GTID: ywang3564

Assignment 2: Randomized Optimization

Summary

This report includes two randomized optimization analysis: 1) apply three optimization algorithms on neural networks (RHC: randomized hill climbing, SA: simulated annealing, and GA: genetic algorithms). A BP(back propagation) result was also included as a control. Compared with BP, all three methods can optimize the problem with better fitness (in term of F1 score). 2) Four algorithms (RHC, SA, GA, MIMIC) were applied to solve three optimization problems: travelling salesmen (TSP), continuous peaks (CP), and flip-flop (FF). For each problem, some algorithms performed better than others, according to its intrinsic design. For example, GA had a far better performance in TSP than all others, while SA is slightly better in CP.

Introduction of background and dataset

The dataset I am using is **Titanic problems from Kaggle**, which includes demographic and detailed information for each passenger. Among all 2,200 passengers in Titanic, only ~700 were rescued after the sinking of Titanic. It was surprising to find that some factors greatly increased the rescue possibility of each passenger. So the problem itself is very interesting to explore. Furthermore, the size of the dataset is moderate and the number of features is low. Therefore, it's easier to apply different algorithms on it and compare their performance. Additionally, a lot of people working on Kaggle are improving the performance of this problem; it could be very helpful to borrow their experience in my analysis.

The whole dataset has 892 records with ground truth. After removing the NA values, the total number is 714. They are splitted as three: training data (#: 401), testing data (#: 179), and validate data (#:134). Training data is used for model training, and validated data is used for tuning the model. Testing data is used for evaluation of model. To make it convenient for all algorithms, I recorded the following factors to binary: Cabin class, Gender. Specifically for age and ticket fare, I re-scaled the value with (x-min)/(max-min) to make sure the range is from 0 to 1.

Method

- ABAGAIL
- Cmaron pipeline
- Fitness measurement: **F1 score** and **accuracy**, where TP: true positive, TN: true negative, FP: false positive, FN: false negative.
    - Precision = TP/(TP+FP)
    - Recall = TP/(TP+FN)
    - F1 Score = 2*(Recall * Precision) / (Recall + Precision)
    - Accuracy = (TP+TN)/ (TP+FP+FN+TN)

I Randomized optimization algorithms

1.1 Randomized hill climbing (RHC)

Randomized hill climbing is a variant of hill climbing. Different with hill climbing starting from the steepest uphill move, randomized hill climbing started with a random location and move towards local optima by comparing the current fitness and its neighbor's. By repeating this process, RHC can reach a local optima. Then it will restarts randomly to find another local optima. After searching all local optima, it can report the global optima as last. RHC can find the global optima although it's a greedy algorithm. Another feature for RHC is: it can converge quickly with a small memory load. RHC was implemented by RandomizedHillClimbing from ABAGAIL.

1.2 Simulated annealing (SA)

The term and philosophy of simulated annealing come from annealing in metallurgy, where the heating and controlled cooling was applied to increase the crystals size and reduce defects. Here, the algorithm is function of initial temperature and cooling rate, aiming to find a balance between new location and nearby neighbours. Initially, at high temperatures, the algorithm explores by randomly seeking new points and as it cools, it proceeds to evaluate neighbors for local optima. It is often used for discrete search space and find an approximation of global optima. SA was implemented by ABAGAIL on Java, using various cooling rate (0.15, 0.35, 0.55, 0.7, 0.95). [1]

1.3 Genetic algorithm (GA)

Genetic algorithm is a metaheuristic method inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms. GA are used to generate high-quality solutions to optimization and search problems by replying on process: mutation, crossover and selection. For each generation, GA allows mutation and crossover within population to elevate the benefit alleles and eliminate irrelevant alleles. The disadvantage of GA is: if the hypothesis space is large, it may take a long time to converge. SA was implemented by StandardGeneticAlgorithm in ABAGAIL, using inputs population Size (50), Mate (10,20), Mutate (10,20). [2]

1.4 Mutual-Information-Maximizing Input Clustering (MIMIC)

MIMIC is a framework for the analysis of the global structure of the optimization landscape. Different with other optimization algorithms, it remembers previous results. By defining probability densities to build the solution space, it can guide the next search to refine the local optima. One major issue for MIMIC is the time. MIMIC was performed using MIMIC in ABAGAIL on Java.

