# CS7641 Assignment II: Prediction via Randomized Optimization

*T. Ruzmetov*

*October 10, 2017*

## Introduction

Purpose of this work is to understand how various randomized optimization techniques like "Hill Climbing", "Simulated Annealing", "Genetic Algorithm" and "MIMIC" perform on different problem sets such as "Continuous Peaks Problem", "Traveling Salesman" and "Knapsack" problem. In addition, we are asked to test performance of ANN with randomized optimization methods such as "Hill Climbing", "Simulated Annealing" and "Genetic Algorithm" against ANN with backpropogation method we used on one of the datasets from assignment one.

### Randomized Hill Climbing (RHC)

RHC method is designed to find optimal(usually maximum of fitness function) by randomly choosing starting position and moving until turning point is reached. By performing many iterations RHC chooses best optima among all local optimas found. RHC method is fast, but it is not robust in its search for global optima in situations where there are many local peaks. I performed RHC using RandomizedHillClimbing in ABAGAIL via Jython.

### Simulated Annealing (SA)

Simulated Annealing is very widespread optimization technique especially in physics, where it is mostly used for energy minimization problems. To find global optima, it randomly samples different states by accepting them based on detailed balance condition, while slowly cooling the system. The algorithm, a function of initial temperature and cooling rate, strikes a balance between exploring new points and exploiting nearby neighbors in search of local optima. I used SimulatedAnnealing in ABAGAIL via Jython.

### Genetic Algorithm (GA)

Inspired by biology, in Genetic Algorithm the population evolves by iteratively mating and mutating parts to crossover the best traits and to eliminate irrelevant traits. Large hypothesis space is a major disadvantage of GA, which is dictated exponentially by the number of attributes. SA was performed using StandardGeneticAlgorithm in ABAGAIL on Java+Jython.

### Mutual-Information-Maximizing Input Clustering (MIMIC)

In contrast to most optimization algorithms MIMIC algorithm is unique in its ability to "remember" previous iterations and use sampled probability densities to build structure of the solution space in order to locate global optima. MIMIC was performed using MIMIC built into ABAGAIL with Java via Jython.

## Part 1: Three Famous Optimization Problems

### Continuous Peaks

In Continuous Peaks problem we are given a fitness function in 1D space with multiple optimas, where the main task is to find global optima. Given its simplicity it is very good candidate for learning purpose. I tested the performance of above mentioned 4 randomized optimization methods on this problem by making some modifications into provided Jython code under ABAGAIL package.

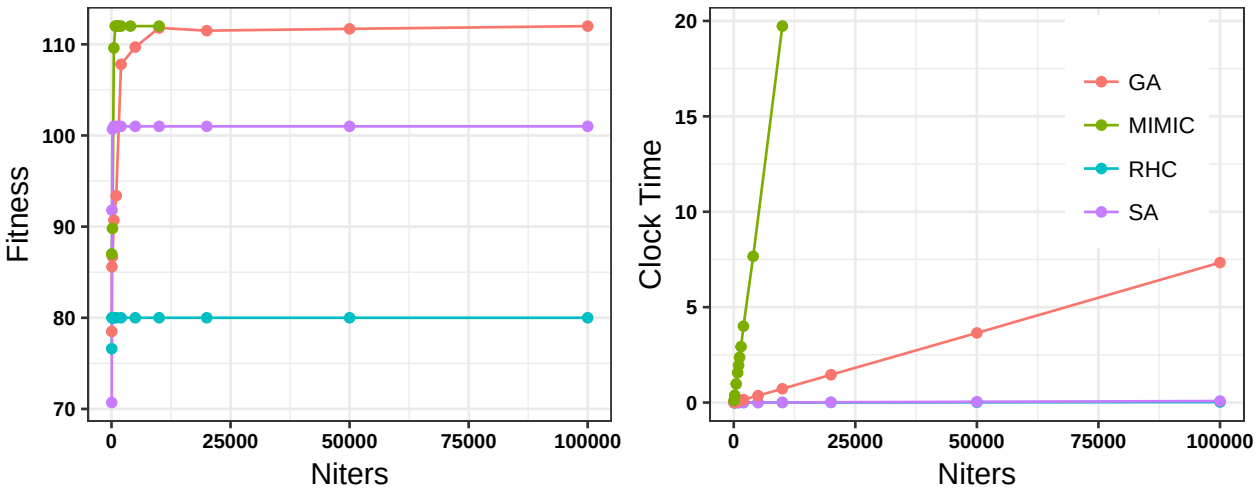


Figure 1: Continuous Peaks Performance

I made comparison between models by plotting fitness function versus number of randomized iterations as depicted in Fig.1. As shown, GA and MIMIC models gave pest performance by achieving optimum value at about Fitness=112. One has to note that due to its extreme time consumption MIMIC model was iterated only 1000 times and yet achieved best performance. For a large number of iterations Genetic Algorithm resulted in best performance. It reached near optimum value after 10000 iterations which is a late when compared to other methods. The worst performance is shown by RHC, where it could only reach Fitness=80 which is probably local optima near global one. Overall, SA and RHC algorithms are the fastest in performance. MIMIC was the slowest and GA was somewhere in the middle.

