RANDOMIZED OPTIMIZATION

Dixon Domfeh
GTID: 903658462

1.0 Introduction

This report is on the implementation and performance evaluation of four randomized optimization algorithms – Randomized Hill Climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA), and Mutual Information Maximizing Input Clustering (MIMIC) algorithm. We assess the performance of these four algorithms on four optimization problems. The first three problems are on discrete-valued parameter spaces carefully chosen to highlight the strength and weaknesses of each algorithm. We specifically choose the Knapsack, N Queens, and Four Peaks problems. The fourth problem involves using the RHC, SA and GA algorithms to find the optimal weights for a neural network. We use the Churn data set analyzed in Homework 1 for this purpose. The churn dataset is particularly interesting since it has a minority class. It will be interesting to see the performance of an algorithm such as GA which does mutations on the strongest genes (i.e., most prominent labels).

2.0 Optimization Problems

The Knapsack problem involves optimizing an objective function subject to some constraints. It is an NP-hard problem with no polynomial time solution. Consider a bag of items, each has weight and values. The goal is to determine the maximum value of items that can be collected into a knapsack such that the total weight is less or equal to a given weight limit. Let $x_i, i = 1, \dots, n$ be the number of copies of n items, $z_i, i = 1, \dots, n$. The value v_i is non-negative. Let constraint W, be the weight one is allowed to carry and c_i is the total copies of each item x_i. The maximization problem can then be represented as follows

$$\arg\max_{v} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to } \sum_{i=1}^{n} w_i x_i \leq W \; and \; x_i \in \{0,1, \dots, c_i\}$$

The Knapsack problem is chosen in this context to highlight the strength of MIMIC as it can in theory exploit the underlying structure of the problem space that has learned from previous iterations.

The Four Peaks problem has two local optima with wide basins of attractions and two sharp global optima. The Four Peaks problem is chosen to highlight the weakness of the local optimization algorithms – Random Hill Climbing and Simulated Annealing. Genetic algorithms are more likely to find these global optima than other methods since these are global algorithms. The third discrete problem is the N-Queens puzzle. It involves placing N chess queens on an $N \times N$ chess board such that no two queens are a threat to each other. The solution is straight-forward logically, place two queens in way that they do not share the same row, column, or diagonal.

However, the computational complexity increases at an asymptotic growth rate in the number of solutions when N is high. In this report we choose $N = 8$ to represent an 8-queen problem. There are 4,426,165,368 possible arrangements, but only 92 distinct solutions when $N = 8$. The task for our four algorithms is to find the maximum number of solutions.

The final problem is a discrete classification problem using a neural network. The goal is to find optimal weights of the neural network such that the network outputs the best prediction. We replace the backpropagation algorithm with RHC, SA and GA and compare their performance across several metrics. The data set is a customer churn data set obtained from Kaggle. The churn dataset contains information on 7043 customers of a telecommunication company. The target label – Churn – is a binary response of about 73% YES and 27% NO responses. The dataset is interesting due to the highly imbalanced nature of the response labels. Prediction of the minority class (i.e., NO responses) will be challenging since the network has relatively less data to learn from. This problem set is chosen to highlight the shortcoming of the genetic algorithm. The GA uses mutation and crossovers to generate new candidate solutions from a family of fittest parents. With more information on the YES labels, it would be expected that the GA algorithm would find "strongest" children from this class.

