CS7641 Fall 2018 Assignment2 Randomized Optimization

Xiaoxi Wang (xwang738)
email: xwang738@gatech.edu

Summary

This report explored the use of randomized optimization in solving different problems. Four local random search algorithms were examined, which included randomized hill climbing (RHC), simulated annealing (SA), genetics algorithms (GA) and MIMIC. In the first part, the first three algorithms were used to find the weights of a neural network trained over the Breast Cancer Wisconsin data set. It was found that all three algorithms achieved similar training and test accuracies, as compared to backpropagation, however, they required significantly higher cost of training time. In the second part, all four random search algorithms were applied to three optimization problems, which are the traveling salesman problem, the continuous peak problem and the knapsack problem, with each highlighting the advantages of GA, SA and MIMIC, respectively.

1. Part1

Finding the weights of an artificial neural network can be treated as an optimization problem. In assignment 1, the problem is solved by backpropagation, which minimized the cost function. Here three local random search algorithms were used to address this optimization problem. The Breast Cancer Wisconsin data set was used, which represents a problem of diagnosing whether a breast lump is benign or malignant [1,2]. This data set contains 683 instances, which were split into 70% training and 30% testing. There are 9 attributes and 2 classification categories, accordingly and to be consistent with Assignment 1, I used a neural network of 9 input units, 1 output units and 1 hidden layer with 20 units.

1.1 Randomized Hill Climbing

Randomized Hill Climbing is a straightforward algorithm aiming to find the global maxima with many randomized runs. The algorithm is problematic if there are many local optima, or the global optima is in a very narrow basin of attraction. Using RHC to optimize the artificial neural network over the breast cancer Wisconsin dataset, it was found that, as shown in Table 1.1 and Figure 1.1:

1) RHC ended up with an average training accuracy of 98.65% and testing accuracy of 95.93%.
2) All the 20 trials seemed to land in the global maxima with similar training and testing accuracies.
3) RHC converges with less than 2000 iterations (Figure 1.1).
4) The gap between training and testing accuracies seemed to enlarge with more iterations, indicating a possibility of overfitting.
5) RHC was fast with an average training time of 19.85 seconds for 5000 iterations.

	Training accuracy	Testing accuracy	Training Time
Average of 20 runs	98.651	95.927	19.854
standard deviation	0.356	0.913	1.055
Maximum among 20 runs	99.163	98.049	21.479
Minimum among 20 runs	97.908	94.634	17.746

Table 1.1. Performance of Randomized Hill Climbing for Neural Network Optimization

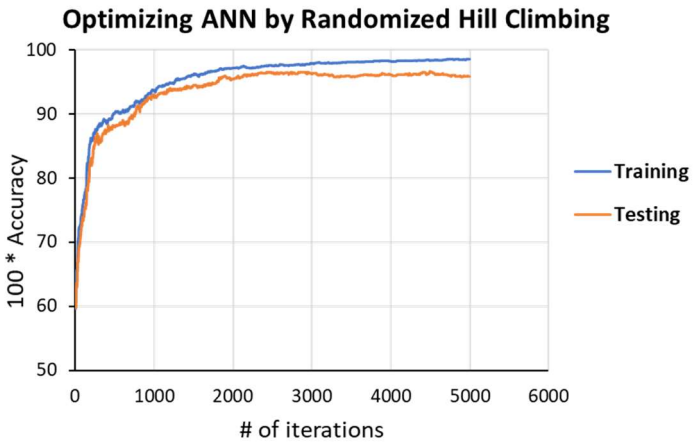

Figure 1.1 Performance of Randomized Hill Climbing over the number of iterations
Each data point on the curve represents an average of 5 runs.