**Traveling Salesman**

In travelling salesman problem one has to choose a rout with minimum net distance in traveling multiple cities assuming each city is visited once. We make a little adjustment to a problem by choosing a fitness function as inverse of net distance traveled, so that searching for max of fitness function is equivalent to searching for min distance. One can easily visualize this problem in a 2D coordinate system, where N points are randomly initialized with (x,y) coordinates. Then those coordinates are used to calculate distances for different routes. I chose 40 points(cities) for evaluation to keep computation less costly. Since values for coordinates are randomly given continuous values between zero and one they do not represent physical units.

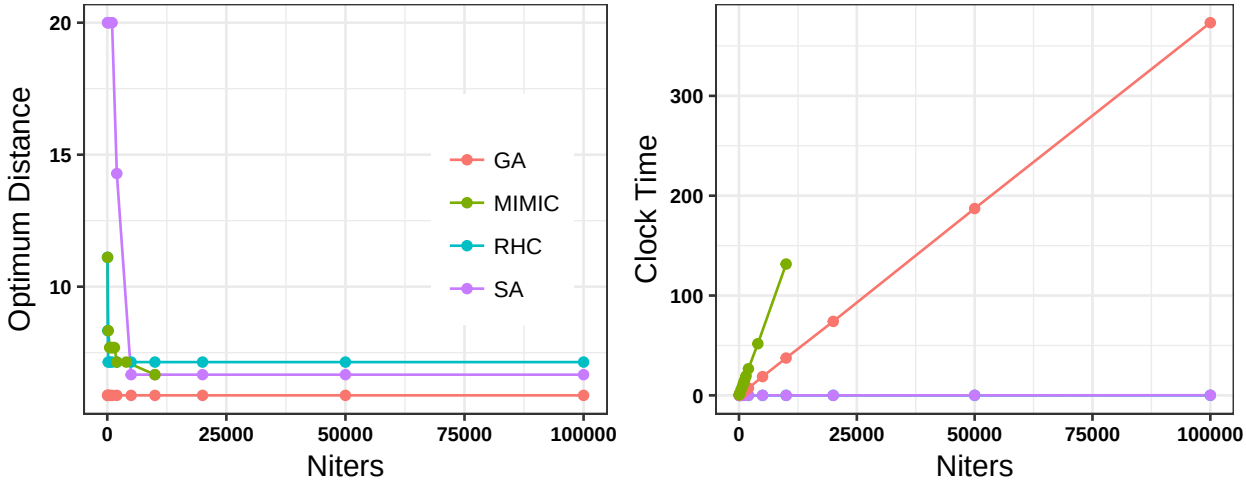


Figure 2: Traveling Salesman Performance

From all four optimization methods used GA gave the best performance as shown in Fig2. It reached global optima value within few iterations at Dist=6, which is quit surprising. This time GA performed better than MIMIC because MIMIC uses joint probabilities and for this problem, various optimal solutions may be significantly different because the order of the points visited matters. Both GA and MIMIC techniques were very computationally intense, thus iterations were done without repetition. SA performed as second best, which is not bad given it can only overcome local optimas if global optima is nearby. In TS problem search space grow exponentially with number op points and global optima may not necessarily be near.

Once optimizations are done, resulting rout corresponding to final best performed iteration is plotted in Fig3. By comparing path in 2D one can clearly see that GA demonstrate the best performance,