3.0 Analysis of Results

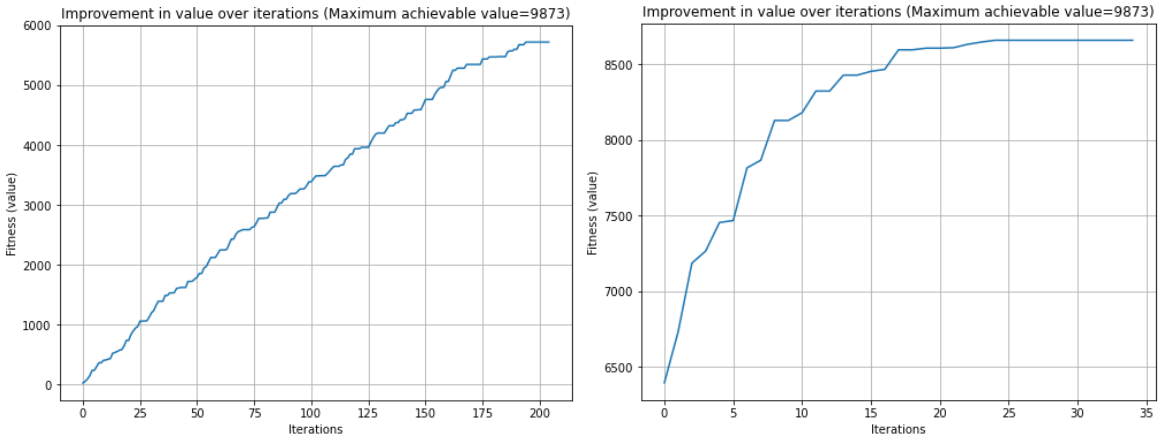
3.1 Hyperparameter Tuning

The hyperparameters for SA, GA, MIMIC and RHC have been chosen through an exhaustive grid search. For instance, the hyperparameters of the number of attempts the algorithms can take to find a better neighbor at each step was varied from 10, 100, 1000 across all four algorithms. The cooling schedule is chosen to be and exponential decay rate varying 0.7, 0.8, 0.95. For MIMIC we tried several sample sizes for each iteration – 10, 40, 60, 80, 100 while the number of samples to be kept at each iteration was varied – 20%, 50%, 80% and 90%. For the genetic algorithm we tried several mutation probability and population sizes.

3.2 Results

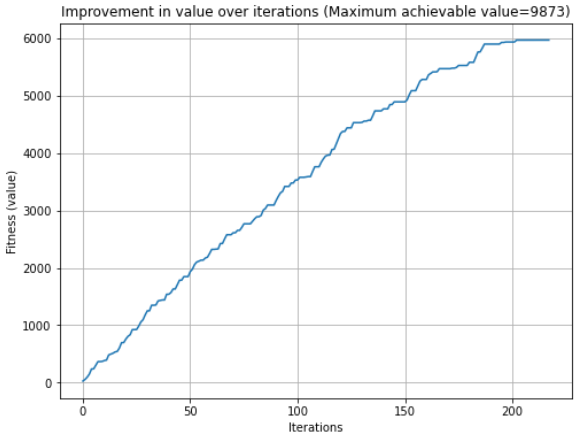
3.2.1 Knapsack Problem

The maximum achievable value in our knapsack problem setting is 9873. As such one way of evaluation all four algorithms is their average maximum value. We also assess the algorithms' run time and number of calls they make to the objective function until convergence. We try two variants of the RHC algorithm – RHC with(out) restarts. Figure 1 displays the convergence plots of the algorithms.