1.2 Simulated Annealing

Simulated annealing is another method of approximating the global maxima[3]. It mimics the process when a metal is heated and then slowly cooled to achieve good arrangement of atoms. It starts with high temperature to "explore", and then "exploits" to land only in "good" locations. The starting temperature and cooling rate could be tuned to improve the performance of this method. For this dataset and optimization problem, it was found that:

1) The performance is not sensitive to changes in starting temperature (compare 1E9 and 1E10 in Table 1.2).
2) As cooling rate increases, it takes more iterations for SA to converge. As shown below in Figure 1.2.1, the curve plateaus at around iteration 1000 with 0.1 cooling rate, at around iteration 1500 for cooling rate 0.9 and > 5000 for cooling rate 0.99.
3) When the cooling process is very slow (cooling rate = 0.99), the gap between training and testing curve enlarges, indicating overfitting.
4) With no cooling, SA results in random landing, failing to find global maxima (Figure 1.2.2).
5) SA was fast with an average training time of < 19 seconds for 5000 iterations.

Parameters		Performance		
Starting temp	**Cooling rate**	**Training accuracy**	**Testing accuracy**	**Training time**
1.00E+09	0.1	98.256 ± 0.121	94.797 ± 1.490	17.434 ± 1.225
1.00E+09	0.9	99.302 ± 0.121	95.772 ± 1.126	16.974 ± 0.778
1.00E+09	0.99	97.699 ± 1.107	92.846 ± 0.282	17.234 ± 1.542
1.00E+01	0.1	98.606 ± 0.121	96.748 ± 1.126	17.490 ± 1.931
1.00E+01	0.9	98.675 ± 0.320	96.260 ± 1.228	18.039 ± 3.191
1.00E+01	0.99	98.257 ± 0.871	93.659 ± 0.976	17.619 ± 2.029

Table 1.2 Performance of Simulated Annealing for Neural Network Optimization

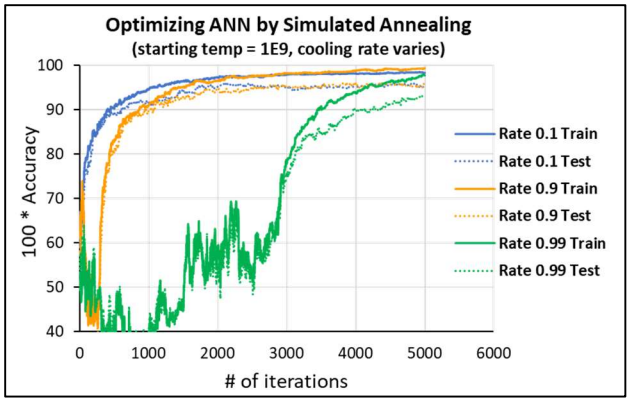

Figure 1.2.1 Cooling rates affects the performance of Simulated Annealing
Each data point on the curve represents an average of 5 runs

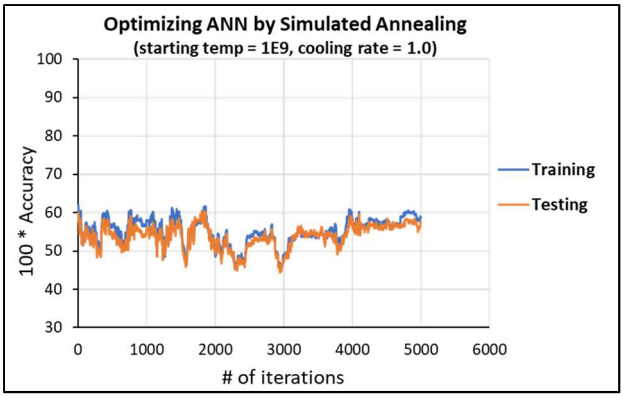

Figure 1.2.2 Simulated Annealing failed to find global maxima with cooling rate = 1
Each data point on the curve represents an average of 5 runs

1.3 Genetic Algorithm

Genetics algorithm is an analog of natural selection, the process that drives evolution [4]. With each generation, GA randomly selects individuals from the population with certain selection rules, apply mutations (changes) to these individuals, and then mate them to produce offspring of the next generation. Population size, mate and mutate rates can be tuned to improve the performance of this method. For this dataset and optimization problem, it was found that:

1) The performance is not sensitive to changes in mate rate (compare 10 and 30 in table 1.3).
2) Population size does not seem to affect GA performance significantly (Figure 1.3.2, and Table 1.3, compare 50, 100, 200 and 400). Moreover, larger population size takes longer training time.
3) Both training and testing accuracies increase with higher mutate rate (Figure 1.3.1 and Table 1.3, compare 5, 10, 30 and 50), at the expense of higher cost of computation time. However, with mate size being fixed (10), mutate rate 30 and 50 performs similarly. Therefore, it is preferable to use 30 as the mutate rate, considering the time cost.

