Randomized Optimization

1. Introduction

The purpose of the paper is to explore random search. The following four algorithms are evaluated.

Randomized Hill Climbing (RHC): This locates the local optima by moving towards more optimal neighbors until it reaches a peak. With random restarts, RHC randomizes its starting points and explores other local optima and selects the highest value as global optimum.

Simulated Annealing (SA): This uses a function of initial temperature (T) and cooling rate (C), strikes a balance between exploring new points and exploiting nearby neighbors in search of local optima. Initially, at high temperatures, the algorithm explores by randomly seeking new points and as it cools, it proceeds to evaluate neighbors for local peaks.

Genetic Algorithm (GA): This is inspired by Biology where the initial data evolves by iteratively mating and mutating parts to crossover the best traits and to remove inappropriate qualities. Population (P) is selected among the given data and a percentage of it is mated (ToMate) and a percentage is mutated (ToMutate).

Mutual-Information-Maximizing Input Clustering (MIMIC): MIMIC algorithm remembers prior iterations and uses probability densities to build structure of the solution space and find optima. *Samples*, the number of instances are evaluated and *ToKeep* number are retained at each iteration.

ABAGAIL Java Library is used for analyzing the above algorithms.

2. Analysis of Neural Network using randomized optimization algorithms

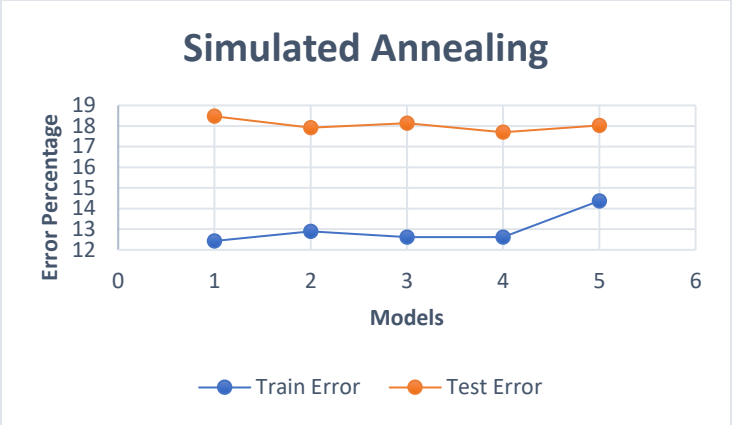
The Adult data set from UCI Machine Learning repository is used for the analysis. There are 3012 adult instances where each instance has 13 attributes which help in deciding whether his income exceeds $50K/year. This was already analyzed using backpropagation which resulted in a neural network with Learning rate=0.1, Momentum=0.2 and 14,3,2 nodes in three hidden layers. Now RHC, SA and GA are used instead of backpropagation to find good weights for the neural network using sum of squared error minimization. The nominal data is converted into numeric and normalized. 70% of the data is used for training and rest 30% for testing. Each model is evaluated by varying the parameters, iterations and training data size.

Each algorithm is made to run for twenty minutes for allowing it to explore the space to reach global optima.

2.1 Simulated Annealing

Starting temperature (T) and Cooling exponent (C) are varied using different combinations of low and high values to find a model with low test error as shown below where model 4 is selected. The low temperature and high cooling rate indicate that some exploration is important at the start, but high exploitation is important to select the optimal neural network weights. No structure is kept around but has a clear probability distribution unlike RHC and GA.