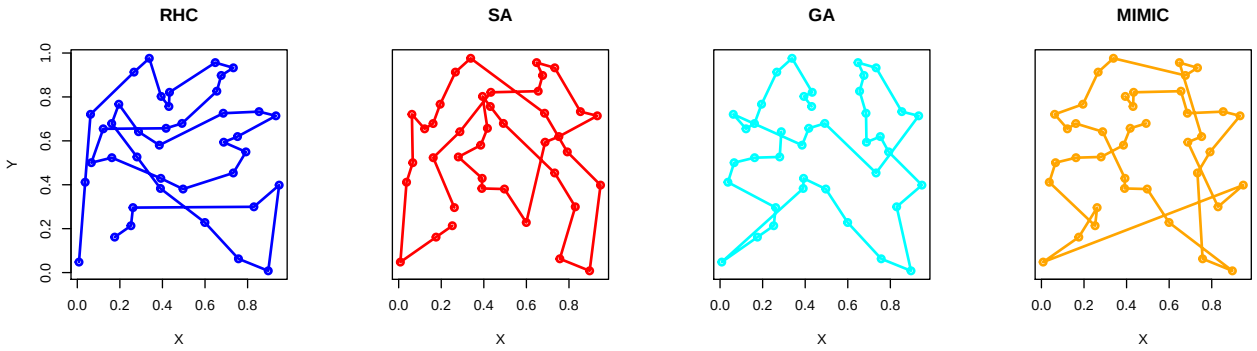


Figure 3: Final best model path for all four algorithms

where its trajectory show less crossings and connections between points are mostly made between nearest neighbors. In contrast trajectory for RHC and SA look like an entangled polymer with many crossings.

**Knapsack Problem**

Knapsack problem originated from backpackers who try to maximize the occupying volume in a knapsack(backpack) while remaining within the weight threshold such that the knapsack can be carried long distances. It is considered as NP-hard(at least as hard as the hardest problems in non-deterministic polynomial acceptable problems) optimization problem with a set of constraints.

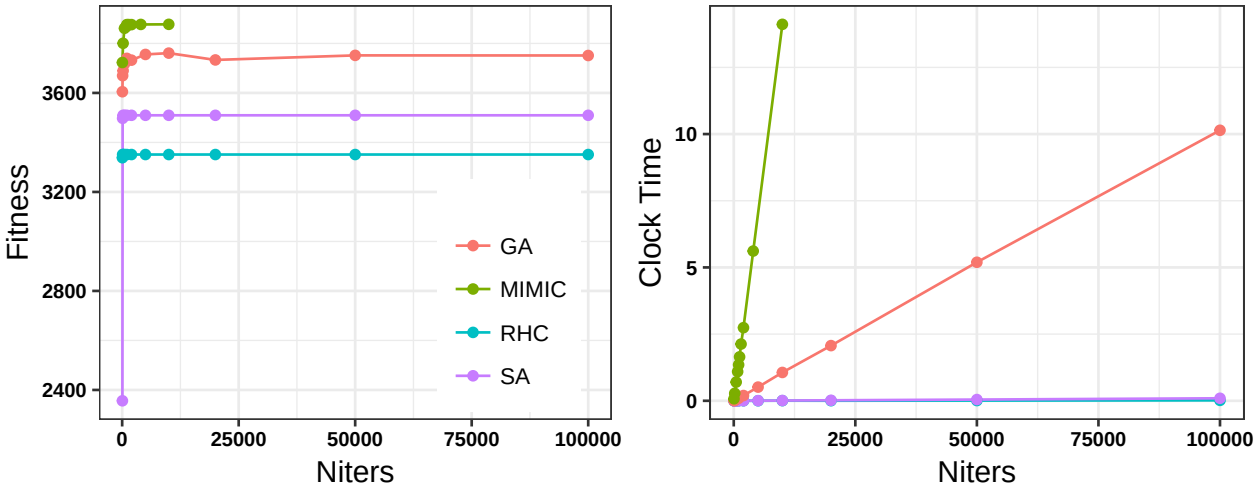


Figure 4: Knapsack Problem Performance

Again all optimizations are iterated up to 100000 times with 10 repetitions in order to take the average, wheres only MIMIC algorithm is iterated up to 10000 times. MIMIC model, once again proved its robustness by scoring the highest in Fitness curve (Fitness=38500) at about 10000 iterations.