[1] https://en.wikipedia.org/wiki/Simulated_annealing
[2] https://en.wikipedia.org/wiki/Genetic_algorithm

2 Titanic result analysis

In this topic, I will compare the performance of four algorithms: BP (back propagation), RHC (random hill climbing), SA (simulated annealing), and GA (genetic algorithm). BP was used as a control to evaluate the performance of all other three.

2.1 BP

Fig1 shows the result for back propagation (BP) algorithm. Both left and right figure show a very similar pattern, indicating two evaluation function are actually coordinate with each other. They are different because of different definition. We are going to focus on the left figure: F1 score. Here, the NN was designed with input layer (7 nodes), hidden layer (6 and 3 nodes), and output layer. From the left figure, we can observe three lines, each represents the training, validating and test dataset. Based on pattern of three lines, there are four distinguish periods:

- Iteration 0 - 10: the algorithm quickly reached the local optima;
- Iteration 10 - 1000: the training data had a 5% higher F1 score than that of validation and testing;
- Iteration 1000 - 2000: this is a perturbation period, it seems like the algorithm leaves previous local optima;
- Iteration 2000 - 5000: the algorithm now have a better performance on training and validation, but a worse performance on testing

Conclusion: BP can quickly converge, with the iteration increased, the method will be overfitted on training data.

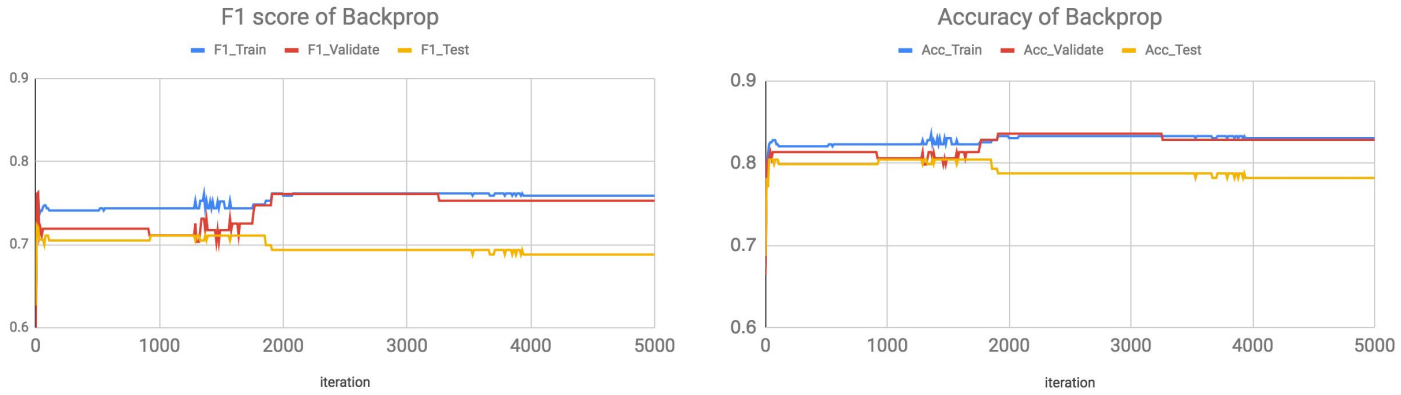


Fig1 F1 score and accuracy for BP

2.2 RHC

Following two figures are results from random hill climb (RHC) algorithm. Similar with BP, both evaluation showed a similar pattern. Unlike BP result, the RHC reached a local optima around 120. Then it had a perturbation period due to searching for the local optima. The performance of training and test is also different with BP, sometimes they had a similar performance. If they are different, the gap of training and testing is smaller than BP.

Conclusion: RHC converge at a moderate speed and improve the performance of testing. The variance of F1 score was caused by the searching of RHC.