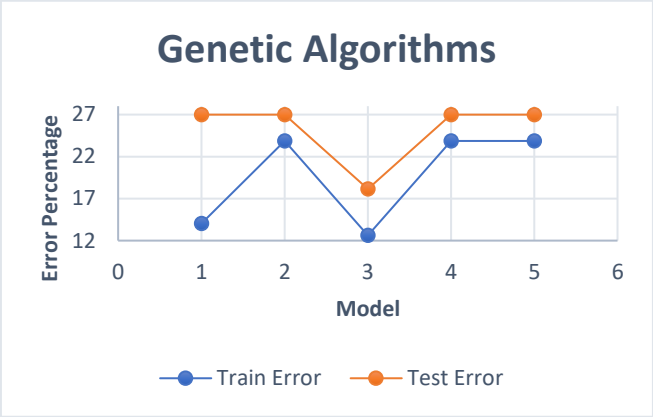
Model	T	C
1	1E15	0.5
2	1E15	0.95
3	100	0.05
4	100	0.95
5	100	0.5



2.2 Genetic Algorithm

Population Size (P), number to mate(toMate) and mutate (toMutate) are varied and the models produced the errors as shown below where model 3 with low test error is selected.

Model	P	toMate	toMutate
1	0.1N	0.04N	0.03N
2	0.5N	0.05N	0.004N
3	N	0.2N	0.15N
4	0.7N	0.1N	0.09N
5	2N	N	0.5N

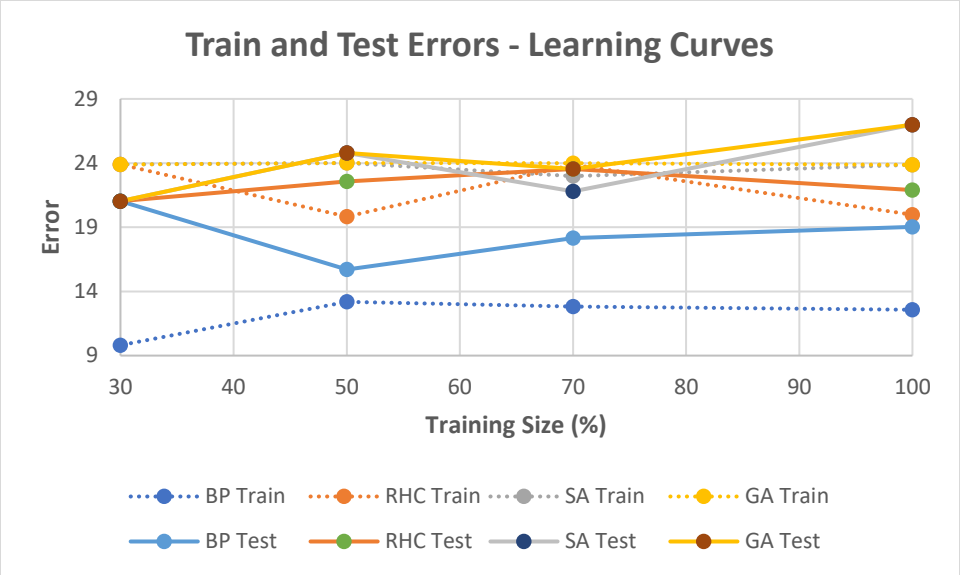


In genetic algorithm, mutation offers exploration while cross-over drives the population to converge on good solutions (exploitation). Subsequently, there is an equilibrium between cross-over trying to converge and mutation trying to avoid convergence and explore more areas. There exists neither a kept around structure nor a probability distribution for GA.

3. Analysis

The selected models of SA and GA, RHC are ran and are compared with already generated backpropagation model.

