Wenjun Wu CS 4641
March 10, 2017 Project 2

Analysis of Randomized Optimization

Introduction

This report explores the utility of randomized optimization (RO) methods, which include Randomized Hill Climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA) and Mutual-Information-Maximizing Input Clustering (MIMIC). To determine their functionality, strength and weaknesses, experiments with three optimization problems were performed with different parameter settings and iterations. These three problems are: Knapsack Problem (KP), the Traveling Salesman Problem (TSP) and the Count Ones Problem (COP). Besides, he randomized optimization algorithms except MIMIC were also used to find the optimized weights for a neural network.

Optimization Techniques

1. Randomized Hill Climbing (RHC)

RHC is an algorithm that built upon the hill climbing (HC) algorithm. It iteratively performs hill-climbing (moves towards increasing value) with randomized starting points. Better result is kept until no further improvements can be made. HC in general is a very greedy algorithm. A major benefit is that it only need a small amount of memory to keep track of the neighbors and it converges relatively quick. For neural net optimization, I adopted the MATLAB function *patternsearch* and for optimization problems, I adopted the java implementation from ABAGAIL toolbox [1]. The Matlab implementation (*pattersearch*) has two neighbor pooling method: 1) Complete polling which evaluates all the neighbors and takes the best one for the next step and 2) First Polling which uses the nearest neighbors with better fitness values. Both of the neighbor polling methods are examined.

2. Genetic Algorithm (GA)

GA is an optimization technique that models the principles of genetics and natural selection. GA allows mutation and crossover within population samples such that these samples can evolve to the optimal state. The advantage of GA includes 1) random exploration of the searching domain with crossover, 2) can easily handle large number of variables and 3) can jump out of local optima. For neural net optimization, I adopted the MATLAB function *ga* and for optimization problems, I adopted the java implementation from ABAGAIL toolbox.

3. Simulated Annealing (SA)

SA is an optimization technique that models the annealing process in metallurgy, in which random local search method that allows downhill moves at the beginning of the optimization process. The advantage is that SA allows enough exploration of the entire search domain at early stage so that the optimal solution is not sensitive to the starting point. This could lower the possibility of getting trapped in local optima. SA is sensitive to the initial temperature. For neural net optimization, I adopted a MATLAB implementation of simulated annealing [2] and for optimization problems, I adopted the java implementation from ABAGAIL toolbox.

4. Mutual-Information-Maximizing Input Clustering (MIMIC)

MIMIC differs from other optimization techniques in that it explicitly communicates lessons about the fitness function and mistakes such as local optima, based on previous iterations. Since MIMIC is only used for optimization problems, java implementation from ABAGAIL toolbox was used.

Optimization Problems

The optimization techniques will be applied to three optimization problems and their performance in terms of fitness score and run time will be evaluated.

1. Knapsack Problem (KP)

The knapsack problem is a constrained optimization problem: You are given a container with limited weight capacity and some items which each has a weight and a value; choose the number of each item to include in this container such that the weight limit is not exceeded but the total value of the items included is maximized (Figure 1). The problem represents a lot of real time problem in resource allocation where the resource is limited by financial constraints while the planning needs to achieve optimal efficacy. KP is a NP-hard problem and there is no known polynomial algorithm that describes the solution.

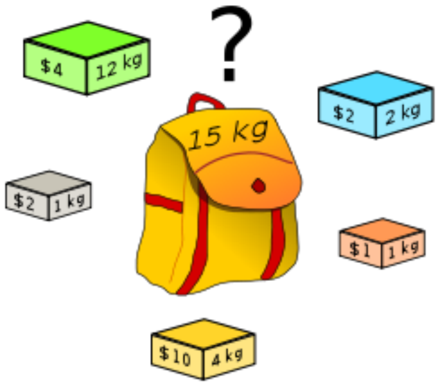


Figure 1 A Visualization Of Knapsack Problem [3]

For this problem, all algorithms were run 3 times each and the mean values are used to generate all graphs. Besides, KP was configured by changing the number of iterations for iteration = {50, 100, 500, 1000, 2000, 5000, 8000, 10000} and the number of items n for n = {40, 80, 160, 200}. The maximum volume and maximum weight of the items were 50. For each item, the weight and volume are randomized in each run.