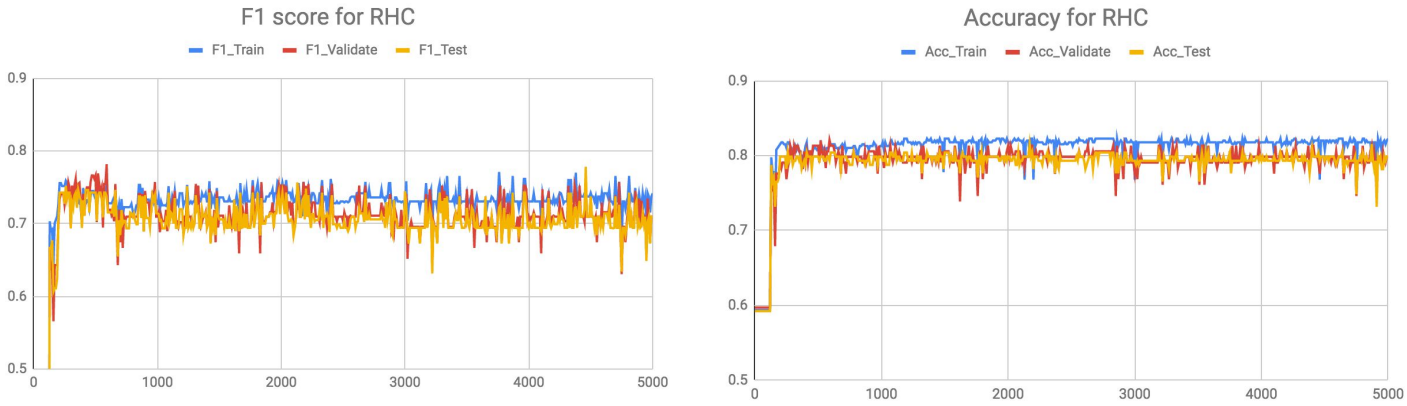


Fig2 F1 score and accuracy for RHC

2.3 GA

Among GA analysis, six different combinations was tested the best one is GA_50_20_10. GA needs about 1000 iteration time to reach the local optima, which is much slower than BP. But GA does not have discriminative performance for training ,validation and test. Although with iteration increased, the performance for testing dropped.

Conclusion: GA has a lower speed due to the search space, but has a non-discriminative performance for testing data.GA is the only one without a strong overfitting toward train. However, with iteration increased, the population become homologous and the performance may drop.

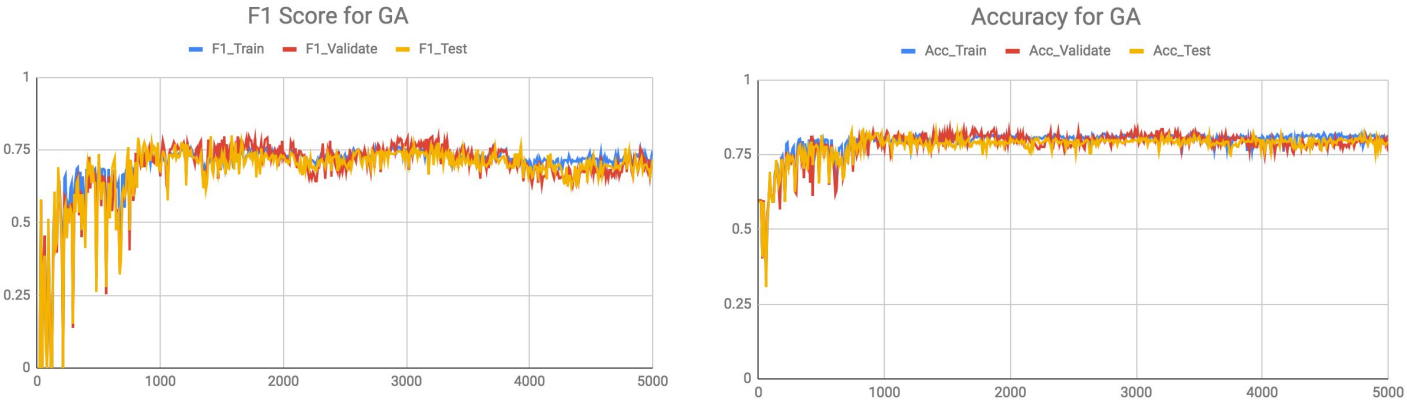


Fig3 F1 score and accuracy for GA

2.4 SA

For SA algorithms, the best cooling rate is 0.15 among all five combinations. Similar with RHC, SA also have a discriminative performance for training and test data.