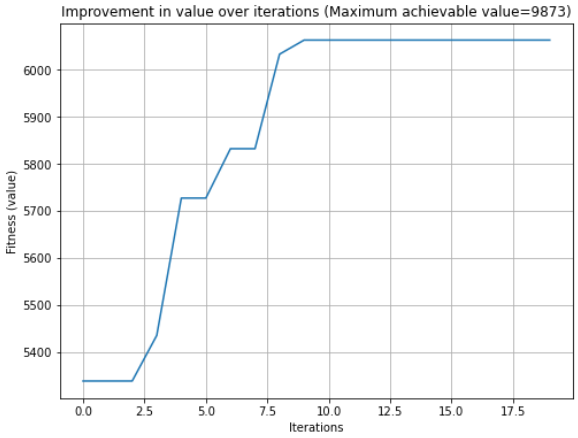


(a) RHC without restart

(b) MIMIC



(c) SA



(d) GA

Figure 1: Convergence plots of all four algorithms on the knapsack problem

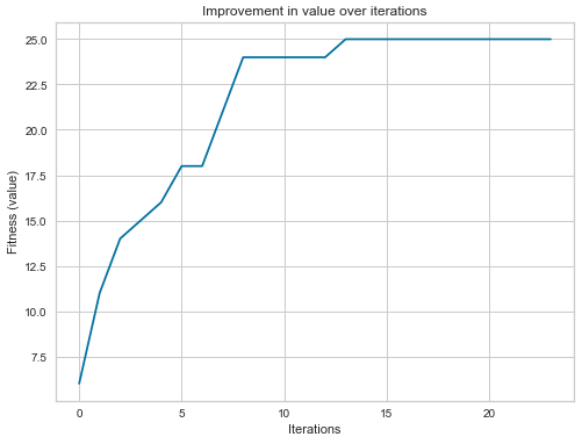
Table 1 below gives a summary of the algorithms' performance on the knapsack problem. From Table 1 we can deduce that the RHC and SA have relatively similar performance in terms of average best value, number of iterations and optimization time. This is expected since the only difference both algorithms is the acceptance probability. Due to their local search preference bias, RHC and SA can be stuck in local optima. Nevertheless, multiple restarts of the RHC may allow the algorithm to locate a global optimum. When we use an instance of RHC with random restarts we realized a relative increase in the average best value (see Table 1 below). The RHC with random restarts produced at better optima with an additional cost in terms of number of iterations and computation time. The MIMIC algorithm has the highest best value as expected. It achieves this because it's able to find patterns in the underlying distribution. Hence, MIMIC may find the global maxima. In our knapsack problem, the maximum value achievable is 9,873. MIMIC achieved 8,730 with a deviation of only about 11.5% from the target. However, MIMIC achieves this score at a higher cost reflected in its computation time (5.37 minutes). Even though the GA doesn't produce maximum value close to MIMIC, it may be considered the next best algorithm in this knapsack problem in terms of less computation time and number of iterations.

Algorithm	Optimization Time	Average Best Value[1]	Number of Iterations
RHC	0.017sec	5870 (3.7%)	205
RHC (with restart)[2]	0.182sec	6382 (1.4%)	2109
SA	0.017sec	5904 (3.5%)	208
GA	0.071sec	6063 (1.04%)	20
MIMIC	5.37min	8730 (0.56%)	35

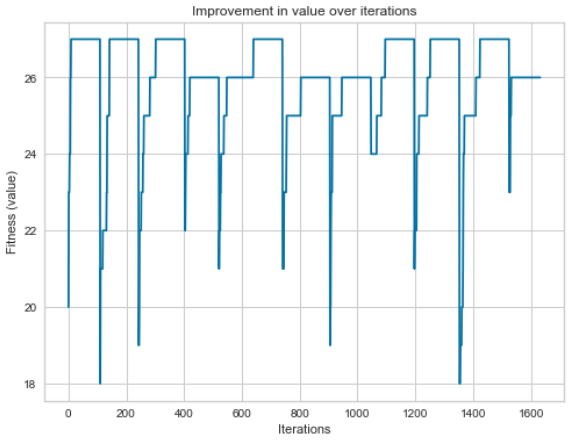
Table 1: Performance summary of algorithms used on the knapsack problem

3.2.2 N-Queens Problem

The eight-queen problem has 92 distinct solutions. We assess all five algorithms on the maximum number of distinct solutions they can find. It appears that all five algorithms are well short of the target distinct solutions (see Figure 2 below). From Table 2, we report that the average best value across all algorithms are relatives similar. In this problem, RHC with random restart can be considered the best algorithm since it finds about 27 distinct solutions with relatively short computation time. Although MIMIC finds relatively similar number of distinct solutions, it does so with a higher duration.