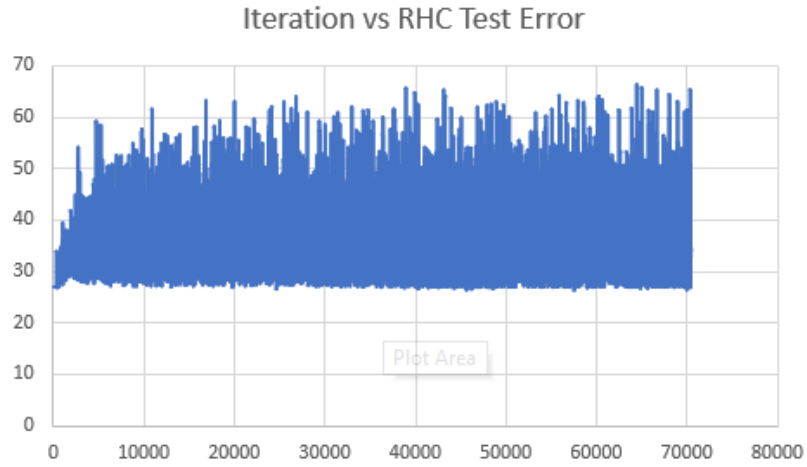
Algorithm	Train Error	Test Error	Function Evaluations	Iterations	Training Time
RHC	19.972	21.903	70402	70401	1200.060s
SA	23.861	26.991	40045	40044	1200.195s
GA	23.861	26.991	230628	348	1202.924s
Backprop	12.57	19.026			



The learning curves shows that RHC has performance approximately equal to backpropagation. They are analyzed below in detail.

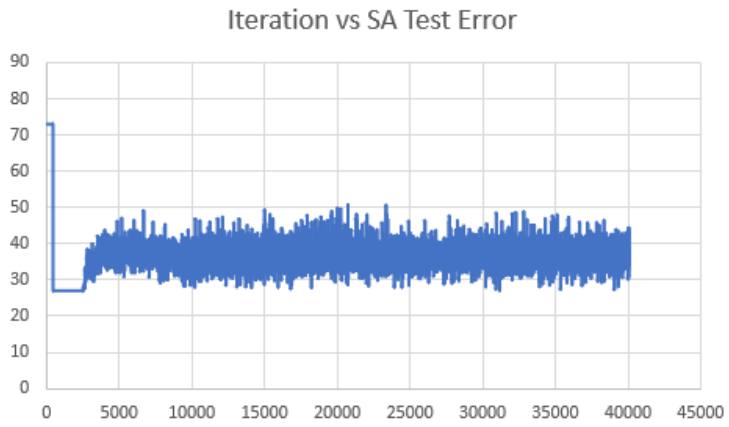
3.1 Randomized Hill Climbing

RHC performs better among all the three optimization algorithms used. This is due to its inherent greedy nature. RHC works well in neural networks because the weights are continuous values and RHC implements a similar principle to gradient descent that tests neighboring points to incrementally climb towards a local optimum or settle at a peak.



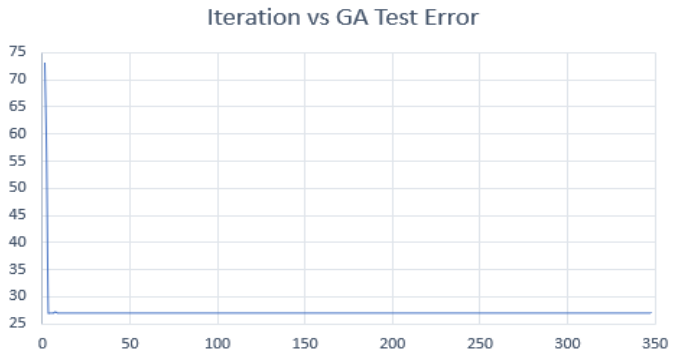
It can be observed in the above graph where the value represents the local optima reached at each iteration which resulted in high error (peaks) compared with the better optima it found along the search space. Since it was run for 20 minutes, the algorithm didn't terminate early when it found local optima instead it expanded the search to check if any better points are present in the vicinity.

3.2 Simulated Annealing



In contrast to the results from RHC, we can observe that at least in the first few iterations, SA tries to look around for the viable points. However, after the temperature became low it does behave like RHC but with better optimal values that can be observed from the above graph. From the number of function evaluations, we can observe that it has 43% lower function evaluations and iterations than RHC. Despite this, SA performed near to RHC, that reflects the hybrid nature of the algorithm, i.e., allowing for random walk behavior at higher temperatures and RHC behavior at lower temperatures.