Conclusion: SA can converge quickly, after a certain searching iteration, SA may have a little overfit towards training data.

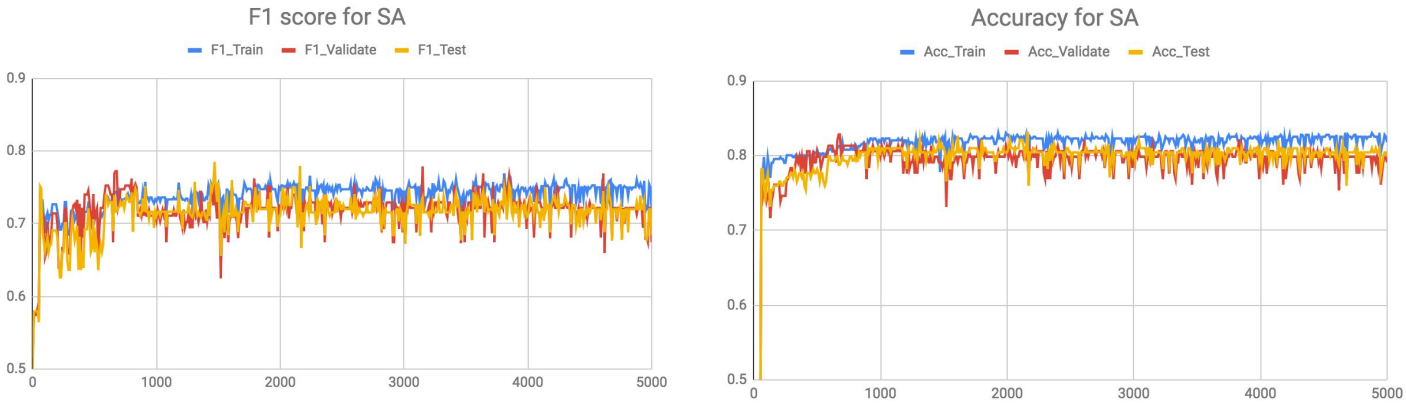


Fig4 F1 score and accuracy for SA

2.5 Combine all

Which one has the best testing fitness (F1 score)

The left figure is a general view of testing score for four algorithm, and the right figure is a zoomed one. From the right, we can see the best algorithm (evaluated by best test F1 score) is GA (yellow) and BP (blue) has the worse performance among four. **The rank for best testing fitness is: GA (0.8) > SA (0.784) > RHC (0.774) > BP (0.723)**. So all randomized optimization strategy can improve the performance. Among all, GA performed better than othr two among the complicated problems.

Fig5 F1 score for test data

How about the convergence time and the best fitness time?

From the figure above, we can observed that GA has the longest convergence time while BP has the lowest iteration time. The corresponding clock time (s) is: BP (0.0672, iteration 10) < RHC (0.267, iteration 200) < SA (0.523, iteration 600) < GA(4.06 iteration 800). GA has a large search space, therefore the time is longer than other three. The Titanic data has seven binary features, so the searching space for GA is actually very big.

Another question is: when did each algorithms get their best test F1 score. The real time is: BP (0.0672, iteration 10) < SA (0.717, iteration 1470) < RHC (1.130, iteration 4460) < GA (8.120, iteration 1580). Among all, GA has the longest searching time and convergence time. BP is on the opposite; it has the quickest convergence time and best score time. However, with the iteration increased, BP will become worse and worse on fitness. RHC is a greedy method, it converges at a short time, but spend a lot of time to find the best score. One additional advantage for RHC is: there is no need to tune the parameters, but GA and SA do.

3. Three problems

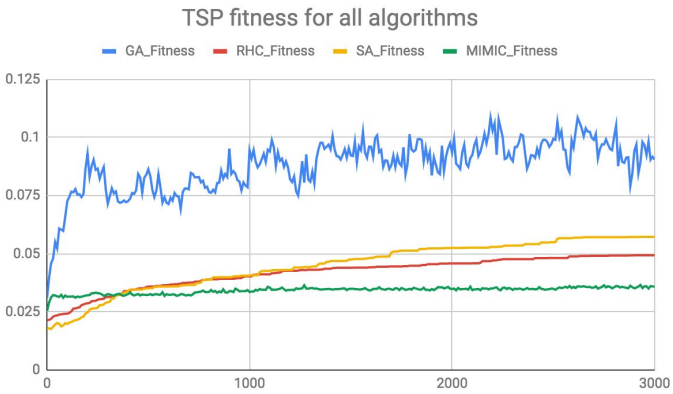
3.1 Travelling Salesman Problem (best:GA)

Travelling Salesman Problem (TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?" It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.. [3] This problem is included in ABAGAIL.jar.