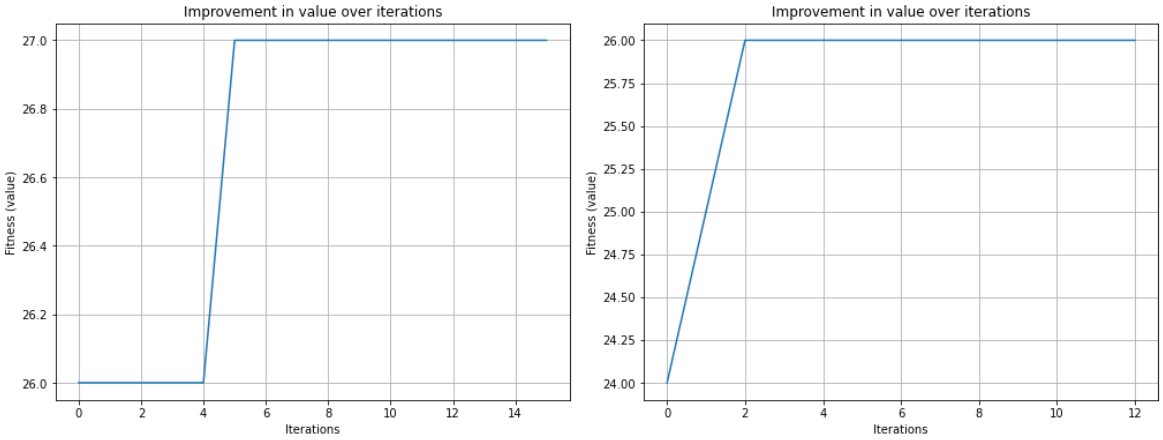


(a) RHC without restart



(b) RHC with random restart

[1] Average of 100 runs of each algorithm, standard deviation in parenthesis
[2] Number of random restarts = 10



(c) MIMIC (d) GA

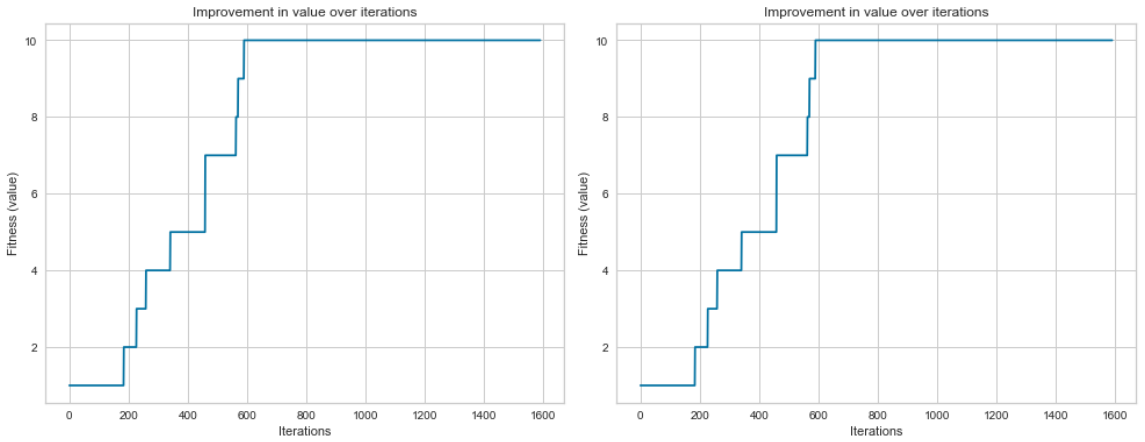
Figure 2: Convergence plots of all four algorithms on the N-Queen problem

Algorithm	Optimization Time	Average Best Value[3]	Number of Iterations
RHC	0.001sec	24.81 (1.2%)	24
RHC (with restart)	0.014sec	26.51 (0.6%)	1633
SA	0.009sec	25.74 (0.01%)	1633
GA	0.330sec	25.36 (0.07%)	13
MIMIC	0.276sec	27.06 (0.6%)	16