3.3 Genetic Algorithm



From the above graph, it's anticipated that GA would have terminated early with fewer generations. But, it continued to find better points that resulted in a similar model to SA with 99% fewer iterations but 4.76% more function calls. This performance in GA is mostly related to the cross over model structural changes of connections of the layers and therefore lead to better results.

The graph shows the way behavior differs from greedy nature of RHC and SA. By observing the graphs, GA appears to be a better choice under restricted resources.

4. Neural Network Analysis Conclusion

It has been shown that neural network weight training can be formulated as an optimization problem. The presented results infer that RHC's performance is almost near to Backpropagation. SA and GA can be evaluated further by trying out different models to enhance their performance.

Bias to identify here is *preference bias* for simple models and those that compute faster to enable a higher volume of iteration while providing information to learn. Another bias to consider is *restriction bias* which is inherent to the dataset which looks only at numeric values. Another aspect of restriction bias is the way the neural network incrementally changes the weights. In the case of the implemented code, the best model is determined from the limited list of possible parameter values. Due to the inherent nature of optimization problem solving technique to find best value of a fitness function, it can be more suited for regression compared to the classification performed here.

5. Randomized Optimization

Flip flop, Travelling Salesman and Knapsack problems are chosen to demonstrate the strengths of RHC, SA, GA and MIMIC randomized optimization algorithms which are part of unsupervised learning. These are evaluated using iteration cut-off which is not ideal for solving real optimization problems. But it serves the current purpose where it allows analysis of relative performance of the algorithms.

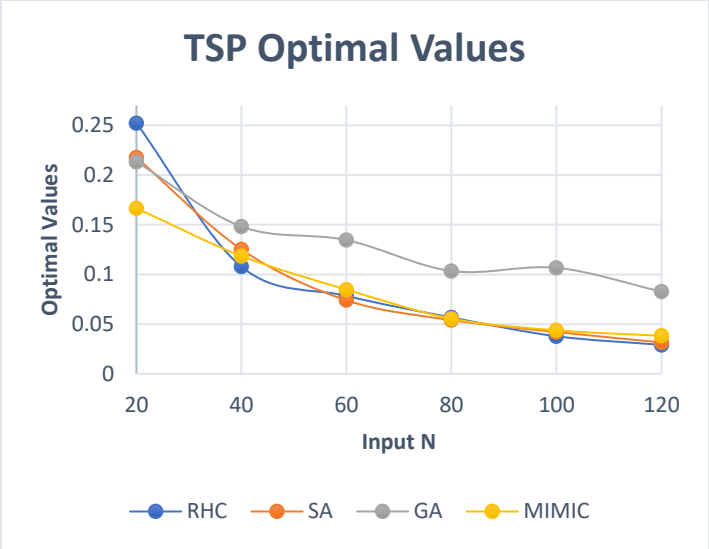
5.1 Travelling Salesman

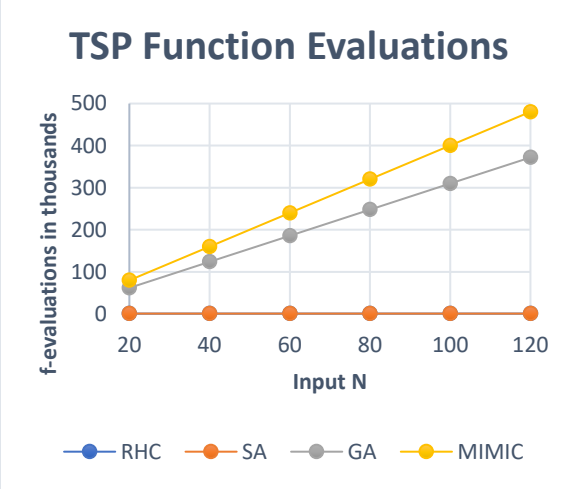
This is a famous NP Hard problem. Given a set of cities (N) and distance between every pair, the problem is to find the shortest route to visit all the cities once. This problem is found to be interesting to analyze how a problem with varying parameters and an underlying assumed structure plays a role in performance of randomized algorithms.