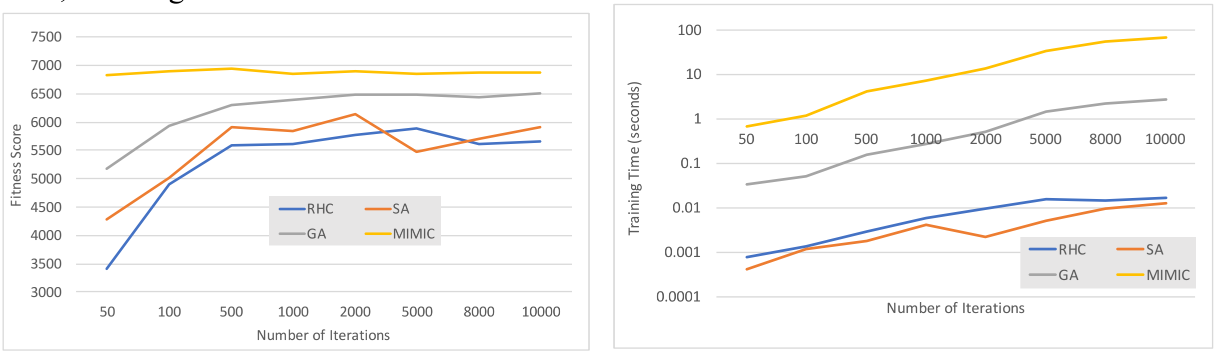


Figure 2 Performance of RO Algorithms on Knapsack Problem with Varied Iterations (n = 80).

As shown by Figure 2 and Figure 3, the best performing algorithm for KP is MIMIC, not only in regards to fitness score, but also in terms of number of iterations. Besides, as the number of items increases, the performance advantage in fitness score of MIMIC also increases (Figure 3). Due to the advantage of MIMIC in extracting as much information as possible from each iteration, MIMIC has very high efficiency even within few iterations (Figure 2). The fitness score from MIMIC is being quite stable as the number of iterations increases. The result generated by MIMIC in 50 iterations gets a fitness score higher than that from RHC, GA, and SA after training of 10000 iterations. This result indicates that MIMIC converge much faster the other three algorithms in terms of number of iterations. Thus, if cost evaluation function is very time consuming, MIMIC will be the best candidate since it takes much less iteration to converge to a solution.

In terms of training time, as shown by the right figure of Figure 2 and Figure 3, the number of iterations and training time has a expotential relationship for all algorithms. The training time is not so correlated with number of items in SA and RHC, but for MIMIC and GA, the training time increases exponentially as the number of items increases from 40 to 160. Among all the algorithms, MIMIC takes the longest training time.

Also, from Figure 2, we can see that GA converges in 1000 iterations while SA and RHC fluctuates and does not converge in 10000 iterations. The fluctuated behavior of RHC and SA, comparing to the consistent performance of GA and MIMIC, demonstrates the greedy nature of RHC and SA, which are more suitable for problems with small volumes of solution spaces. The reason that GA performs better than RHC and SA in this problem is the fact that instead of searching for optimal "points" in the sample domain, GA tries to capture the structure of the optimization landscape using the evolution process in several generations. The crossover operation represents distribution sampling, which allows GA to model the underlying structure of the problem from previous sampling.

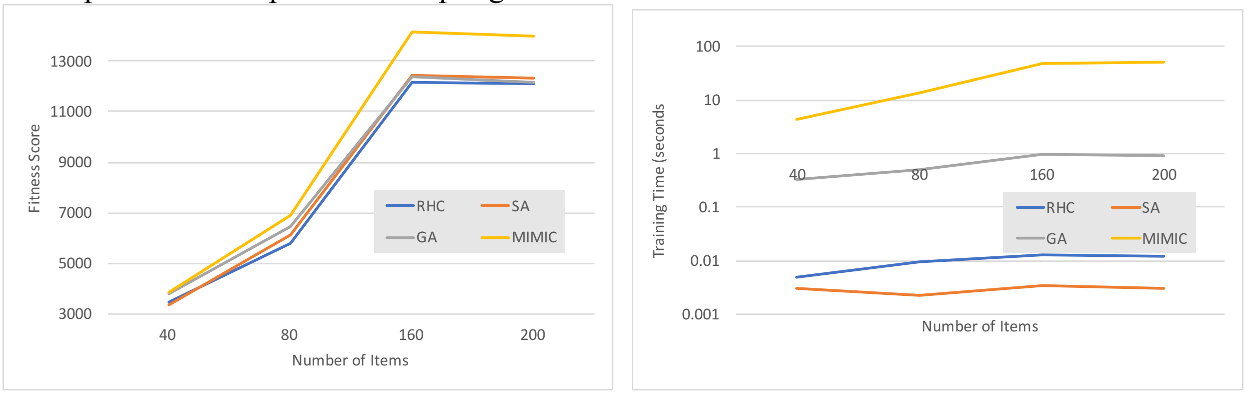


Figure 3 Performance of RO Algorithms on Knapsack Problem with Varied Items (iterations = 2000).

