Assignment 2: Random Optimization

Minming Zhao

In this study, we will have two parts, Part I is to use Random Optimization problem to find best Neural Networks weights for the student alcohol dataset in Assignment 1, and Part II is to compare three Random Optimization algorithms, i.e. Simulated Annealing, Genetic Algorithm and MIMIC. Both studies used ABAGAIL library.

Part I: Find for NN weights

The Assignment #1 used student data to predict whether they are inclined to be alcohol addictive. The Neural Network in Assignment used 2 layers and the training accuracy is 99.7% while testing accuracy is 70.8%.

In this assignment, we studied three random search algorithms, i.e. randomized hill climbing, simulated annealing and genetic algorithm. Training accuracy, training time are studied. I modified the code to be able to take in training dataset and testing dataset separately. Hence the accuracy on training set and testing set are both displayed below. To be noted, here iteration are all 1000.

	Training				Testing		
	Correctly classified	Incorrectly classified	Accuracy	time	Correctly classified	Incorrectly classified	Accuracy
RHC	274	42	86.709%	0.933	59	20	76.684%
SA	260	56	82.278%	0.797	61	18	77.215%
GA	260	56	82.278%	3.726	61	18	77.215%
Backpropagation			99.7%				70.8%

We could take a look at how fast each algorithm converges by its error per each iterations as Fig 1.