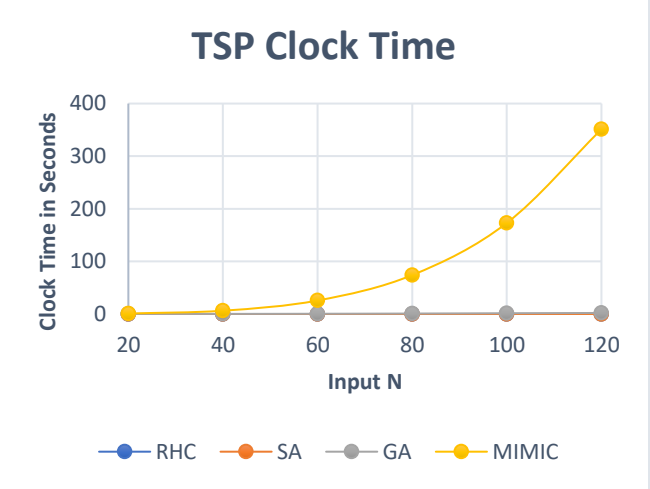
For N=100 and 1000 iterations each, three algorithms are evaluated with different parameters and the models with highest optimal values is selected (in bold). This was used to demonstrate the importance of choice of parameters which resulted in range of optimal values. Optimal value is 1/totalDistanceToCover calculated by the evaluation function.

Simulated Annealing			**Genetic Algorithm**				**MIMIC**		
T	**C**	**Optima**	**P**	**To Mate**	**To Mutate**	**Optima**	**Samples**	**To Keep**	**Optima**
100	0.7	0.0846	0.2N	0.05N	0.04N	0.0621	0.2N	0.05N	0.04811
100	0.8	0.0725	0.5N	0.2N	0.05N	0.0799	0.5N	0.2N	0.0353
100	0.95	0.0695	N	0.3N	0.1N	0.0886	N	0.3N	0.0419
1E15	0.95	0.06732	2N	N	0.3N	0.0935	2N	N	0.0436
1E15	0.8	0.0857	**4N**	**3N**	**0.4N**	**0.1070**	**4N**	**2N**	**0.0514**
1E15	0.7	0.09074							
1E15	**0.5**	**0.0954**							
1E15	0.4	0.0786							
1E15	0.3	0.07719							

Each selected model and RHC are evaluated with N=20, 40, 60, 80, 100, 120 and the results are shown in the below graphs.







As N is increased, GA performed exceptionally well in terms of optimal value. Whereas, RHC, SA and MIMIC with approximate similar values. GA has created offspring by picking different cities from the parent city positions until an offspring has no empty positions which leads to an optimal solution eventually with all positions. This simple method with mutation and crossover made GA to perform the best. In terms of fitness function evaluations RHC and SA have very less number compared to GA and MIMIC, where MIMIC being the highest. Whereas in terms of clock time, RHC, SA and GA are minimal compared with MIMIC. It is expected that RHC and SA does not perform well as they are better suited for continuous functions.