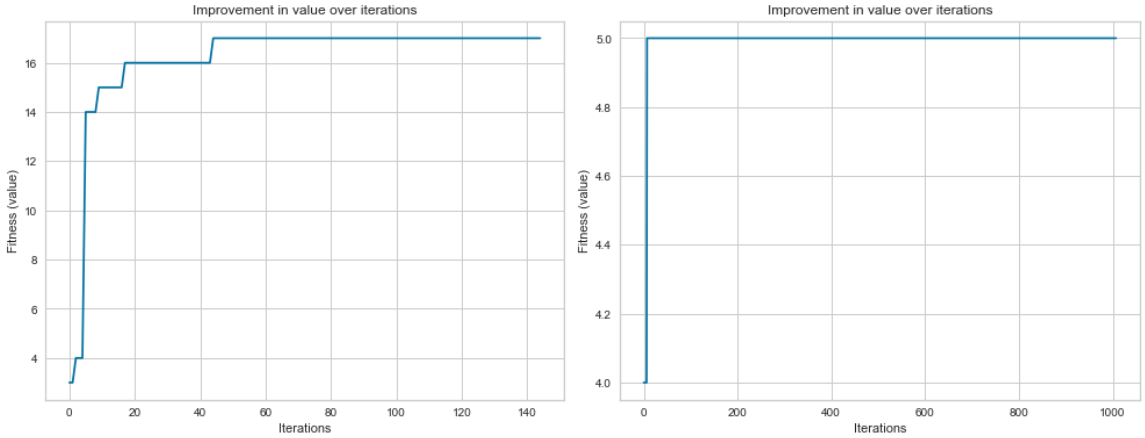
Table 2: Performance summary of algorithms used on the N-Queen problem

3.2.3 Four Peaks Problem

The goal of this problem is to evaluate how each algorithm deals with the temptation of falling into the wide basins of attraction between two sharp global optima. We had to give the RHC algorithm had to give 1,000 attempts to find a best value of 10. A similar case is with the random restart variant of the RHC and SA algorithms. This observation can be attributed to the fact that local search algorithms such as RHC and SA will need more attempts and iterations to escape the wide basins of attraction and hopefully find the global optima. Surprisingly, MIMIC reports the lowest score even after many attempts and increase in population size. Our intuition at this point is that, unlike the knapsack problem, the four-peak problem doesn't have any underlying structure that MIMIC can exploit to its advantage. Clearly, the GA has the best score on the four-peak problem. Figure 3 and Table 3 report the convergence plots and performance summary respectively.



(a) RHC without restart (b) SA



(c) GA (d) MIMIC

Figure 3: Convergence plots of all four algorithms on the Four-Peak problem

Algorithm	Optimization Time	Average Best Value	Number of Iterations
RHC	0.04sec	10.00 (0.0%)	1591
RHC (with restart)	0.43sec	10.98 (0.02%)	19709
SA	0.56sec	10.77 (0.02%)	1591
GA	2.68sec	16.96 (0.2%)	145
MIMIC	68.32sec	2.9 (0.9%)	1008

Table 3: Performance summary of algorithms used on the Four-Peak problem

3.2.4 Neural Network

In finding the optimal weights for a neural network, the goal is to minimize the objective function. Therefore, the problem for our algorithms here is to minimize the sums of square errors to enable better prediction. We assess the RHC, GA and SA and compare their performance to the gradient descent which is commonly used in neural networks. In the problem examples previously tackled, the input space is unit dimensional. However, in this case of the churn dataset, the input dimensions are 29. This increases the search space with many local and global optima. As such local search algorithms such as RHC and SA may suffer in performance relative to GA. Since neural networks tend to overfit with increase in number of layers, we keep a 4 hidden layers architecture for all algorithms. We use the gradient descent here only as a strong baseline to compare our stochastic algorithms to. By its design, gradient descent algorithm is directional and travels the error surface in the direction where error is reducing. Our randomized algorithms are the opposite.