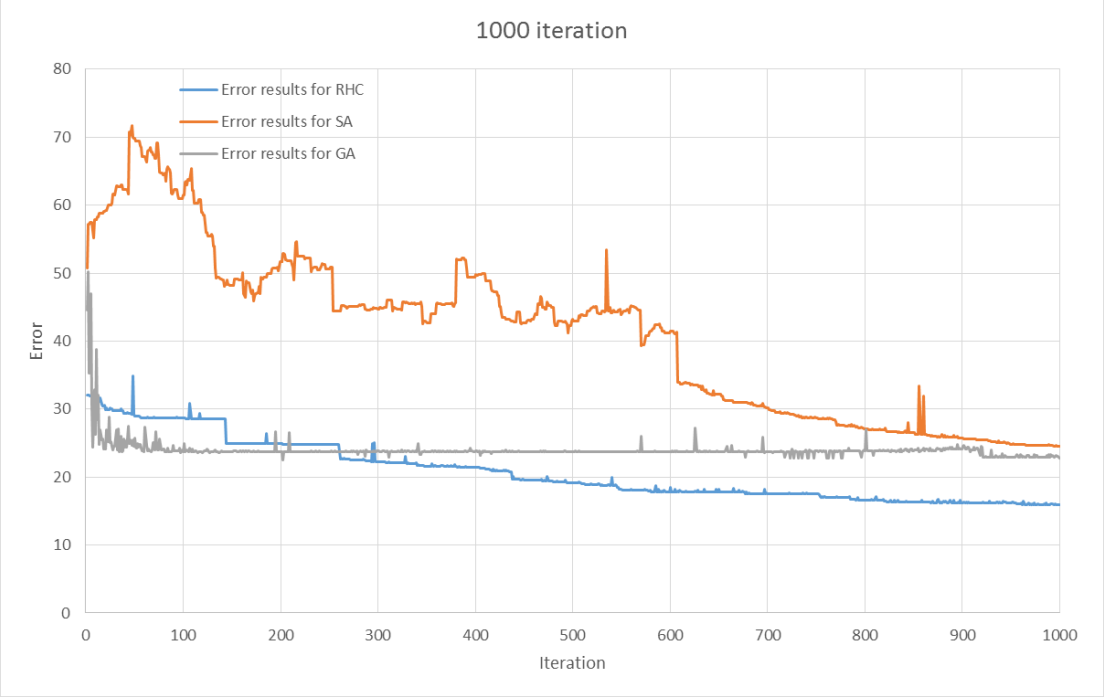


Fig.1: Error per iteration for three random optimization algorithms

As we can see Generic Algorithm converges fastest in terms of iterations although it takes most of time per each iteration. This is because Genetic Algorithm has to compute a population size (in this case I used k=50) of function fitness evaluation each iterations while other tow algorithms only compute once each iteration (but within acceptable time region). Hence for same iterations such as 1000, Generic Algorithm achieving optima requires a lot fewer iterations although total training time is not necessarily shorter. Actually as later study shows Table 2., GA achieves the same optima even with 200 iterations. I changed the population size to 200, 100, 50 and 25, 10, all of them could converge to the same optima using similar time. And the Mate number (25,10,5,1) and Mutate number (25,10,5,1) is changed to study their impact, all of these setting converge to the same optima.

Randomized Hill Climbing seems to converge the second fastest in terms of iterations but it actually did not still get to the optima as other two algorithm achieved. Its higher training accuracy and lower testing accuracy indicates Randomized Hill Climbing with 1000 iterations seems to be overfitting already. So we change the iterations to 200,500,1000,2000,3000 to see the optimization results. It indicates iterations of 200 to 500 can lead RHC to the optima otherwise it goes to overfitting.

	RHC		SA		GA	
Iterations	Training	Testing	Training	Testing	Training	Testing
3000	90.19%	70.89%	87.34%	75.95%	82.278%	77.215%
2000	85.76%	70.89%	85.13%	75.95%	82.278%	77.215%
1000	86.709%	76.684%	82.278%	77.215%	82.278%	77.215%
500	82.278%	77.215%	43.04%	50.63%	82.278%	77.215%
200	82.278%	77.215%	55.38%	50.63%	82.278%	77.215%

Fig.2: Iterations impact on the training and testing accuracy.

Simulated Annealing converges to the same optima as GA does at iteration 1000, but it went through some high error region at the half of the whole iterations. This is due to the nature of SA algorithm, because by its definition, it will accept some high error point as long as the probability is close to 1. Especially when we are at the beginning of iterations, SA is inclined to accept next random point. Here the T is set at 1E11. If we have smaller T, we will be more like Randomized Hill Climbing, which means in this case, we will converges faster. This is been tried as Fig 2 shows.

This time we set iterations to 200 for SA, but varying the T parameter to be 1E11, 1E5, 50, 5, 1, 0. From the lecture, we know when T = 0, SA is essentially Randomized Hill Climbing.

SA cooling = 0.95	**Training**				**Testing**		
Iteration = 200	Correctly classified	Incorrectly classified	Accuracy	time	Correctly classified	Incorrectly classified	Accuracy
T=1E11	80	236	25.316%	0.189	19	60	24.051%
T=1E5	82	234	25.949%	0.180	25	54	31.646%
T=50	56	260	17.722%	0.180	18	61	22.785%
T=5	221	105	66.772%	0.176	35	44	44.304%
T=1	260	56	82.278%	3.726	61	18	77.215%