Parameters			Performance		
Population	Mate	Mutate	Training accuracy	Testing accuracy	Training time
200	10	5	94.979 ± 0.209	92.846 ± 0.745	69.174 ± 0.090
200	10	10	97.001 ± 0.241	95.447 ± 0.282	99.41 ± 4.558
200	10	30	98.256 ± 0.121	96.098 ± 1.291	170.295 ± 4.592
200	10	50	98.675 ± 0.320	94.959 ± 2.031	237.018 ± 6.854
200	30	10	96.722 ± 0.639	95.447 ± 1.713	176.105 ± 7.148
200	30	30	98.257 ± 0.604	96.423 ± 0.563	238.891 ± 2.614
200	30	50	99.093 ± 0.121	95.935 ± 0.563	295.570 ± 5.668
50	30	30	97.838 ± 0.121	95.122 ± 0.488	184.480 ± 2.563
100	30	30	98.396 ± 0.527	96.423 ± 0.281	205.769 ± 8.933
400	30	30	98.047 ± 0.483	94.634 ± 0.845	237.499 ± 7.240

Table 1.3 Performance of Genetic Algorithm for Neural Network Optimization

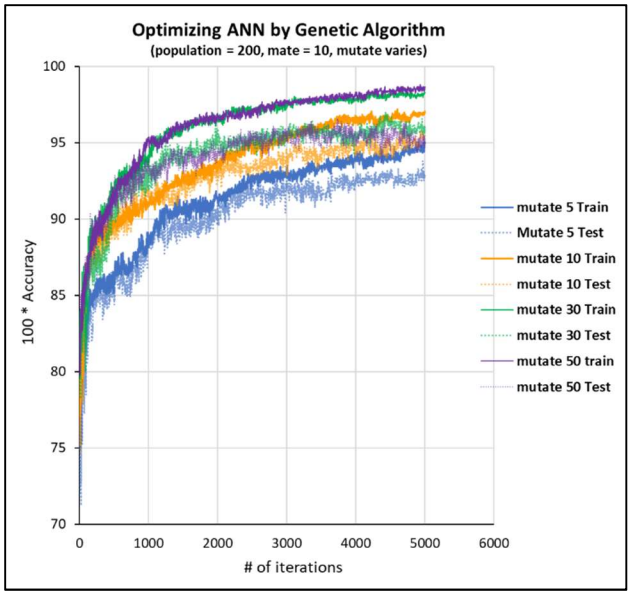

Figure 1.3.1 Mutate rate affects the performance of Genetic Algorithm
Each data point on the curve represents an average of 5 runs

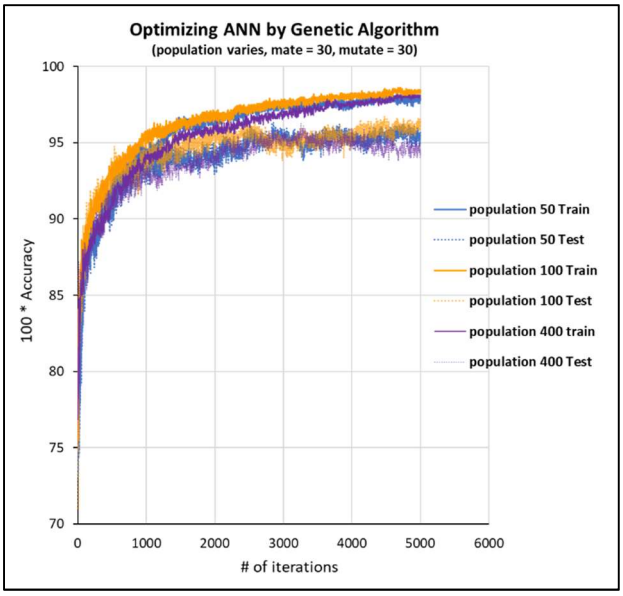

Figure 1.3.2 Population size does not affect the performance of Genetic Algorithm
Each data point on the curve represents an average of 5 runs