2. Traveling Salesman Problem (TSP)

TSP is also one of the most classic NP hard optimization problem. In short, a salesman who needs to travel a few cities (n), represented by scattered points in a coordinate plane. The problem is to determine the least costly round-trip route for this salesman in terms of travel time, ticket price and distance. In other word, given a list of vertices and edges, the goal of the problem is to figure out the shortest possible route that visit each vertex once and return to the starting point. TSP also represents a lot of real-life path planning problems such as what is the least time-consuming roundtrip route from home to a supermarket, a gas station, a library and go back, and the quickest route for assembling machine or manufacturing line.

This problem is considered NP hard since for N number of cities, there are (n-1)! Possible paths. For example, if we want to solve a route with 20 cities, there are 19! ≈ 1.22 $19!/2 \approx 6.08 \cdot 10^{16}$ possible routes to examine. Also, finding a universal function that address the general condition of this problem is impossible since small changes in location of the vertices will give completely different routes. Due to this nature of TSP, it is hard for greedy algorithms like SA and RHC to be able to accurately determine the best route in general.

For this problem, all algorithms were run 3 times each and the mean values are used to generate graphs. Besides, TSP was configured by changing the number of iterations for iteration = {50, 100, 500, 1000, 2000, 5000, 8000, 10000} and the number of cities n for n = {5, 20, 40, 60, 80, 100, 200}. The fitness score is evaluated by 1/distance traveled and thus higher fitness score means shorter travel distance.

Figure 4 shows the performance of RO algorithms in solving TSP problem with n = 80 and varied iterations in terms of fitness score and training time. Similar trend was observed with different number of cities n. As can be seen from the figure, SA, GA and MIMIC have very similar performance in regards to fitness score. GA outperformed all other RO algorithms in terms of fitness score. However, this advantage decreases as the number of iterations used to train increases. Besides, greedy algorithms (RHC and SA) requires least training time and have very similar performance. MIMIC, as reported before has the longest training time. The training time of all four RO algorithms increases exponentially as the number of iterations increases, which conforms with the fact that potential routes increases factorially as the number of cities increases.

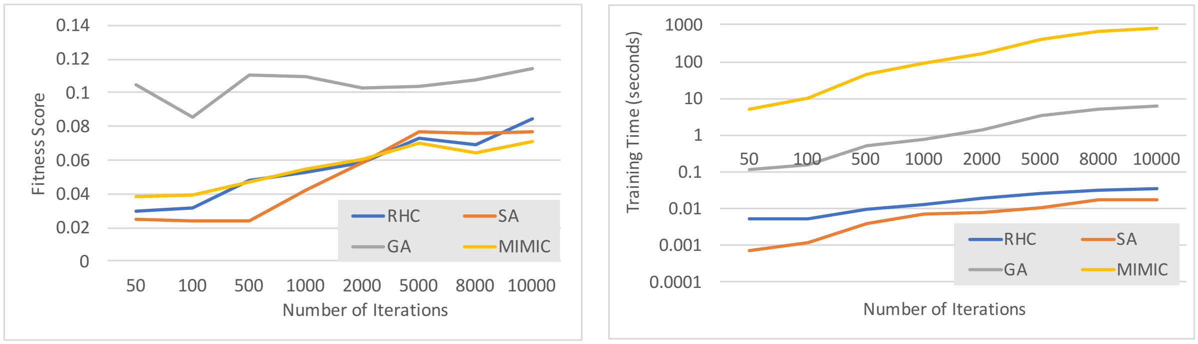


Figure 4 Performance of RO Algorithms on Traveling Salesman Problem with Varied Iterations (n = 80).

Figure 5 below shows the performance of RO algorithms in solving TSP problem with iterations = 2000 and varied number of cities. Similar trend was observed with different iterations. As can be seen from the graph, the advantage of GA over other RO algorithms in terms of fitness score increases as the number cities increases. On the other hand, the training time of GA has a linear relationship ($R^2 = 0.78277$) with number of cities. The training time of MIMIC, again, increases exponentially as the number of iterations increases ($R^2 = 0.97985$) while the training time of SA and RHC is relatively consistent over different number of cities.