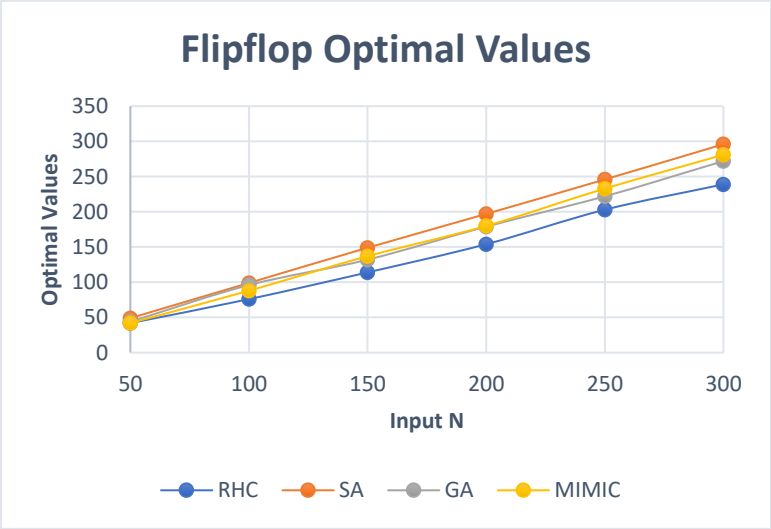
5.2 Flip flop

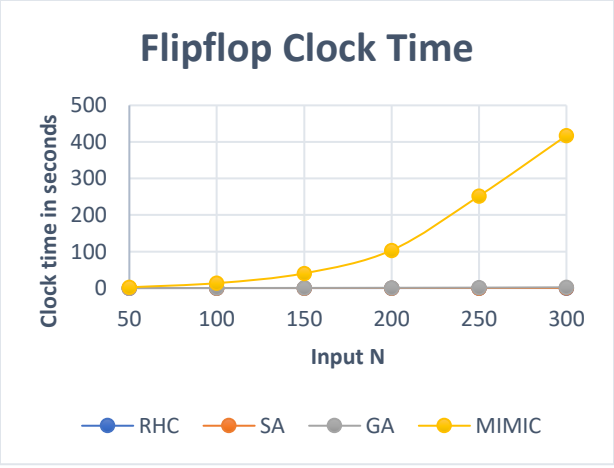
Given a N dimension input vector, a flip flop evaluation function counts the pairs of alternate 1 and 0 or 0 and 1. It has two global optima where there are all alternate 1s and 0s, each starting with 1 or 0. The solution is intuitive but not obvious for a machine. This problem is picked to analyze the performance of randomized algorithms where the problem evaluation is a simple and pure functional evaluation.

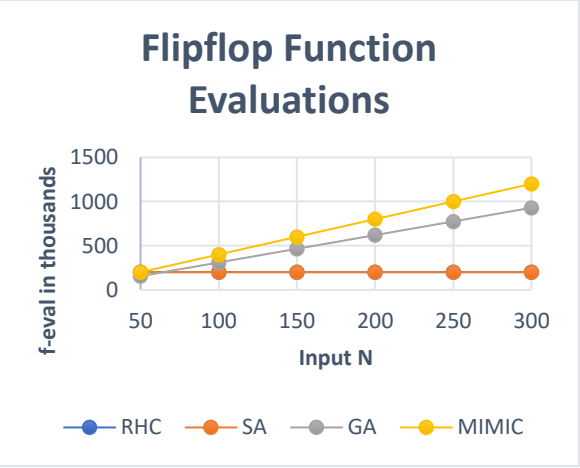
For N=50, 100, 150, 200, 250, 300 and with 200,000 iterations in RHC and SA, 1000 in GA and MIMIC, the following parameters are evaluated to find a model in each algorithm with the best optimal value (in bold).

SA		GA			MIMIC	
T	**Cooling exponent**	**P**	**To Mate**	**To Mutate**	**Samples**	**To Keep**
100	0.5	0.2N	0.05N	0.04N	0.2N	0.05N
100	0.7	0.5N	0.2N	0.05N	0.5N	0.2N
100	**0.95**	N	0.3N	0.1N	N	0.3N
1E15	0.95	2N	N	0.3N	2N	N
1E15	0.7	**4N**	**3N**	**0.4N**	**4N**	**2N**
1E15	0.5					

Below graphs show the optimal values produced, time taken and number of function evaluations performed for each input vector of size N.







