Assignment 2

Xiao Xiangjun
xxiao62@gatech.edu

INTRODUCTION

In PART I, three randomized optimization methods: randomized hill climbing, simulated annealing and genetic algorithm were used to solve one real data (used in Assignment 1). In PART II, four randomized optimization algorithms (one more MIMIC) were used to solve simulated optimization problems: OneMax problem (OMP), travelling salesman problem (TSP), continuous peaks problem (CPP).

Four randomized optimization algorithms were covered in course video comprehensively, here is the summary of these four algorithms implemented in this assignment [1]:

1) Randomized Hill Climbing

Randomized Hill Climbing (RHC) method is to find optima through exploring solution space and moving in the direction of increased fitness after each iteration. Random starting values are used to avoid local optimum.

2) Simulated Annealing

Simulated Annealing (SA) method is a modified randomized hill climbing, which puts more weight on the exploration of solution space through choosing sub-optimal next-steps with more probability. SA could increase the likelihood of finding global optima.

3) Genetic Algorithm

Generic Algorithm (GA) basically is one of evolutionary algorithms, it can produce new generations based on fitness of prior generations, it is a good way to find the best solution when several local optimums exist in function.

4) MIMIC

MIMIC is to exploit the underlying distribution of a problem to decrease the probability of finding sub-optimal portions of the solution space on future iterations. MIMIC tries to find common underlying structure using second-order statistics.

METHODS

- Datasets and Samples

Red Wine Quality [2] has 1599 instances, 11 numeric features and one response quality measurements, the quality will be transformed to dichotomous (good/bad) variable for classification analysis. It was directly download from website and all features were normalized prior to analysis. In Red Wine Quality dataset, quality score was transformed to Good/Bad indicator based on cutoff: quality > 6.5. Dataset was randomly separated to be a 70% training set and a 30% testing set.

- Analysis Platform

All data manipulation and machine learning procedures were performed in Python3.7, with such list of loaded libraries: sklearn, numpy, pandas, matplotlib.pyplot, time, mlrose [3].

- PART I: ANN implemented in Red Wine dataset

Standard MLPClassifier (backpropagation) and mlrose.NeuralNetwork with three optimization algorithms were used to analyze Red Wine dataset, and error rate in testing dataset was used to compare the performance of all methods. In order to make all comparisons equivalent, two layers of 15 nodes were used in all models: **[15,15]**. Hyperparameter, **max_attempts, [5, 10, 20, 30, 40]** were used to compare performances for each algorithm except for genetic algorithm with **mutation_prob in [0.01,0.05,0.1,0.2,0.3]**, and then all algorithms were compared using the best hyperparameter. Iteration: **[1,10,50,250,500,750,1000,1250,1500,2000]** was also used.

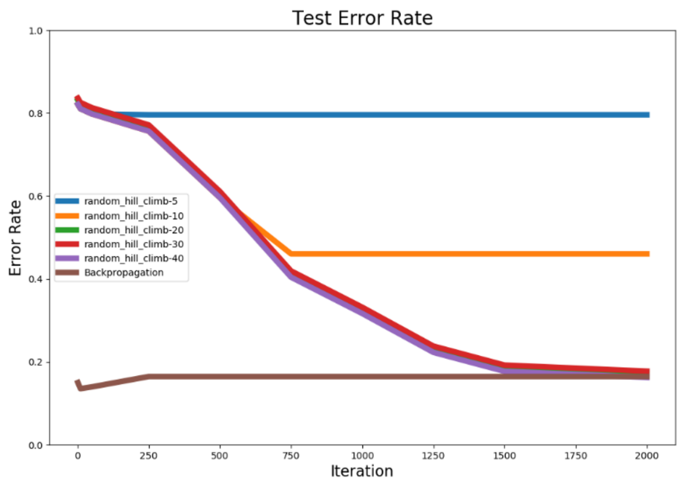
- PART II: Fitness Function

In simulated optimization problems: OMP, TSP, CPP, four algorithms were applied and the fitness value for each iteration was recorded through default setting: curve = True. The best algorithm among 4 candidates was further investigated through changing hyperparameter(s). Finally, the size of problem was also investigated through changing problem size and its effect on the performance of four algorithms. All runs were repeated 10 times to get mean fitness values to avoid unstable estimations.