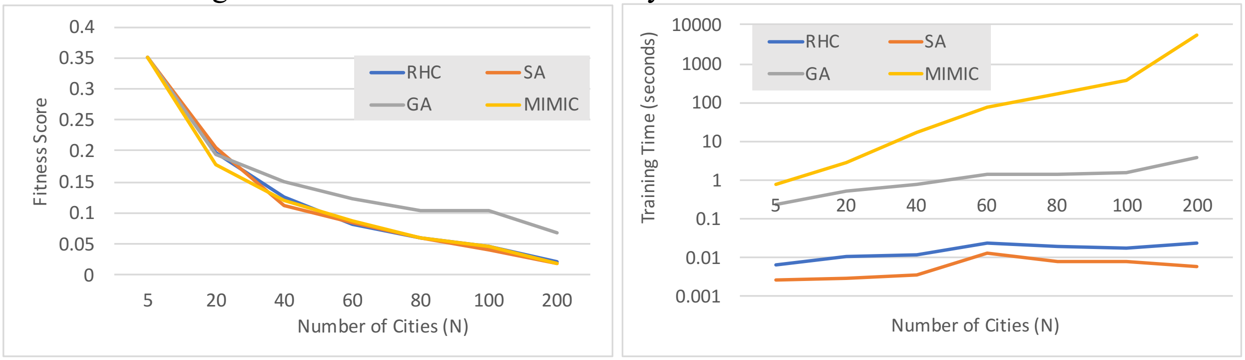


Figure 5 Performance of RO Algorithms on Traveling Salesman Problem with Varied Number of Cities (iterations = 2000).

The contradicting superiority of GA in regards to number of iterations and number of cities can be explained by the nature of these RO algorithms. First, unlike greedy algorithms which randomly search for the best solution, GA assumes the solution generated by two great solutions will be better than random sampling. Although not so focused in structuring the solution domain as MIMIC does, GA attempts to model the underlying relationship and relies on the inter-relationship of attributes to be somewhat representative of the solution space. The inter-relationship of attributes, however, needs to be addressed beforehand in GA's crossover function. The crossover function used in this problem is very well-founded and determines the success of GA in solving this problem: in a shortest path, any sub-path is also the shortest.

Secondly, the reason that the superiority of GA decreases as the number of iterations increases is that greedy algorithms like SA and RHC will eventually find some plausible solution since their induction bias is very low. As there is "no free lunch", they will need considerable iterations to find better result in a large solution space. Those greedy methods are more suitable for small solution space, in which they are more effective. This is also the reason why when N = 5 and N = 20, there is almost no difference between the fitness score returned by four RO algorithms with 2000 iterations. However, as the potential routes, which is the solution space, increases, 2000 iterations are not enough and RHC and SA tends to be impotent.

Moreover, the reason that MIMIC does not perform well in TSP is that any small change in the position and weight of the cities could result in completely different scenarios, which makes structuring the solution space more difficult.

2. Count Ones Problem (COP)

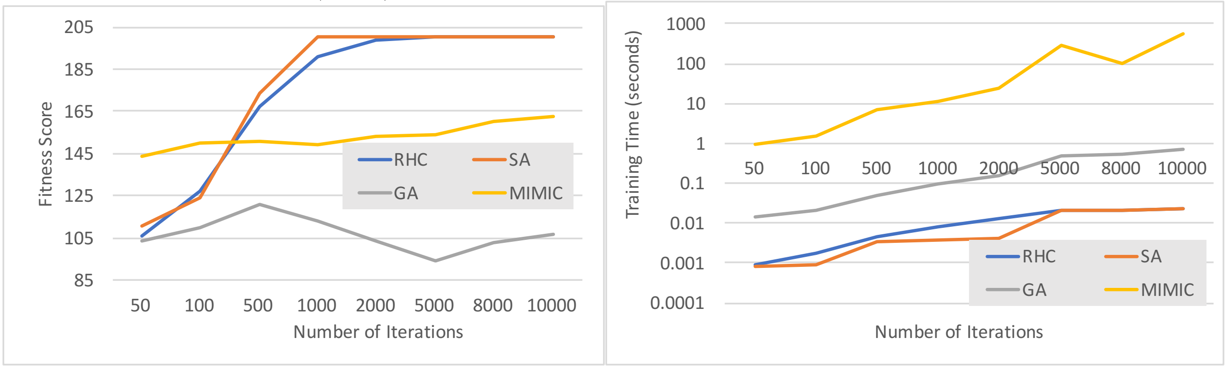


Figure 6 Performance of RO Algorithms on Count Ones Problem with Varied Iterations (n = 200).