Fig.3: Temperature parameter impact on SA algorithm

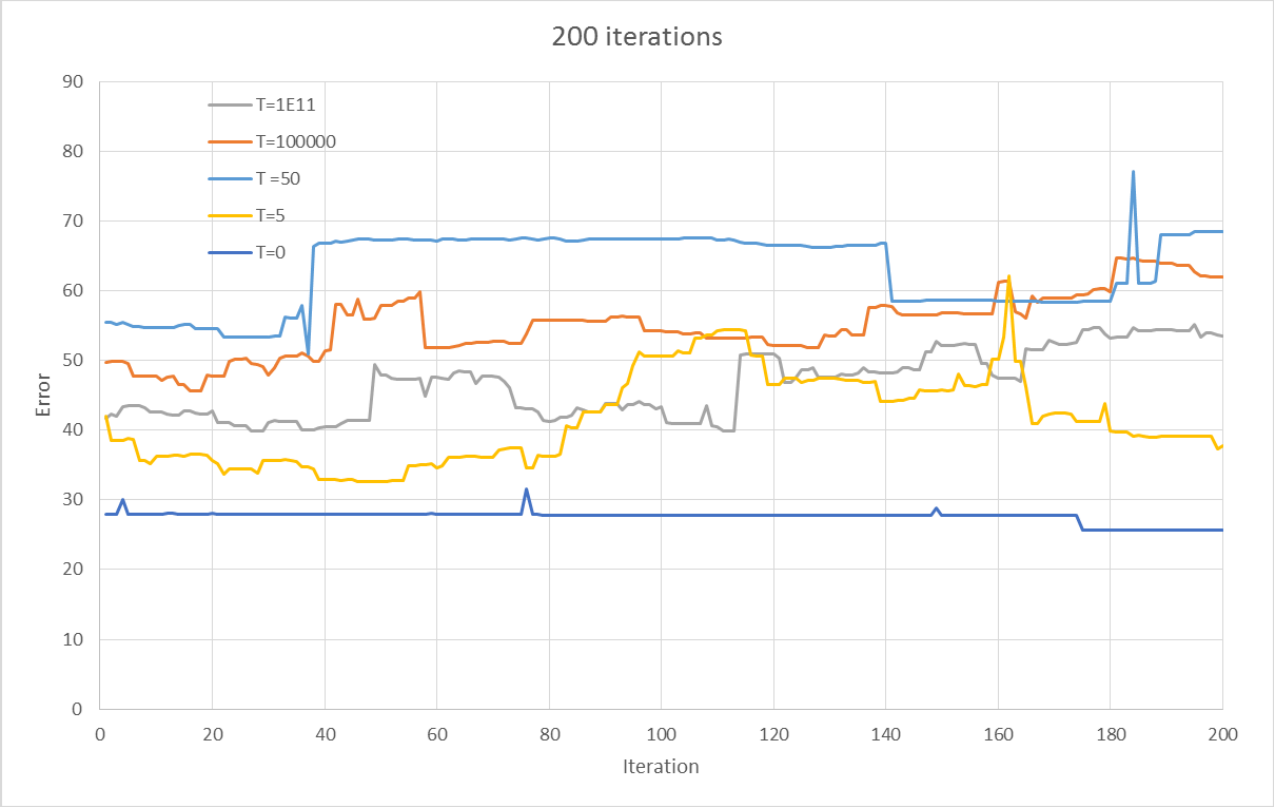


Fig.4: Temperature parameter impact on SA algorithm

To be noted that weights in a neural network are continuous and real-valued instead of discrete. In practice, this means that in a single iteration of hill climbing each variable will be incremented by some value between 0 and the endpoint of the line search, whereas in discrete case, it will be either 0 or a fixed quantity.

Compared to the backpropagation method in Assignment 1, we were able to reach testing accuracy 70.8% and apparently backpropagation in previous Assignment were not global optimum yet. If we had played different parameters in layers or nodes we may be able to find better fit results.

Below is the weights from each random optimization algorithms for the neural networks.