1.4 Comparison and discussion

With hyperparameters being tuned, RHC, SA and GA were compared to backpropagation in neural network weight optimization. The three random search algorithms seem to have similar performance in terms of training and testing accuracies (Figure 1.4).

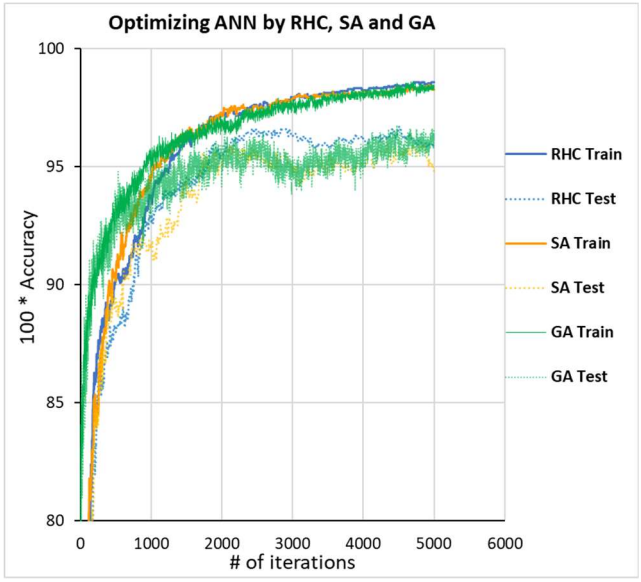

Figure 1.4 Comparison of RHC, SA and GA in neural network weights optimization
Each data point on the curve represents an average of 5 runs

As shown in Table 1.4, all three random search algorithms resulted in similar training and testing accuracies as compared to backpropagation, however, RHC takes ~30 folds longer training time, SA takes ~26 folds longer training time, and GA takes ~308 folds longer training time, as compared to backpropagation. In conclusion, backpropagation is preferable for neural network weights optimization.

Method	Training accuracy	Testing accuracy	Training time
Backpropagation	98.117 ± 0.419	96.098 ± 0.488	0.668 ± 0.072
RHC	98.651 ± 0.356	95.927 ± 0.913	19.8536 ± 1.055
SA	98.606 ± 0.121	96.748 ± 1.126	17.49 ± 1.931
GA	98.396 ± 0.527	96.423 ± 0.281	205.769 ± 8.933

Table 1.4 Comparison of RHC, SA, GA and backpropagation in neural network optimization

2. Part 2

2.1 Traveling salesman problem

The traveling salesman problem is a classic NP-hard problem that asks the question "given a list of nodes and distances between them, find the shortest possible route that visits each node only once and returns to the original". It has many applications in routing and scheduling.

For this report, we set number of nodes N = 50. As shown in Figure 2.1, the genetic algorithm (GA) outperforms the other three by several folds, despite longer training time as compared to SA and RHC. It is also impressive that it takes about 100 iterations for GA to converge, although for more complicated problems (bigger N values), it may take more iterations to converge.

The traveling salesman problem is a combinatorial optimization problem, and the hypothesis space is very large. RHC and SA only retain one solution at a time, whereas GA works with a large set of possible solutions, and "evolve" the population with crossover and mutation, which allows it to explore the hypothesis space much faster. Therefore, GA is the preferable algorithm to solve the traveling salesman problem.