Clearly SA is the best algorithm in terms of all the three measures which reflects the greedy nature playing a good role in calculating the fitness function which is counting the different values next to each other. For N=50 and 300 it in fact finds the global optimum. But RHC didn't work as expected in par with SA, and this may be due to the SA's nature of heating and cooling which gained an advantage to find the better optima in evaluating the fitness function which involves counting. Looking at function evaluations and iterations, MIMIC has the highest number compared to GA, SA and RHC where SA and RHC both being the lowest. Whereas, MIMIC being slightly better in terms of optimal value compared with GA, that shows the structure and history of the problem being suitable for MIMIC, while GA's crossover operation didn't gain much here.

This highlights the situation where SA would perform well, compared to MIMIC and GA. When there is some form of structure, GA and MIMIC's representation allow it to find better points, whereas when the problem at hand lacks such a structure, RHC and SA would perform better.

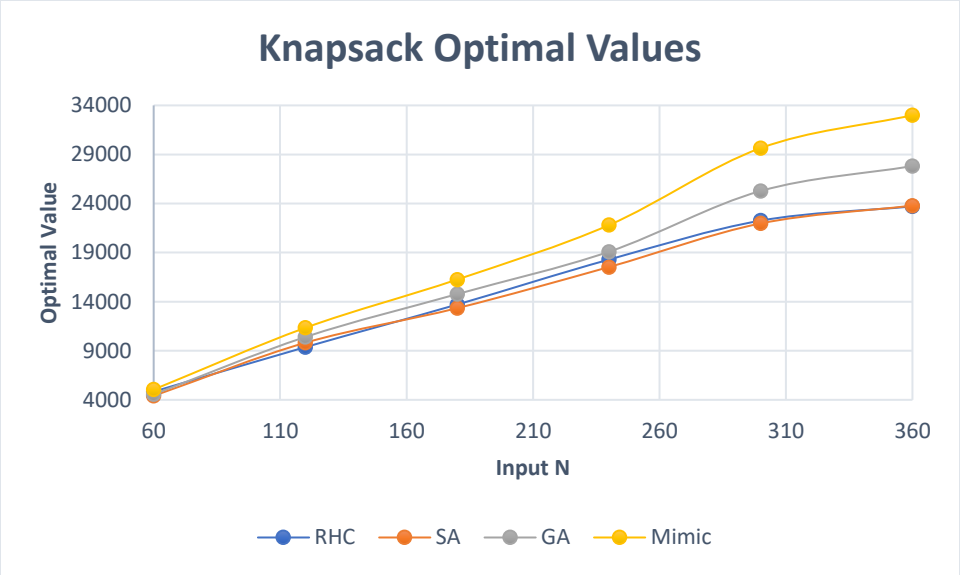
5.3 Knapsack

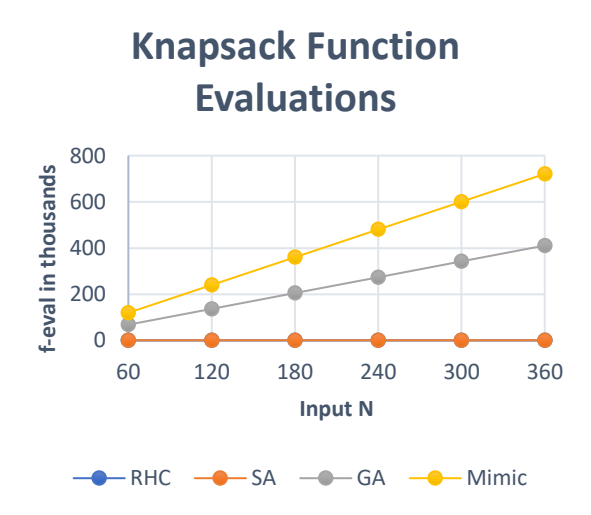
Knapsack is a classic NP Complete problem. Given a set of items (N), each with a weight and a value, the problem is to determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It is chosen as the changes in configuration would lead to different solutions mean that a greedy algorithm is unlikely to find the optimal knapsack solution and to analyze how structured optimization problem is solved using different algorithms.