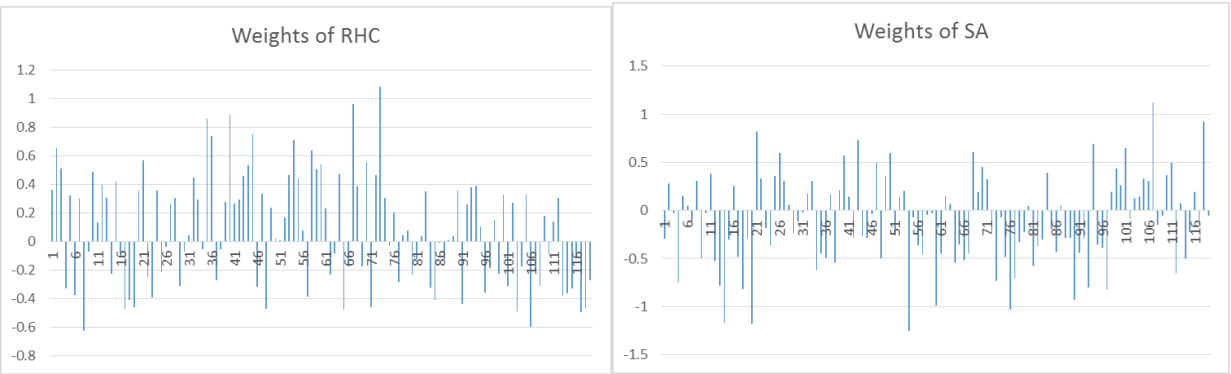


Fig. 5 & 6: Weights of NN for RHC and SA

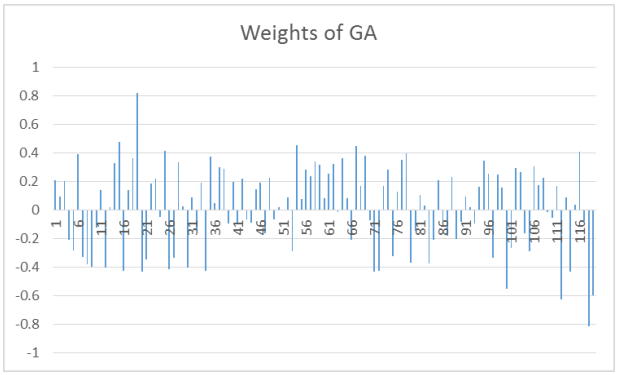


Fig.7: Weights of NN for GA

Part 2: Optimization Problems

Here we come up three optimization problems with discrete valued parameter space and use all four random optimization algorithms to solve. And we will discuss each algorithms performance metric in optima and processing time for each problem.

2.1. Four peaks

The Four peak problem was populated by Baluja and Caruana, 1995. The problem definition is to consider a fitness class functions for bit string containing N bits and parameterized by T.

$$z(x) = Number\ of\ contiguous\ Zeros\ ending\ in\ Position\ N$$

$$o(x) = Number\ of\ contiguous\ Zeros\ starting\ in\ Position\ 1$$

$$Reward = \begin{cases} N & \text{if o(x)>T AND z(x) > T} \\ 0 & \text{otherwise} \end{cases}$$

$$f(x) = max(o(x), z(x)) + Reward$$

It is called four peaks problem because it has two global maxima and two suboptimal local maxima. The global optimum is 2N-T+1, by having (N-T+1) 1's followed by (T+1) zeros or (N-T+1) 0's followed by (T+1) 1's, while two local optimums are just N, by having bits all 0's or 1's. For this study, I select the T = N/5.

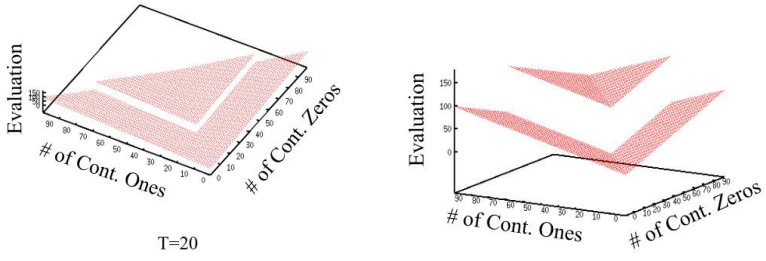


Fig.5: Two views of the same four peaks problem.

2.2. Count ones

This problem is to count ones in a bit vector of length N. It is a discrete function and can be defined by maximizing

$$f(\boldsymbol{X}) = \sum_{i=0}^{N} g(x_i) \quad \text{Where } g(x_i) = \begin{cases} 1 \text{ if } x_i = 1 \\ 0 \text{ otherwise} \end{cases}$$

Apparently this function is very obvious to a human but it may not be to a machine since the function oscillates with integer representation.