RESULTS

- ### PART I: ANN implemented in Red Wine dataset

1) <u>Randomized Hill Climbing Algorithm (Figure 1: Error rate/Time vs. iteration)</u>



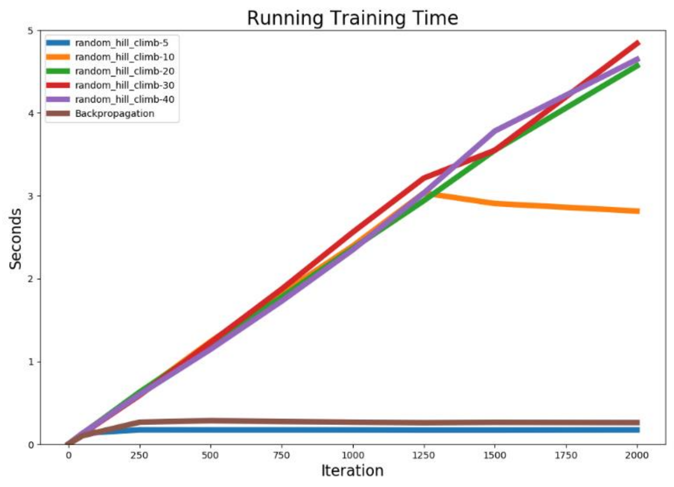


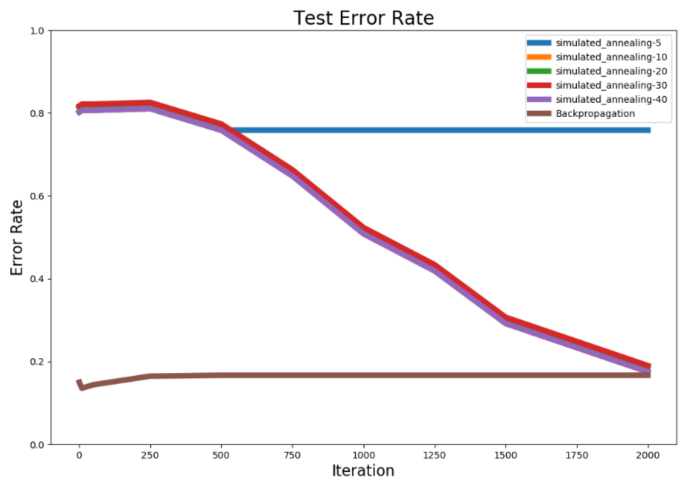
Figure 1: Randomized Hill Climbing (RHC) Method used in ANN implemented in Red Wine Data: Hyperparameter is max_attempts **[5, 10, 20, 30, 40]**, iteration is in **[1,10,50,250,500,750,1000,1250,1500,2000].** Standard backpropagation method was used as a baseline in plot. **Please note that max_attempts [20, 30, 40] were almost overlapped**.

- Discussion:

When max_attempt equals 40, RHC method achieves the same error rate as baseline at iteration 2000, it is obvious that backpropagation algorithm converged very fast after 200 iteration. Therefore, in running training time plot, there is no change for backpropagation after 200 iterations. The same situation is for max_attempt = 5, which seems to be stuck at iteration ~100 with an inferior local optimum. The best max_attempt value is **40**.

The hyperparameter max_attempt looks like a very promising parameter to tune the performance of algorithm. It can sufficiently differentiate performances of different values; the reason is that larger max_attempt value can increase the chance of finding better local optimum.