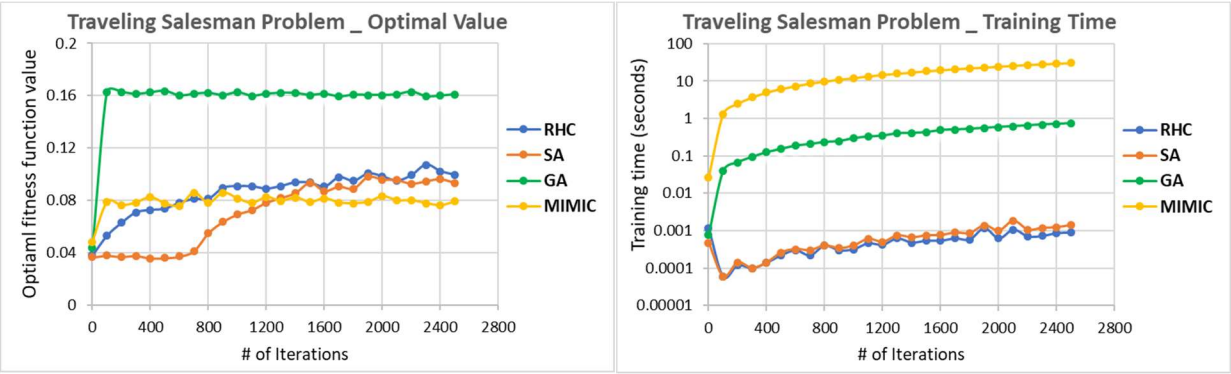

Figure 2.1 Traveling salesman problem fitness function optimization with RHC, SA, GA and MIMIC.
Left: optimal fitness function values over iterations; right: training time over iterations.
Each data point on the curve represent the average of 5 runs**.**

2.2 Continuous Peak Problem

Four peaks problem is defined as given an input vector of N binary elements, maximizing at binary string that have contiguous 0s or 1s up to the trigger point (T), and then the complementary bit contiguously until the end of the string [5]. The continuous peak problem is a variation of four peaks.

For this report, I set number of nodes N = 60, T = 6. As shown in Figure 2.2, it is found that both SA and MIMIC reached global maxima after a certain number of iterations. MIMIC was able to capture the underlying structure and combine information from all maxima [5]. The annealing process in SA allowed it to "explore" and "exploit", and therefore avoid being trapped in local optima. In contrast, GA and RHC were trapped in local optima and failed to reach the best solution. Finally, although MIMIC required less iterations to converge (Figure 2.2, left), SA takes significantly less computational time (Figure 2.2, right). Therefore, SA is the preferable method for solving the continuous peak problem.

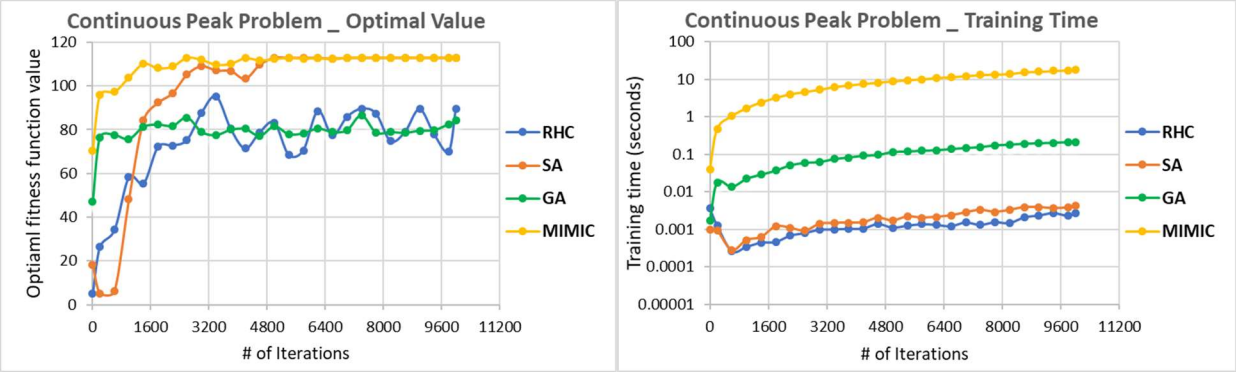

Figure 2.2 Continuous peak problem fitness function optimization with RHC, SA, GA and MIMIC.
Left: optimal fitness function values over iterations; **right:** training time over iterations.
Each data point on the curve represent the average of 5 runs.