2.3. Bounded Knapsack Problem

The Knapsack problem is very famous for combinatorial optimization. The problem statement is that, given a set of items, each with copies c_i, with a weight w_i and a value v_i, 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 W and the total value is maximized. It can formulated to maximize

$$MAX\left(\sum_{i=1}^{N} v_i x_i\right) \; subject\ to \sum_{i=1}^{N} w_i x_i \leq W, \;\; x \in \{1,2,3,\dots,c_i\}$$

2.4. Selecting parameters for each algorithm

We will need to compare Simulated Annealing, Genetic Algorithm and MMIC, and all these three contains parameters. The difficulty in comparing them is that different control parameters such as population size, mutation rate, cooling etc. can affect the performance of each algorithms. In this study, we will run three algorithms 10 times and average the results. The parameters for SA, GA and MIMIC are varied:

For SA,

- Cooling schedules: 0.7, 0.8, 0.95

For GA, five parameters are varied:

- Population size: 50%, 100%, 200%, 400% of Problem size
- Crossover Type: One point, two point, Uniform
- Crossover Rate: 10%, 40%, 50%, 80%, 100%
- Mutation Rate: 0.001, 0.05, 0.2, 0.5, 0.8, 1

For MIMIC, two parameters are varied

- # of samples taken each iteration: 40, 60, 80, 100, 120, 140, 200
- # of samples to be kept each iteration: 10%, 20%, 50%, 70%, 90%

2.5 Results

In this study, ABAGAIL library was used. The codes in the library has been modified and the instructions can be found in README.

2.5.1 Results of Four peaks

Bitstring length N is varied here, N = {20, 40, 60, 80, 120}. The average maximum value found by the each algorithm and the training model time are displayed below. From the definition of the problem, we know the global optima is 2N-T-1.

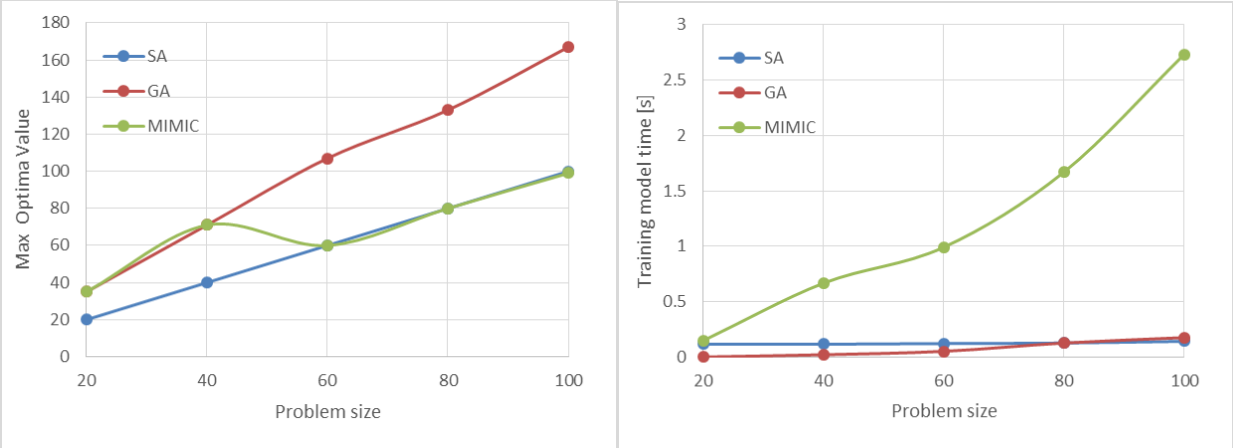


Fig.8 & 9. Maximum optimal values and processing time for each random algorithms for Four Peaks.

As we can see, apparently GA performs best here because GA obtain the maximum value for the objective function while still being able to maintain relatively fast speed. In fact, on problem size 20, 40, 60, GA actually finds the global maximum. MIMIC did relatively good job when problem size is 20 and 40, although it takes much longer time to converge. When problem size increases, the processing time grows exponentially for MIMIC.