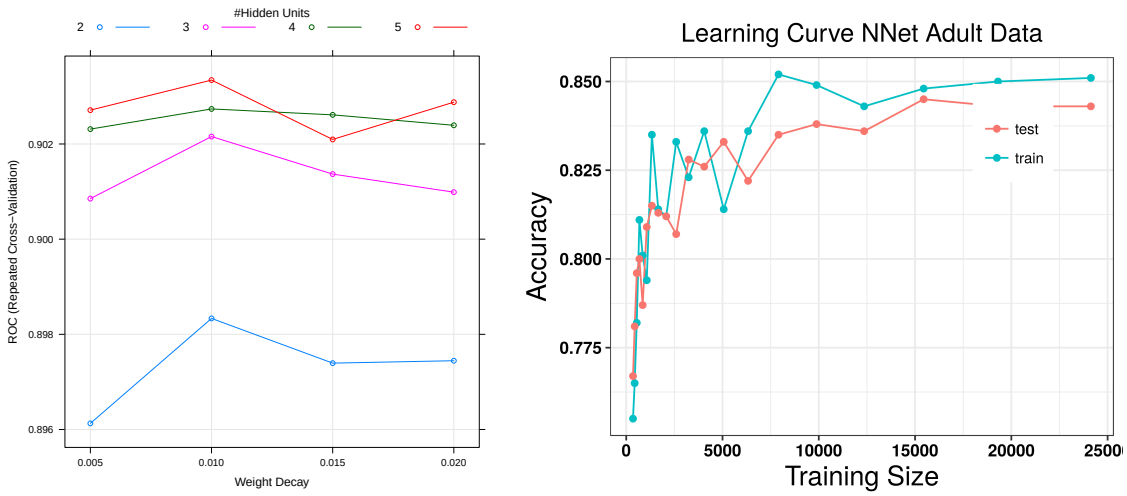


Figure 5: Cross validation(left) and learning curve(rights) plots for ANN taken from previous assignment.

## Part 2: ANN with Randomized Optimization

In this part of the assignment we are asked to use one of the data sets used from previous supervised learning assignment and apply feed forward neural network with randomized optimization technique for finding its optimal weights. Main purpose of the task is to compare performance of backpropagation against three randomized optimization techniques.

### Data Preparation and Setup

Adult data set used in a assignment 1 has 11 features and 45000 instances. For this assignment this data set is used making exactly same train test splitting as in a previous assignment. All multi-class categorical variables are mapped into binary class numeric variables[0,1], which increased number of features from 11 to 66. Data preprocessing is performed with R language and stored as csv inside clean_data folder. In Fig.5 we show results from previous assignment. As shown via cross validation plot(left) optimal value for hidden units were 5, and with that best performance achieved with backpropagation was 84% accurate on test set. For all optimization techniques same network structure is where input.units=66, only one hidden layer with hidden.units=5 and single output unit.

**RHC** is applied to feed forward nn to find optimal weights for network. Results are shown in Fig.6, where performance is reported via RMSE for both training and testing sets versus number

of randomized iterations.

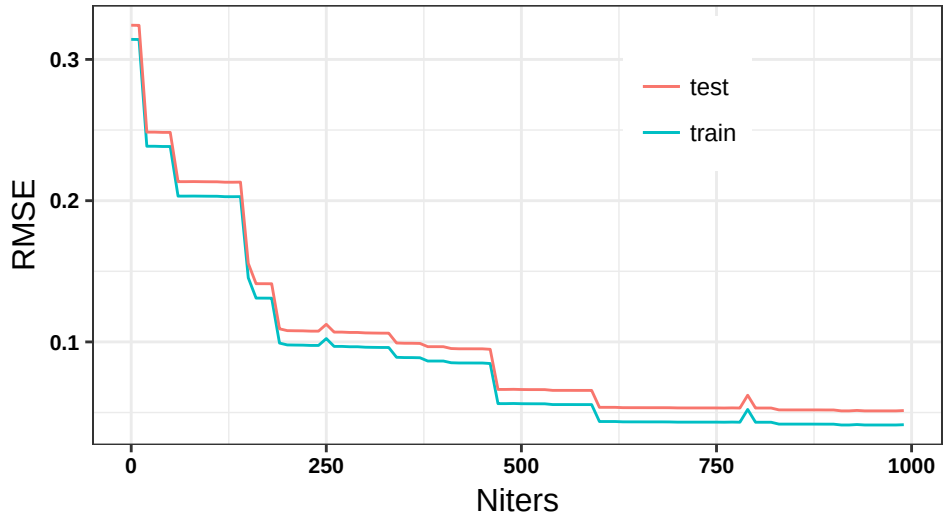


Figure 6: Randomized Hill Climbing on Adult Data

**SA** optimization method is applied to find best parameters(weights) for 1000 iterations performed at various cooling rates [0.15,0.35,0.55,0.70,0.95] Fig.7. The highest cooling rate (0.95) iteration shows high fluctuations in error as expected since at high cooling rate SA method is prone to fall into local optima. In contrast the slowest cooling iteration gradually falls into lowest RMSE by reaching global state.