2) <u>Simulated Annealing Algorithm (Figure 2: Error rate/Time vs. iteration)</u>



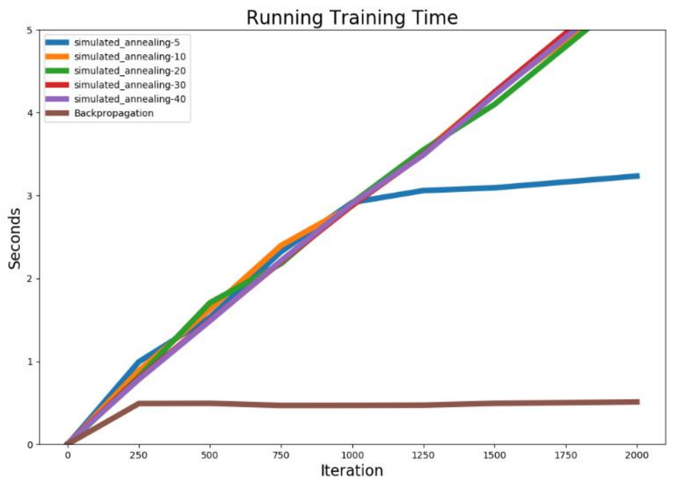


Figure 2: Simulated Annealing Algorithm (SA) used in ANN implemented in Red Wine Data: Hyperparameter is max_attempts **[5, 10, 20, 30, 40]**, iteration is in **[1,10,50,250,500,750,1000,1250,1500,2000].** Standard backpropagation method was used as a baseline in plot. **Please note that max_attempts [10, 20, 30, 40] were almost overlapped**.

- Discussion:

When max_attempt equals 40, SA method almost achieves the same test error rate as baseline at iteration 2000. When max_attempt equals 5, algorithm converged on iteration ~600 with an inferior local optimum. All other choices of max_attempt had the similar performance, except for minor differences in running training time used.

In SA algorithm, max_attempt value made little difference in performance in [10, 20, 30, 40], which may be due to its intrinsic property that other hyperparameters dominate its performance when max_attempt value is reasonable large.

3) Genetic Algorithm (Figure 3: Error rate/Time vs. iteration)