Count Ones Problem (COP) was chosen because the solution is intuitive to human but not obvious for the algorithms. COP is also referred to as a "population count" problem. It is a discrete function defined for a bit vector with certain length (n). The goal is to find a "1" filled vector. In other words, the fitness function can be defined as,

$$\mathrm{C(X)} = \sum_{i=0}^{n} f(x_i) \text{ where } \mathrm{f(x)} = \{1\ if\ x = 1|0\ other\ wise\}$$

Maximizing the fitness score will give us the "1" filled solution. For this problem, all algorithms were run 3 times each and the mean values are used to generate graphs. Besides, TSP was configured by changing the number of iterations for iteration = {50, 100, 500, 1000, 2000, 5000, 8000, 10000} and the number of cities n for n = {40, 80, 120, 160, 200}.

Figure 6 above shows the performance of RO algorithms in solving COP with n = 200 and varied iterations in terms of fitness score and training time. Similar trend was observed with different number of n. n = 200 is chosen because the result range is larger and the trend is more obvious.

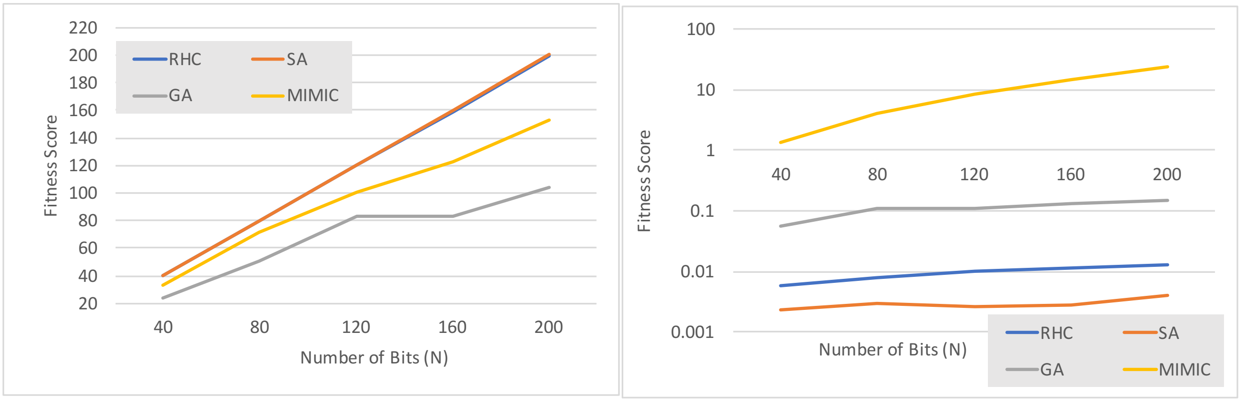


Figure 7 Performance of RO Algorithms on Count Ones Problem with Varied Bits. (iterations = 2000).

From Figure 6 and 7, we can see that SA and RHC both reach the optimal result, with the absolute advantage over GA and MIMIC. Among SA and RHC, SA reaches the optimal first and RHC follows. In terms of training time, the relationship between the four RO algorithms is: SA < RHC < GA < MIMIC. The training time of MIMIC follows the same pattern as demonstrated in the previous problem, which increase exponentially as the number of bits and number of iterations increases (right figures of Figure 6 and 7). The training time of all other algorithms stays relatively consistent over different number of bits and increases linearly as the number of iterations increases. This problem illustrates the strength of RHC and SA when the structure of the solution space is not detectable. Since GA cannot gain information from mutation and MIMIC could not detect the structure of the solution space from knowledge of history, greedy algorithms that has low induction bias outperforms GA and MIMIC. Besides, SA only needs around 1000 iterations to reach the optimal while RHC requires less than 5000 iterations when n = 200. The number of iterations required for RHC and SA to reach optimal solution is displayed in Table 1. In general, SA requires less number of iterations and training time to reach the optimal solution. Thus, SA should be the best optimization algorithm for Count Ones Problem.

	Number of Iterations	
Number of Bits	RHC	SA
40	1000	500
80	1000	1000
120	2000	1000
160	5000	2000
200	5000	1000

Table 1 Number of Iterations required for RHC and SA to reach optimal solution on Count Ones Problem

In short, COP, in contrast to TSP, demonstrates the scenario where greedy algorithms (SA and RHC) could outperform MIMIC and GA. When the problem to solve does not have an underlying structure, RHC and SA perform better; When the problem has some sort of structure in the solution space, GA and MIMIC 's attempt to capture this structure allow them to have better performance.