I tested different algorithms with various parameters and choose the best one. The best parameters for SA is: CR=0.75. For GA, the best parameter was Pop=100, Mate=30, Mutate=30. For MIMIC, the best parameter was Pop=100, Keep=50, m=0.1.

Fig 6 indicates that the best methods is GA, while the worse one is MIMIC. For the clock time, MIMIC is using the longest time, while all others can solve this problem in a short time. Since TSP is a NP-hard problem, if you want to try some greedy search, you need to consider all (N-1)! possible paths, which is impossible for the algorithm to search if N is large. Therefore, algorithm like GA can handle complicated problems maybe a better choice. One disadvantage for GA is: the fitness have a small fluctuation caused by the dynamics of population.

[3] https://en.wikipedia.org/wiki/Travelling_salesman_problem



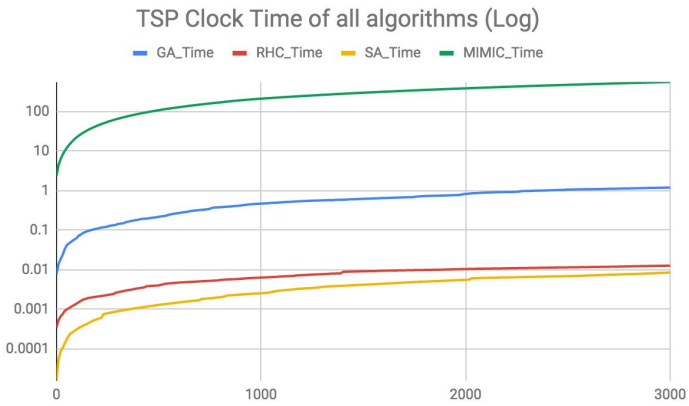
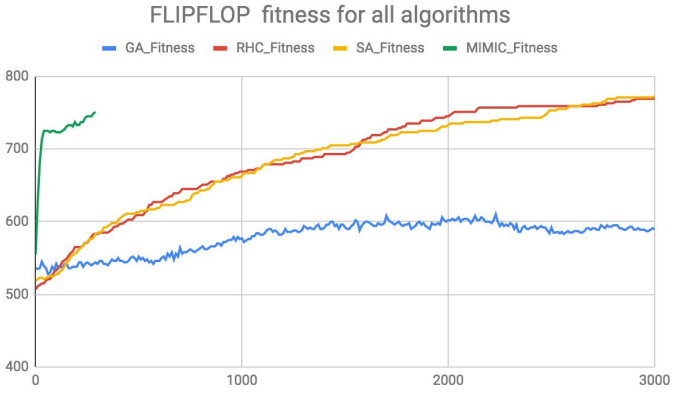


Fig6 Fitness for TSP

4.2 Flip Flop Problem (best: SA)

This problem is included in ABAGAIL.jar. Same with previous TSP problems, I tried different parameters for SA, MIMIC, and GA. The best cooling rate for SA is: 0.75; best parameter for GA is: Pop=100, Mate=50, Mutate=30; the best parameter for MIMIC is: Pop=100, Keep=50, m=0.1.

Fig 7 indicted the best algorithm is SA, while the worse is GA. MIMIC in my test terminated at iteration 300, but still acquired a very high fitness in a very short time. My guess is: MIMIC may performance best with the same iteration. Among all software, MIMIC still used the most time, while all others have a short time (<=1). Therefore, SA is the best algorithm for Flip Flop. The reason may be SA can work better in a local exploration and very fast. MIMIC may have the potential to be the best, but the speed for MIMIC is slow.