Table 4 presents a summary of the algorithms' performance. Notably, the f1 score on the minority class label (i.e., NO class) is lower relative the YES class across all four algorithms. However, the NO class is practically zero for the Genetic Algorithm. How is this the case? We reason that the crossover assumption in GA where a population hold more information than local mutation may be the cause here. In GA two parents are repeatedly chosen at random to reproduce in the genetic sense to give birth to a new offspring. This is in the hope that the children will inherit the positive attributes of both parents. In our case since the distribution of the YES class is higher in the dataset, the probability of picking those attributes is relatively higher. Hence, any two parents chosen at random are more likely to be from the YES class than the NO class. Consequently, the offspring inherits the most dominant gene or attribute (i.e., YES class labels). This is corroborated by the high f1 score for the YES class (85%) in Table 4. The GA learns more about the majority class than the minority.

Metric	Gradient Descent	RHC	GA	SA
Yes (F1-score)[4]	0.86	0.61	0.85	0.61
No (F1-score)[5]	0.57	0.47	0.00	0.47
Macro Average (F1-score)	0.72	0.54	0.42	0.54
Wall time (training)	5.62s	3.8s	1min 29s	5.27s
Wall time (query)	10.2ms	9.32ms	9.38ms	7.88ms

Table 4: Performance comparison of algorithms

Further evidence can be observed from the loss curves on the test data set presented in Figure 4 below. Particularly, we see that the GA converges relatively quickly having found the best weights with the lowest error relatively to RHC, SA and even gradient descent. Upon closer inspection however, we see that the GA has the lowest macro average f1 score even though it achieves the lowest error in Table 4. Hence the GA does not generalize well in this case.