4. Parameter Tuning

To compare the four RO algorithms, I have also tuned systematic parameters, such as the cooling rate of SA, total population, number of crossover and mutation of GA and samples, samples to keep of MIMIC with fixed number of items (n) and fixed number of iterations. The fitness score and training time are evaluated with different combinations of parameter values.

This allows us to examine how each parameter affects the overall performance of the algorithms. The result is summarized in Table 2 below. For SA, the cooling rate is highly correlated with the fitness score of KP, TSP and the run time of KP and COP. This result complies with the fact that slower cooling rate means more function evaluation and thus more training time. Also, the probability for SA to find the optimal solution will be higher if the cooling rate is slower, which is shown by the equation below: $\Pr(\text{ending at global optima}) = \frac{e^{\frac{f(x)}{T}}}{Z_T}$

$$\textit{where } T \textit{ is the initial temperature and } Z_T \textit{ is the cooling rate}$$

SA	Cooling Rate	0.95	0.75	0.5	0.3	0.1	R^2
KP	fitness	5748.247318	5991.839592	5992.961457	6103.203628	6309.476718	0.8995
	time	0.002722905	0.005325981	0.005989967	0.006153125	0.006638374	0.7820
TSP	fitness	0.056884124	0.059470726	0.059824637	0.065963402	0.065851222	0.8804
	time	0.007699076	0.006467934	0.008867851	0.004956182	0.008539083	6.5536e-04
COP	fitness	80	80	80	80	80	NaN
	time	0.002735045	0.002796034	0.003789461	0.005206028	0.005691556	0.9376
GA	Population	80	80	80	80	80	
	to Mate	20	40	40	60	60	
	to Mutate	10	10	20	10	30	
KP	fitness	5823.646041	5949.644216	5819.494487	5822.537771	5610.436916	
	time	0.149684805	0.168299688	0.164627732	0.207005243	0.207957478	
TSP	fitness	0.102944825	0.112438105	0.107566253	0.104699373	0.093912086	
	time	0.264156742	0.528676268	0.532207912	0.626559535	0.667663081	
COP	fitness	70	73	66	72	70	
	time	0.149111686	0.191735878	0.189325754	0.21311433	0.225729075	
MIMIC	samples	80	80	80	80	80	
	to keep	10	30	40	50	60	
KP	fitness	6943.414959	6704.10139	6454.497139	6408.318518	6213.21099	
	time	5.84537742	6.741243021	7.536802019	7.993262026	8.527228469	
TSP	fitness	0.07513503	0.061812247	0.052963173	0.047281224	0.041014944	
	time	172.0783235	166.2040005	166.2340499	168.8011205	173.7477182	
COP	fitness	68	71	70	72	71	
	time	4.34410438	5.365456747	6.388135383	6.617111048	7.235074145	

Table 2 Effect of Systematic Parameters for SA, GA and MIMIC on three optimization problems with 2000 iteration and n = 80.

The mating (crossover) size of GA does not have strong impact on the fitness score given 2000 iterations, which is sufficient for most algorithms to reach an acceptable result. However, as crossover size increases, the training time also increases since it takes more time to "pair up" more samples. With same population and mutation size, increasing the mutation size has a negative impact on the fitness score of all three optimization problems. Mutations are used to prevent GA falling into local optima, but if increase the mutation to be too large, GA will essentially become random search. Thus, a smaller mutation size is preferable in those optimization problems.

For MIMIC, the retain size, which is the maximum number of samples MIMIC will retain after each iteration, is in general negatively associated with the fitness score. Thus, for each problem, there exist some optimal retain size that can generate the best fitness value. Also, higher retain size is associated with longer training time, which matches our expectation since less samples retained will obviously allow the algorithm to run faster.

5. Ranking
Through comparisons of four RO algorithms in regards to three optimization problems, we can summarize the most suitable algorithms for each problem in Table 3 below.

Problems	RHC	SA	GA	MIMIC
KP	3	3	2	1
TSP	2	2	1	4
COP	4	1	2	3

Table 2 Ranking of RO algorithms in regards to different optimization problems

Neural Network Problem

To compare RHC, SA and GA, I reused the neural network setup from assignment 1. Instead of using back propagation to find the weight of the neural network, RO algorithms will be used to find the optimized weight that has the sum of square error in this problem. The performance of these three optimization techniques in terms of training time and accuracy in training and testing dataset will be compared to the performance of back propagation scheme used previously. First, I will examine the relationship between accuracy and iterations and relationship between accuracy and run time because we are interested in the run time cost of each algorithm. Second, I will examine the relationship between training data size, accuracy and run time because we are interested in the computational complexity.