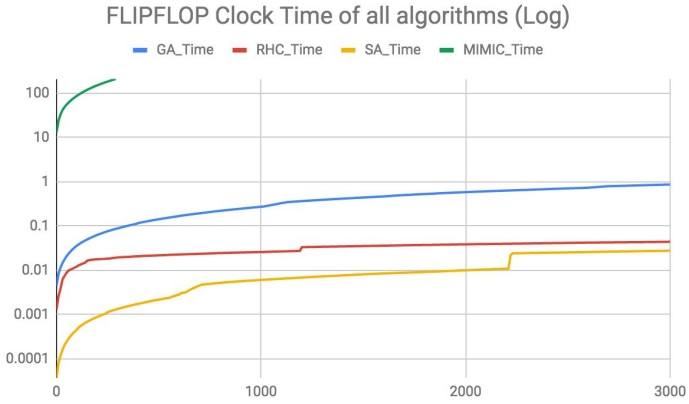


Fig7 Fitness for FF

4.3 Continue Peak (SA)

This problem is included in ABAGAIL.jar. Same with previous TSP problems, I tried different parameters for SA, MIMIC, and GA. The best cooling rate for SA is: 0.55; best parameter for GA is: Pop=100, Mate=30, Mutate=30; the best parameter for MIMIC is: Pop=100, Keep=50, m=0.7.

In Fig8, the best algorithm is SA, however, different algorithm was the best at a certain time. For example, MIMIC is the quickest learner to reach a local optima, but can not find another option. GA also find a local optima, and the performance was stable since the population is very similar. The time for MIMIC is still higher than others, SA is using the shortest time to get the best result.

The design for continuous peaks makes SA the suitable method to use. SA can search an approximate global optima, rather than a precise local optimum. By giving so many local optima on continuous peak, SA does not stop on any local one, but continue to increase the fitness by repeat the "annealing" methods. MIMIC is very rely on previous knowledge and can not escape the local optima.

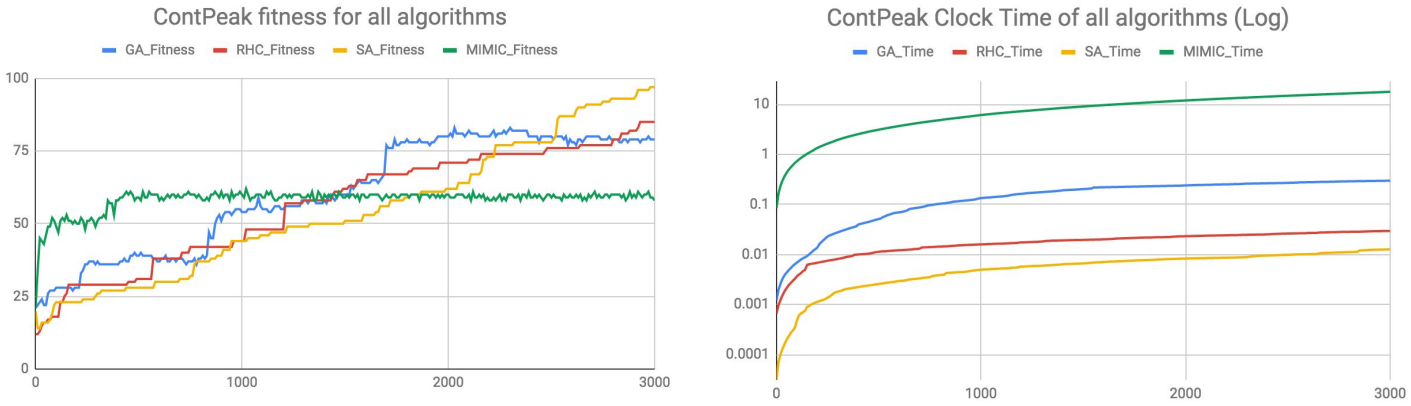


Fig8 Fitness for Continue Peak

Combining all conclusion so far, there are some features for all these four methods:

GA: Suitable for complicated problems, and NP-hard problem. May take a longer time; need tune parameters,

SA: can have an approximate global optima, fast to run; need tune parameters,

RHC: fast on every problems, no need to tune parameters, may acquire a moderate performance on any problems.

MIMIC: fast to convergence, may falled in a local optima, need more time and memory, need tune parameters