[4] Support = 1033
[5] Support = 374

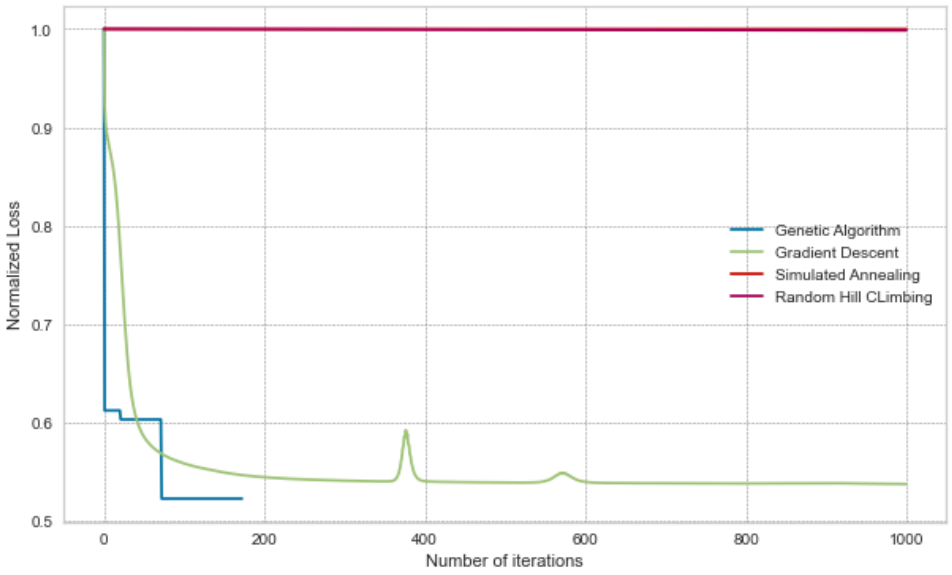
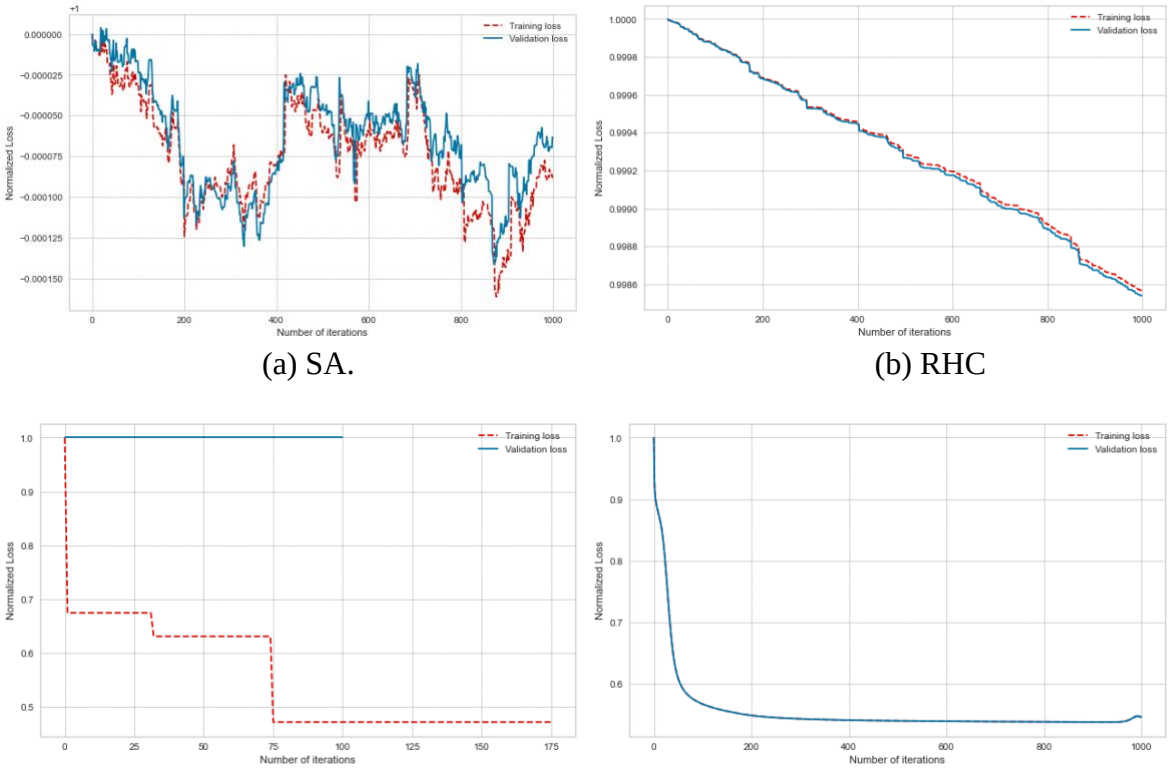


Figure 4: Normalized loss curves for algorithms on the test set

We also tested the performance of the algorithms on a validation set and the normalized loss curves are presented in Figure 5 below. We would like to make a note on the bumpy error surface of the simulated annealing algorithm. The temperature parameter allows the algorithm to explore sometimes (wonder around a little bit). Therefore, the loss curve looks wavy. Overall, the loss curve still stays within a range like the RHC.



(a) SA. (b) RHC

(c) GA (b) Gradient Descent

Figure 5: Normalized loss curves for algorithms on the validation set

In terms of overall performance, we rank the RHC as the best algorithm to be used in this context. Even though RHC and SA have the same performance in terms of f1 scores, we choose RHC because of its relatively shorter training time.

4.0 Conclusion

Thus far, this report has compared and contracted the performance of four stochastic optimization algorithm in several different contexts. We particularly highlight the strengths and weaknesses using carefully chosen examples. For general discrete optimization problems where there are clear patterns could be exploited, the MIMIC algorithm would be the best choice. However, the MIMIC algorithm cannot be used on a continuous-valued parameter space. That is the reason why it was not included in the neutral network weights problem. Genetic algorithms work generally well on both discrete- and continuous-valued parameter space. GA are fast and often performs well in finding a global optimum. The local search algorithms – Randomized Hill Climbing and Simulated Annealing – are ideal for unimodal optimization problems.