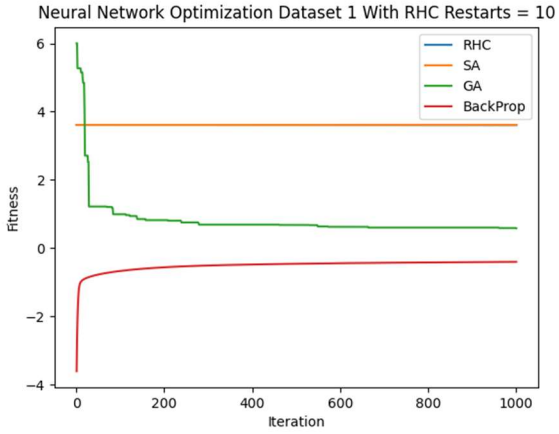
Below are the accuracy scores and model times for the experiments from HW1 and HW2 for neural networks for dataset 1 – car evaluation [5].

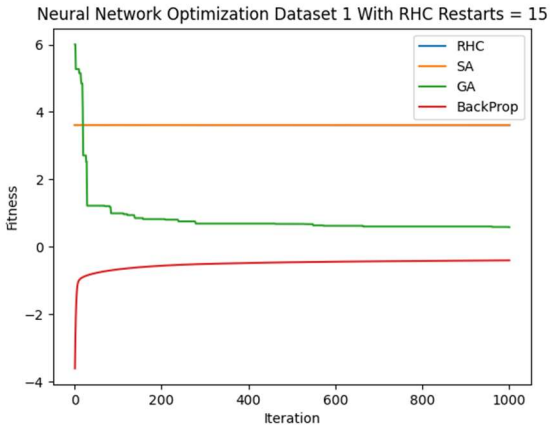
Assignment	Accuracy Score Without Hypertuning		Accuracy Score With Hypertuning	
	In Sample	Out Sample	In Sample	Out Sample
HW1 Scores	0.730355666	0.736030829	0.730355666	0.741811175
HW2 - Random Hill Climb (Restarts = 0)			0.081885856	0.077071291
HW2 - Random Hill Climb (Restarts = 5)			0.081885856	0.077071291
HW2 - Random Hill Climb (Restarts = 10)			0.081885856	0.077071291
HW2 - Random Hill Climb (Restarts = 15)			0.081885856	0.077071291
HW2 - Simulated Annealing			0.081885856	0.077071291
HW2 - Genetic Algorithm			0.70306038	0.693641618
HW2 - Backprop (Gradient Descent)			0.785773366	0.766859345

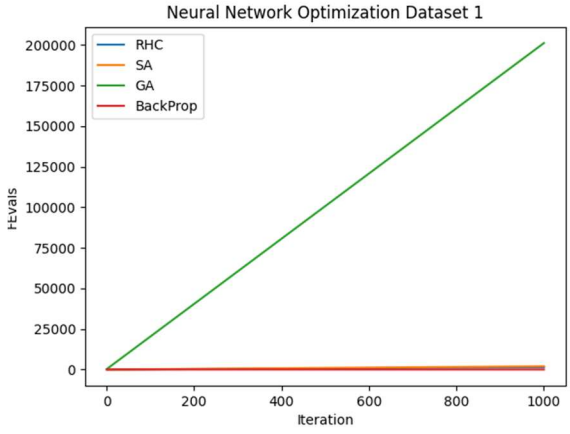
Assignment	Times	
	Learning Time	Query Time
HW2 - Random Hill Climb (Restarts = 0)	3.11116457	0
HW2 - Random Hill Climb (Restarts = 5)	18.70243239	0
HW2 - Random Hill Climb (Restarts = 10)	32.08863211	0
HW2 - Random Hill Climb (Restarts = 15)	45.49785447	0
HW2 - Simulated Annealing	4.095730782	0
HW2 - Genetic Algorithm	455.1071651	0
HW2 - Backprop (Gradient Descent)	3.710525751	0

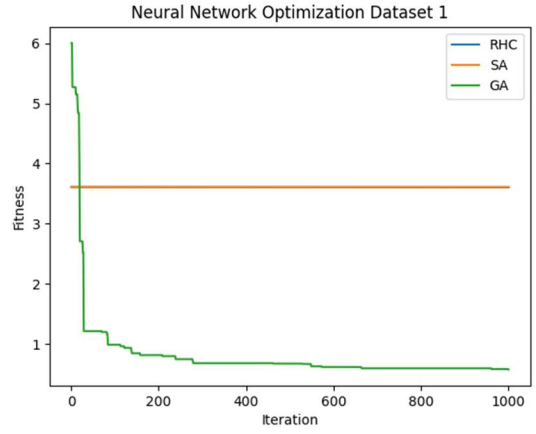












From the results, for the same features as chosen in HW1, the backprop algorithm returns the best results. Random hill climb and simulated annealing do very poorly as they are unable to determine the correct global optimum.

Different restart options were tried for random hill climb but it did not help. The fit times increased from the initial 3 seconds for 0 restarts to 45 seconds for 15 restarts.

Genetic algorithm does considerably better as its able to navigate local optima through the use of its crossover and mutation features. However, the time taken is relatively large. It took 455 seconds to fit this small dataset.

Backprop algorithm works the best. It calculates the gradient of a loss function with respect to the variables of the model. The algorithm allows the information from the cost to flow backwards through the network to compute the gradient effectively. The loss function here is the error of the model, the variables for the function are the weights and the gradients of the error with respect to the weights is the error gradient.

The accuracy has increased from 74% (hyper tuned) to 76%. When considering the original run without hyper tuning, the accuracy has increased from 73% to 76%. The time taken is also small. It took close to 4 seconds to fit the data.

References

1) Randomized Local Search as Successive Estimation of Probability Densities Charles L. Isbell, Jr
2) S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. Technical report, Carnegie Mellon Univerisity, May 1995.
3) Mathews, G. B. (25 June 1897). "On the partition of numbers" (PDF). Proceedings of the London Mathematical Society. 28: 486–490. doi:10.1112/plms/s1-28.1.486.
4) Dantzig, Tobias. Numbers: The Language of Science, 1930
5) Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.