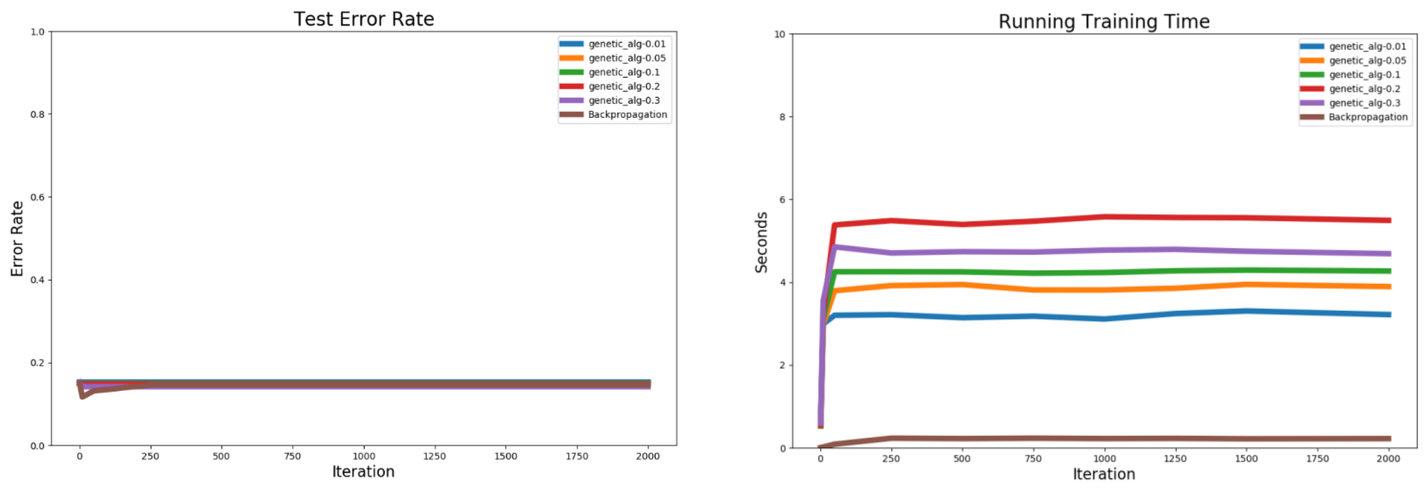


Figure 3: Genetic Algorithm (GA) used in ANN implemented in Red Wine Data: Hyperparameter is **mutation_prob [0.01,0.05,0.1,0.2,0.3],** iteration is in **[1,10,50,250,500,750,1000,1250,1500,2000].** Standard backpropagation method was used as a baseline in plot.

- Discussion:

Genetic Algorithm (GA) is very interesting that all values of max_attempt, learn rate and pop_size have the similar test error rate, which indicates that the number of random restarting, learn rate and pop size are not sensitive to final test error rate. Therefore, mutation probability was used as hyperparameter to tune, although the differences were still quite similar. The possible reason is that genetic algorithm is stable for red wine dataset.

GA algorithm indeed had different running time for mutation prob values. When mutation probe is 0.01, it had the minimum time used for converging. GA can converge even faster than backpropagation within several iterations, but with longer time, which is consistent with GA's properties, a population based algorithm.

Based on Figure 3, the candidate best hyperparameter used in Genetic Algorithm is mutation prob = 0.01 with other default hyperparameters in the model.

4) <u>An overall comparison among RHC, SA, GA (Figure 4: Error rate/Time vs. iteration)</u>

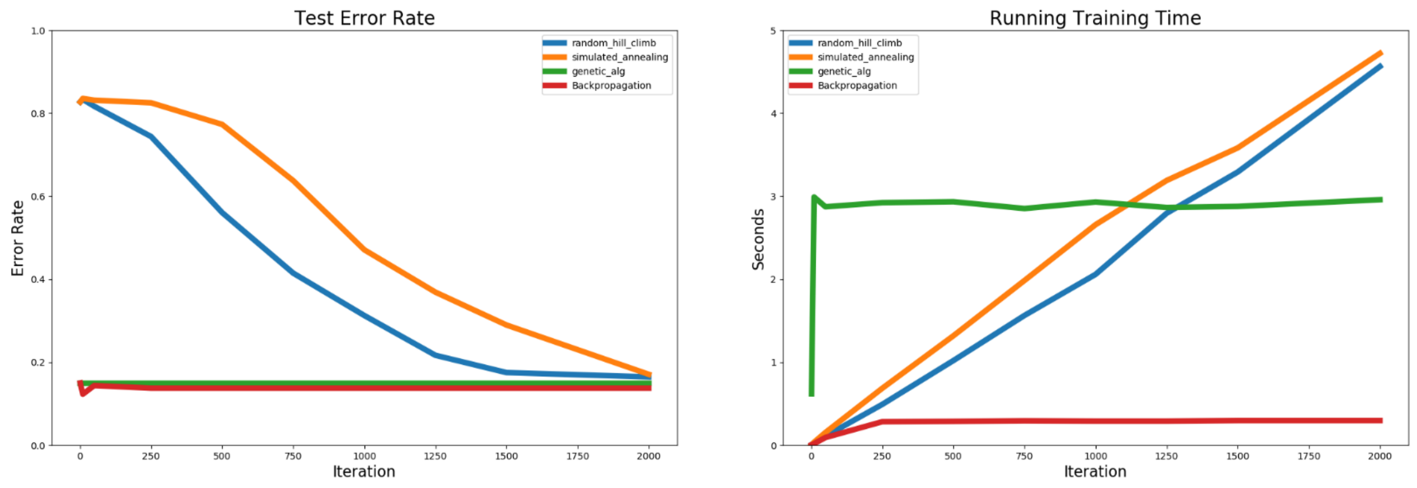


Figure 4: RHC, SA and GA with the best hyperparameter used in ANN implemented in Red Wine Data: iteration is in [1,10,50,250,500,750,1000,1250,1500,2000]. Standard backpropagation method was used as a baseline in plot.