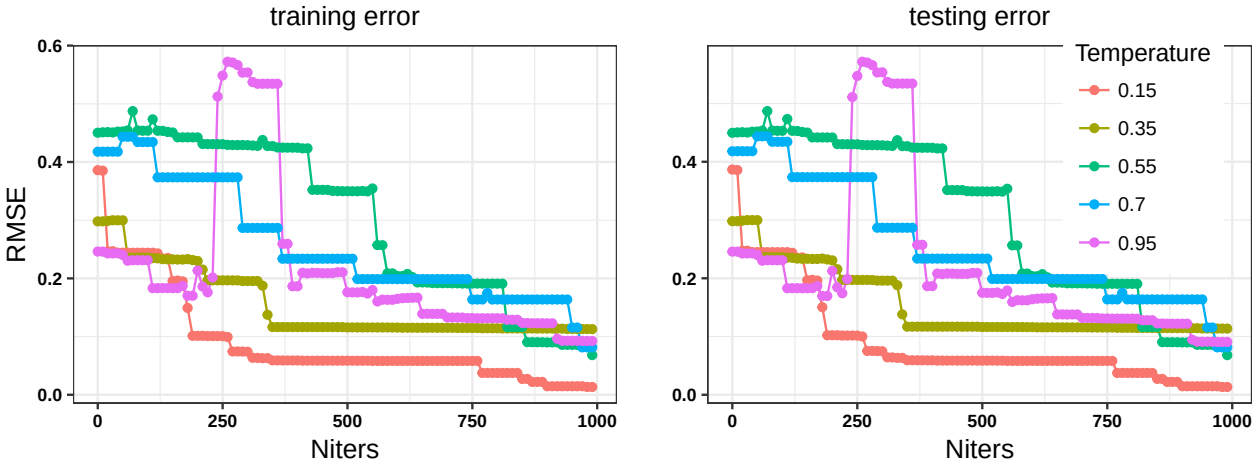


Figure 7: Simulated Annealing Cooling

**GA** algorithm is used with different parameters of p=50,mate[20,10] and mutate[20,10]in a grid search passion. Best performance is achieved by p=50, mate=20 and mutate=20 as shown in Fig.8. It showed best performance in contrast to other two algorithms by dropping the RMSE to as low as

0% within 200 iterations. In genetic algorithm, mutation provides exploration while cross-over leads the population to converge on good solutions (exploitation). Unfortunately, it is hard to understand how mate and mutate ratios affect exploration and exploitation. Thus, GA is known to be a black box for finding optimal solutions while unable to make it clear or intuitive.

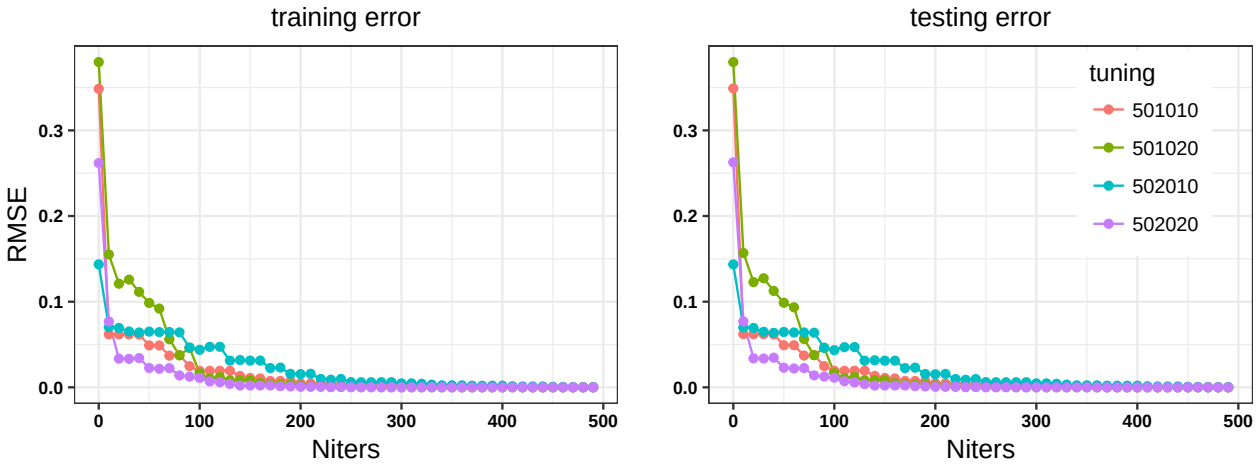


Figure 8: Genetic Algorithm RMSE for different parameters

## Conclusion

In the first part MIMIC gave best performance among all optimization techniques for all 3 problems except for traveling salesman problem, where GA algorithm outperformed MIMIC. For the second part of assignment, GA algorithm demonstrated best performance with RMSE as low as 0.1% and very high accuracy 99%. All three algorithms performed better(SA.RMSE=2%, RHC.RMSE=4%) compared to 84% accuracy obtained with backpropagation previously. This, of course, is problem dependent and further study is highly recommended.