However, RHC and SA only managed to reach the local maxima for all the problem sizes. This is because both RHC and SA is greedy algorithm, inclining to accept better neighbor while reject neighbors which performs worse than current points. When there is a big gap between two peaks, the greedy nature of RHC and SA stuck on the local optima. In contrast, MIMIC could record the structure and history could allow it to perform better than local optima sometimes, but GA could do the crossover in this case, has allow it to be the best to find the global optima.

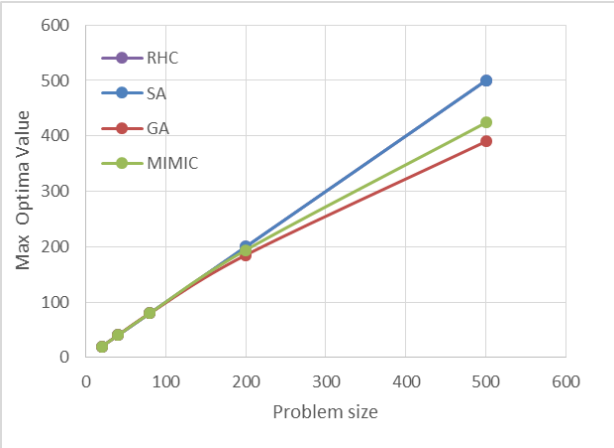
2.5.2 Results of Count Ones

We have Count ones N = {20, 40, 80, 200, 500}, where N is the size of bitstring on which we would like to find the maximum ones. For every problem size, apparently RHC and SA algorithm can reach the global optima within minimum processing time and in fact, SA performs faster than RHC.

MIMIC performs a bit worse then RHC and SA, while GA's optimal solution becomes degrading when problem size grows. And as expected, when problem size grows, MIMIC need exponentially growing time, while GA still could maintain relatively acceptable time, but still much slower than RHC and SA.

This problem is designed such way that there is no locality nor structure where MIMIC used to assist to perform, that is why MIMIC does not performs best. Also this problem does not have multiple local optima's therefore RHC and SA could be excellent in this case without worrying about similar situation like Four Peaks problem. Because there is not much benefit from mutation, again it is due to no locality or benefit on segment of bistring, GA could not gain benefit neither.

In short summary, Count ones shows SA and RHC could perform well than GA and MIMIC when there is no locality or structure. When there is structure information, GA and MIMIC allow to find better points and not stuck at local optima.



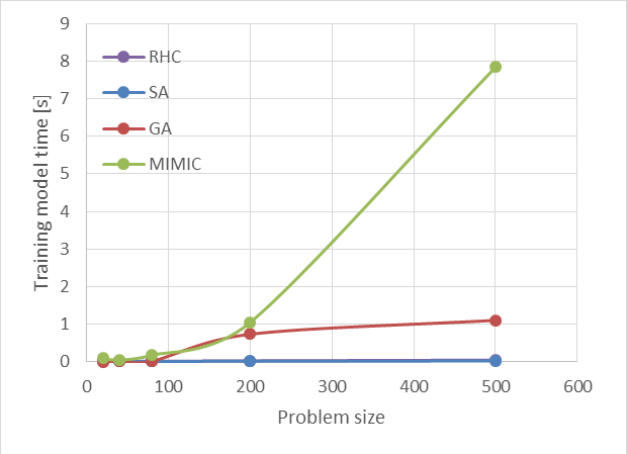
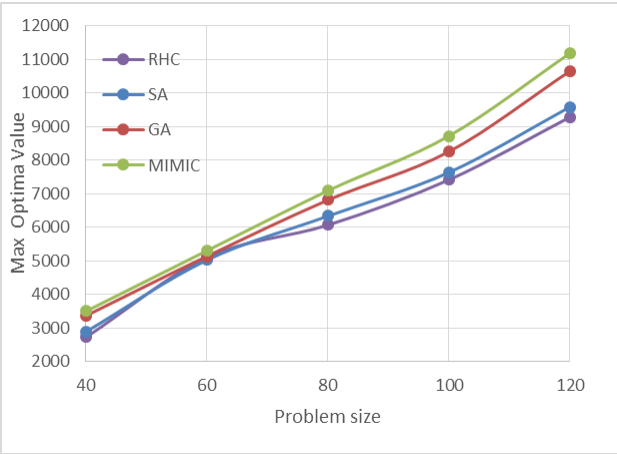


Fig. 10 & 11. Maximum optimal values and processing time for each random algorithms for Count Ones.