- Discussion

Based on Figure 1, 2 and 3, if only considering test error rate, the best hyperparameter max_attempts are: 40 for randomized hill climbing, 40 for Simulated Annealing, and mutation probe 0.01 for Genetic Algorithm. Figure 4 left panel shows comparison for three algorithms using the best max_attempts values. At ~2000 iterations, all methods achieved the similar test error rate (~0.15). Both backpropagation and genetic algorithm converged fast at ~50 iterations, SA and RHC almost converged at the end of iterations (2000).

Figure 4 right panel shows time comparison among algorithms, as expected, genetic algorithm took much more time before convergency than other methods.

Comparisons in Figure 4 clearly indicate that:

- <u>Backpropagation method performed the best among all methods, it can converge fast, running time is short.</u>
- <u>Genetic algorithm is as good as backpropagation if only test error rate was considered, but its running time is the longest among all methods.</u>
- <u>Randomized hill climbing and Simulated Annealing methods have moderate performances in both error rate and running time.</u>

- **PART II: Four Algorithms implemented in OMP, TSP, CPP**

1. OneMax Problem (OMP) with problem size = 50 (Upper) and 100 (Lower), Figure 5: Convergency for four algorithms (Left), hyperparameter pop size tuning in the best algorithm mimic (Right)

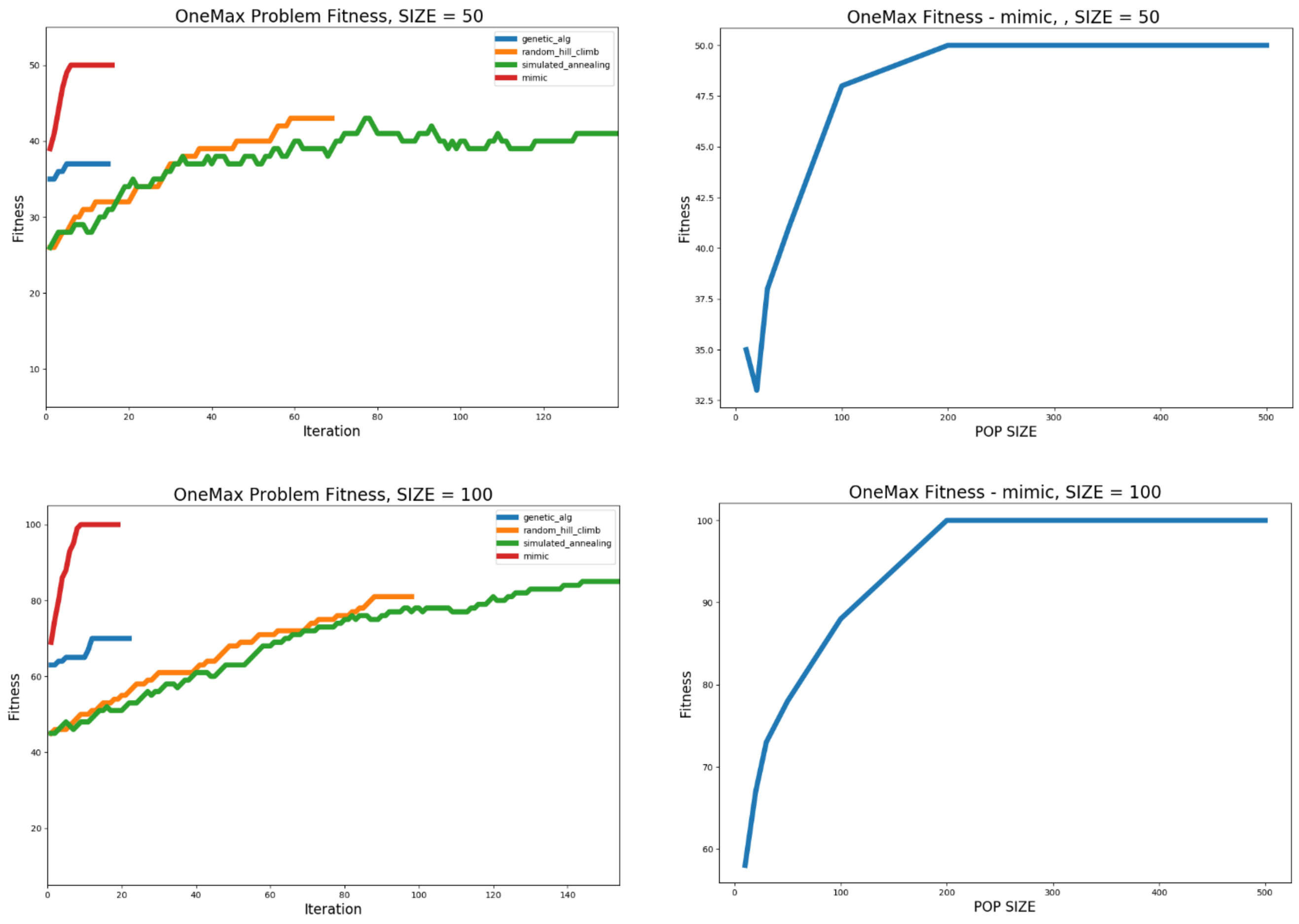


Figure 5: OneMax Problem with size of state array = 50 (Upper). The running time for randomized hill climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA), mimic is **[0.0, 0.00456, 0.4951, 18.63446],** size = 50. The running time for randomized hill climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA), mimic is **[0.005, 0.015, 0.718, 80.608],** size = 100. All running parameters are the default values (e.g., max_attempts = 10, except for hyperparameter tunning.

- Discussion

OneMax problem is maximization problem with known global optimum value, 50 and 100 in this analysis, therefore, it is good to compare the performance of different algorithms.

1) **Convergency:** only mimic algorithm found the global value correctly.
2) **Iteration:** both genetic algorithm and mimic can converge within 10-20 iterations.

3) **Time:** RHC and SA algorithms ran much faster than other two even they needed more iterations to converge.
4) **Problem Size:** larger set had longer running time as expected.
5) **Hyperparameter Tuning:** mimic has stable performance after pip size > =200.
6) **Conclusion:** mimic looks like the best algorithm not considering running time.

2. Continuous Peaks Problem (CPP) with problem size = 50 (Upper) and 100 (Lower), Figure 6: Convergency for four algorithms (Left), hyperparameter pop size tuning in the best algorithm mimic (Right)