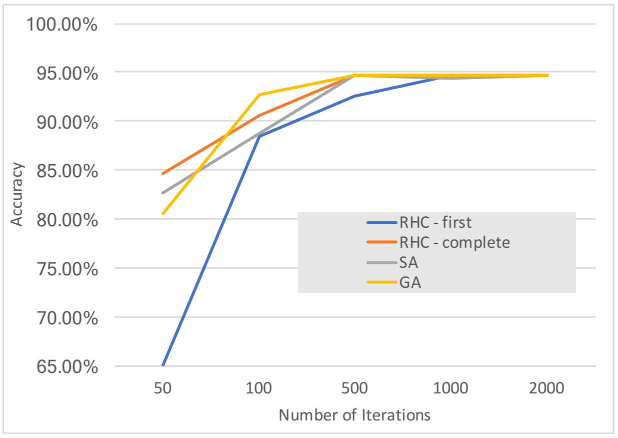
1. Dataset and Neural Network parameters

The wilt dataset which was used in the previous assignment of analysis of supervised learning was used again to evaluate RHC, SA and GA in terms of finding the optimized weight for Neural Network. This dataset is extracted from Quickbird imagery that detects diseased trees and since the data is derived from real-world images, it should contain certain level of noise. There are 4889 instances and 6 attributes in total. The task is to detect diseased and non-diseased trees from those multispectral image attributes. The dataset is split into 3387 training cases and 1452 testing cases (70% training and 30% testing).

Based on the experiment result of previous assignment, we used the best performing parameters, which are 10 neurons with 2 hidden layers, to configure the neural network in the following experiments.

2. Accuracy and Efficiency

First, number of fixed iterations of 50, 100, 500, 1000 and 2000 was used to train RHC, SA and GA. test, train accuracy and run time vs iteration are plotted (Figure 8 and Figure 9). In terms of iterations required to converge, RHC with first polling converge around 1000 iterations while RHC with complete polling, SA and GA all converge around 500 iterations (Figure 8). In terms of accuracy of training data and testing data, all optimization algorithms achieve very similar level (Figure 8). This indicates the optimization complexity of this dataset is very low and relevant to the classification problem so that all optimization techniques can achieve relatively good result. In terms of running time, RHC with complete polling has the longest training time and RHC with first polling has the second longest training time. This is expected since the first polling method skipped a lot of neighbors and thus reduce the number of function evaluations. The training time of RHC algorithms exhibits an exponential increase as the number of iterations increases while the training time of SA and GA increase linearly as the number of iterations increases (Figure 9). This is due to the greedy nature of hill climbing, which requires more iterations and many evaluation function calls. Unlike the optimization problems mentioned previously, the mean squared error evaluation of the neural net takes much longer time with a training dataset size of 3387 instances and 6 attributes. Thus, optimization with RHC takes the longest time in this problem.