2.5.3 Knapsack test

The knapsack problem is configured by varying the item numbers N, {40,60,80,100,120}, the maximum weight and maximum volume of the items are held constant at 50. and the weights and volume of each item is still kept randomized for each run.
The graph below shows that MIMIC performs best of these two algorithms as it reaches the larger values than the rest 3 algorithms. It is because Knapsack does not have analytical solution and it is a NP hard optimization problem. Overall MIMIC is the best, GA is second, while SA and RHC are worst and no much different between them. Knapsack shows greedy nature of RHC and SA again, as we can see RHC and SA groups together while MIMIC and GA seems more similar. The reason that GA still performs relatively acceptable in this problem because GA's nature to "capture structure by an ad hoc embedding of the paramters onto a line i.e. Chromosome" (De Bonet, Isbell, and Viola 1997). The crossover operation of GA is something like sampling from a distribution in MIMIC, which helped GA to have some underlying structure match of the problem. and that is why GA performs stronger than RHC and SA which does not look at the structure in the problem.



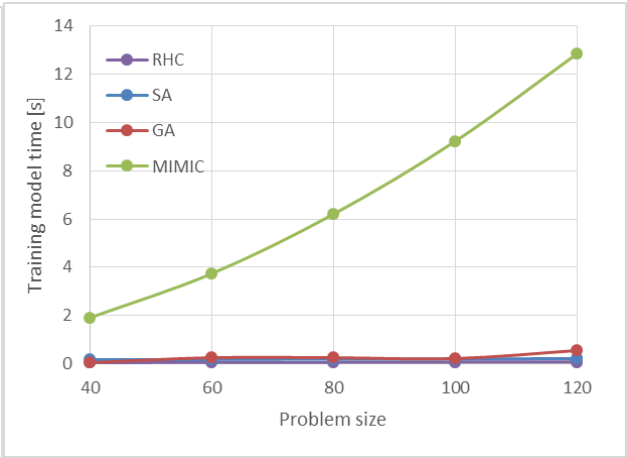


Fig. 12 & 13: Maximum optimal values and time for each random algorithms for Knapsack test

2.6 Conclusion

Therefore in conclusion, for problems which retain history and structure, MIMIC and GA would perform better while MIMIC is even better but need exponentially grown time to solve the problem, just as Knapsack problem shown. For the problems where there is multiple local optima and global optima, such as Four Peaks, GA performs best and MIMIC is relatively okay, while RHC and SA will stuck at the local optima due to their greedy nature. In contrast, for problems which has optimal point random over a wide range and no locality or structure exists, RHC and SA performs best within shortest time.

Eventually it will depend on the nature of each problem and what particular factors are most interesting to solve, thus to determine which of the four random optimization algorithm will be the best to pick. normally RHC and SA can finish the optimization fairly quickly with potential sacrifice on stuck to local optima if the problem relates to, however if the function is very expensive to compute and time constraint is not in too much concern, then MIMIC would become a best one, GA can be selected for cases in the between.

3. References

1. Baluja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. Technical report, Carnegie Mellon Univerisity.
2. De Bonet, JS., Isbell C, and Viola P (1997). MIMIC: Finding Optima by Estimating Probability Densities. Massachusetts Institute of Technology
3. https://en.wikipedia.org/wiki/Knapsack_problem
4. http://abagail.readthedocs.io/en/latest/index.html