2.3 Knapsack Problem

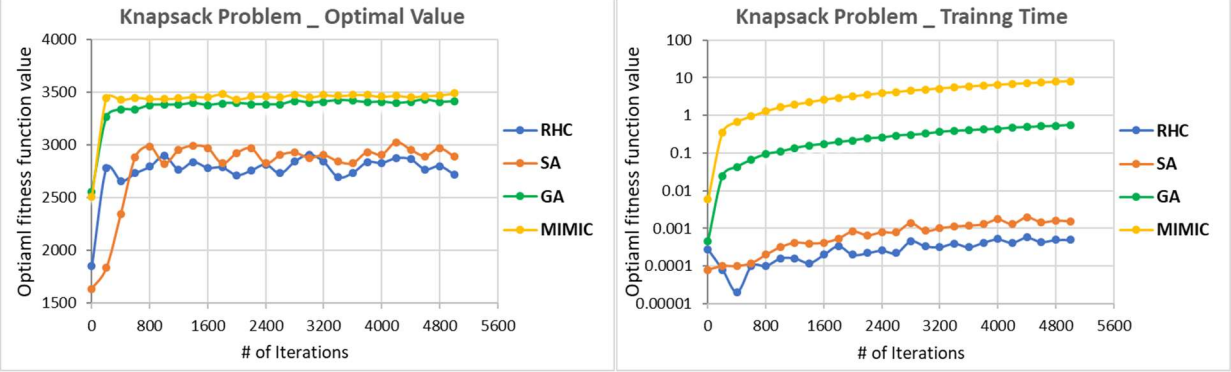

Figure 2.3 Knapsack problem fitness function optimization with RHC, SA, GA and MIMIC.
Left: optimal fitness function values over iterations; **right:** training time over iterations.
Each data point on the curve represent the average of 5 runs.

The Knapsack problem is another combinatorial optimization problem, which asks the question “given a set of items each having a weight and a value, determine the number of each item to include in a collection to maximize the total value of the collection, with total weight less than or equal to a given limit ”. The Knapsack problem is also a NP-hard, nondeterministic problem.

For this report, the problem was defined as having 4 copies of 40 items, with maximal weight and volume of each item being set as 50. As shown in Figure 2.3, it is found that both GA and MIMIC reached global maxima after less than 400 iterations. In contrast, GA and RHC were trapped in local maxima and failed to reach the best solution even after 5000 iterations. Finally, MIMIC outperforms GA in terms of optimal value, and it converges with less iterations, although it requires longer computational time. Therefore, MIMIC is the preferable method for solving the knapsack problem. This is attributed to the strength of MIMIC in discovering the underlying structure of the optimization landscape.

2.4 summary

By applying RHC, SA, GA and MIMIC to different optimization problems and observe their behaviors under different circumstances, it was found that both RHC and SA were efficient in terms of computational time, and they were suited for solving well-defined, relatively simple problems. SA is improved compared to RHC, as it is more resistant to being trapped in local maxima, which was attributed to the “explore” and “exploit” process enabled by annealing. GA is more computational expensive, although it can solve complicated optimization problems with large search space and many local optima. MIMIC requires a significant smaller number of iterations to reach optimal solution, although it is generally the most computational costly. However, MIMIC reveals the underlying structure of optima, and it is a powerful method of solving different complicated optimization problems.

Reference

1. W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.
2. O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. Operations Research, 43(4), pages 570-577, July-August 1995.
3. Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. "Optimization by Simulated Annealing." Science 220, 671-680, 1983.
4. D. E. Golberg, Genetic algorithms in search, optimization and machine learning, Addison-Wesley Publishing Company, 1989.
5. Bonet, J.S.D, Isbell, C.L., Viola, P. “MIMIC: Finding Optima by Estimating Probability Densities”. Advances in Neural Information Processing Systems 1996, 424