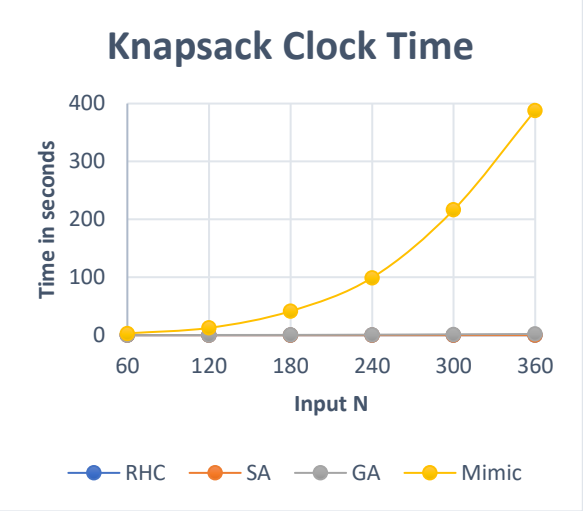
Following parameters are evaluated and the model which produced the knapsack with highest value (in bold) is chosen.

Simulated Annealing			**Genetic Algorithm**				**MIMIC**		
T	**C**	**Optima**	**P**	**toMate**	**toMutate**	**Optima**	**Samples**	**To Keep**	**Optima**
100	0.5	1975.43	0.2N	0.05N	0.04N	1479.422	0.2N	0.05N	1729.271
100	0.7	1978.481	0.5N	0.2N	0.05N	1325.024	0.5N	0.2N	1993.088
100	0.95	1153.779	N	0.3N	0.1N	1193.516	N	0.3N	1256.740
1E15	**0.95**	**2309.396**	**2N**	**N**	**0.3N**	**2126.94**	**2N**	**N**	**2294.51**
1E15	0.7	1687.539	4N	3N	0.4N	1977.4	4N	2N	2013.5
1E15	0.5	1636.30							

The chosen models are evaluated with values of N=60, 120, 180, 240, 300, 360, 4 copies of each item, and random weights and values for each item, ***N * max_weight (50)*** * ***copies_each (4) * 0.4*** as the maximum sack value and with 1000 iterations each, produced the following results.







MIMIC performed well in terms of optimal values whereas number of function evaluations and clock time being the highest. The improvement in performance is significant as N value is increased. This is due to the MIMIC's strength with structured optimization candidates, the value of one parameter depends on the value of another, and its ability to estimate the shape of the fitness landscape rather quickly. This problem is not completely dependent on structure, but the presence of a given item in the knapsack influences whether another item can/should go in the sack.

RHC and SA has the worst performance compared with GA and MIMIC. This is most likely because the solution to the problem depends on the structure of the binary candidates, which RHC and SA do not handle well and the performance highlights the greedy nature of RHC and SA.

One reason why GA performs comparably to MIMIC and well compared with RHC and SA for the knapsack problem is due to GA's attempt to capture structure. The ability to perform the crossover operation would be equivalent to sampling from a distribution, thus allowing GA to match the underlying structure of the problem. This allows GA to perform stronger than RHC and SA which does not utilize the structure in the optimization problem.

Therefore, for problems where retaining history and structure would assist in creating a better optimization problem, MIMIC or GA would be a better performing algorithm, as indicated by Knapsack and Travelling salesman problems. However, when the optimal point is random over a wide range of values RHC and SA may be the better solution as shown by Flipflop problem.

6. Conclusion

From the above analysis, we can see the close relationship between RHC and SA. For complex problems where structure is important, GA or MIMIC would be a preferred choice. For exhaustive search problems or when function evaluation is costly, RHC or SA would be desirable. MIMIC would be the most preferable when retention of history is helpful and structure of the problem is significant. Therefore, it depends on the nature of the problem and other important computation factors to determine the algorithm to be used.