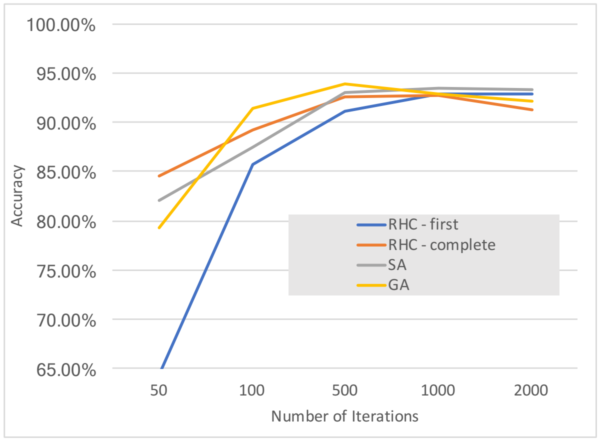


Figure 8 Training accuracy (Left) and Test accuracy (Right) vs Iterations

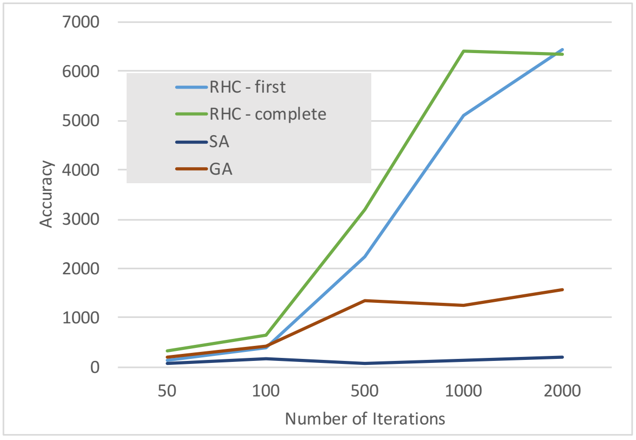
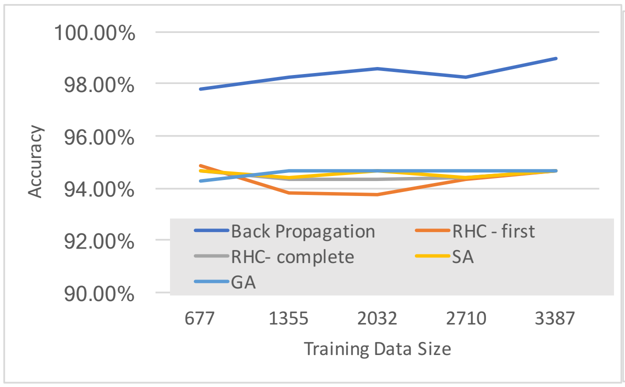


Figure 9 Running Time (Right) vs Iterations

3. Computational Complexity

I configured the neural network with different input training data size. The result from Back propagation, RCH, SA and GA are displayed below. First, none of the optimization algorithms achieve comparable results as the back propagation does, which is expected since back-propagation is known to perform well in classification problem in previous literature. Another reason could be that finding the optimal weight with the optimization techniques only involves minimizing the mean square error and could suffer a lot from overfitting since no cross-validation is performed.

Moreover, all optimization algorithms have very similar performance in both training and testing dataset. Besides, the performance of all algorithms is very consistent and there is no obvious relationship between the training data size and the training and testing data size. This also confirms with the result discussed above that the dataset is very well defined for this classification problem that a small dataset has the prediction power of a large dataset.



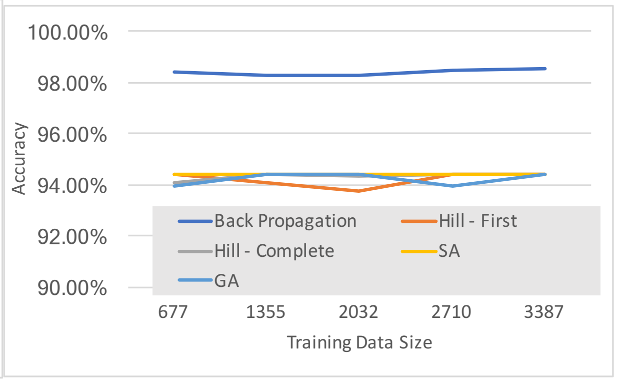


Figure 10 Training accuracy (Left) and Test accuracy (Right) vs Training Data Size

Conclusion

Through the result of experimentation demonstrated above, we can summarize the advantages and disadvantages of RHC, SA, GA and MIMIC in Table 3.

	Advantage	Disadvantage
RHC	- Simple, easy to understand - Low induction bias, suitable for discrete problem that no structure exists in the solution space (Count Ones Problem) - Fast when the solution space is small	- Greedy nature, the training time will increase exponentially as the number of items and iterations increases - Not desirable when problem has underlying structure -Requires large iterations in NP hard problems (curse of dimensionality)
SA	- Lower training time and training time growth rate - Suitable when good solutions are usually found to be close. - Heuristic, try to find the optimal solution	- Does not perform well in NP hard problems (Knapsack Problem and TSP) where solution space is very large (no free lunch) -Requires large iterations in NP hard problems (curse of dimensionality)
GA	- Lower training time and consistent over number of items - Suitable for problem with underlying structure (good solutions share similar characteristics) - High performance in NP hard problem	- Not suitable for discrete problem that no structure exists (Count Ones Problem)
MIMIC	- Suitable for problem that has time consuming cost evaluation function - Able to generate acceptable result within few iterations - Suitable for problem with underlying structure	- Training time intense - Memory intense - Training time increases exponentially as the number of items and iterations increases

Table 3 Advantage and Disadvantage of the four Optimization Algorithms

Reference

[1] https://github.com/pushkar/ABAGAIL).
[2] https://www.mathworks.com/matlabcentral/fileexchange/10548-general-simulated-annealing-algorithm)
[3] https://en.wikipedia.org/wiki/Knapsack_problem