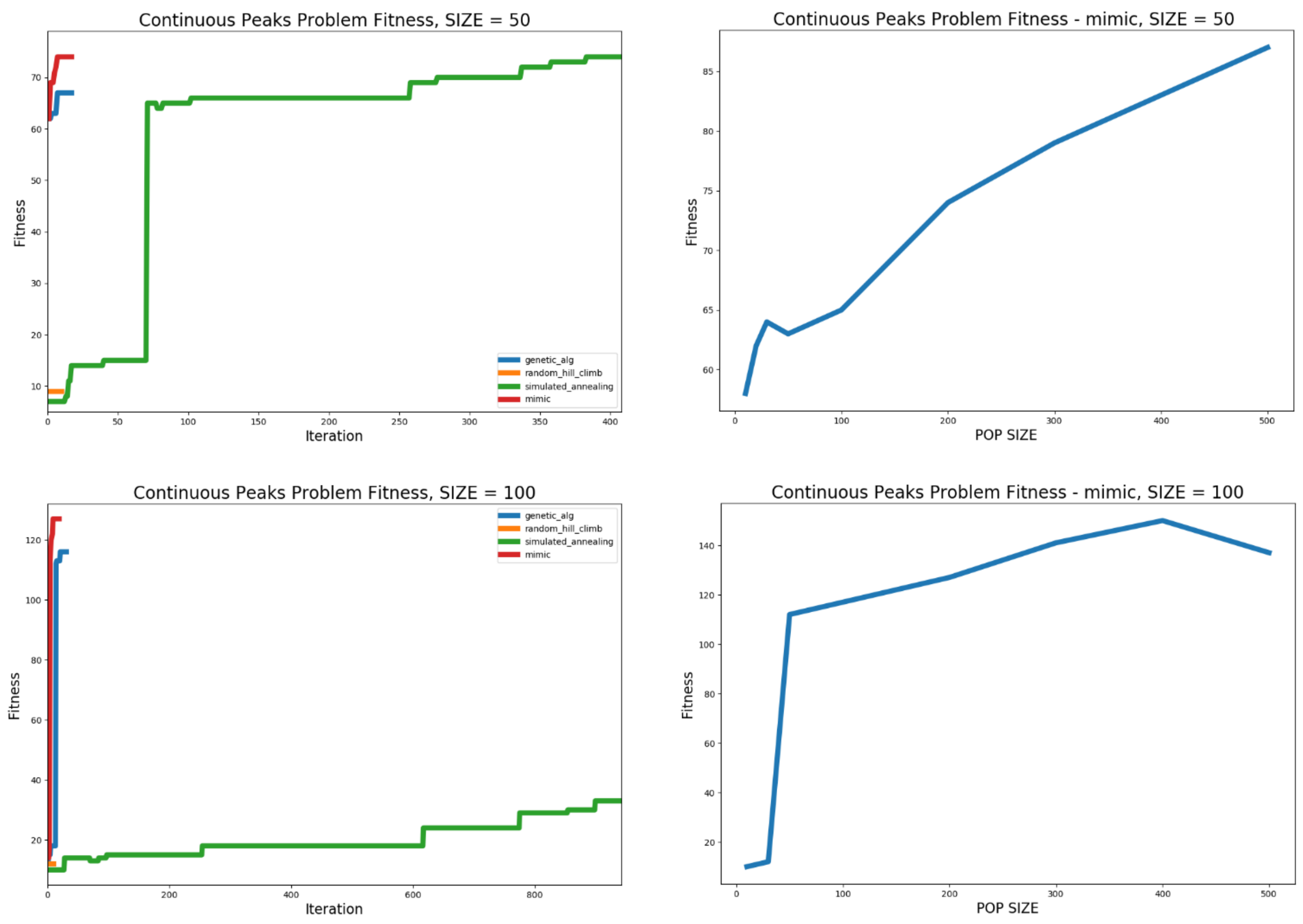


Figure 6: Continuous Peaks Problem (CPP) with size of state array = 50 (Upper). The running time for randomized hill climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA), mimic is **[0.0, 0.031, 0.454, 18.183], s**ize of state array = 50. The running time for randomized hill climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA), mimic is **[0.001, 0.1639, 1.415, 76.18],** size=100. All running parameters are the default values (e.g., max_attempts = 10), except for hyperparameter tuning.

- Discussion

Continuous Peaks Problem (CPP) is also a maximization optimization problem. The same strategy was used as OneMax problem, where problem size was set to be 50 and 100.

1) **Convergency:** only mimic algorithm found the largest optimization optimum within 10 iteration, simulated annealing (SA) almost achieved the same value after 400 iterations. RHC performed badly and converged at a very inferior local optimum.
2) **Iteration:** genetic algorithm, randomized hill climbing, and mimic can converge within 10-20 iterations.
3) **Time:** RHC and SA algorithms ran much faster than other two even SA needed more iterations to converge.
4) **Problem Size:** larger set had longer running time as expected, simulated annealing performed much better in small set (50) than large set (100).
5) **Hyperparameter Tuning:** mimic has better performance when pop size is larger (~400-500)
6) **Conclusion:** mimic looks like the best algorithm not considering running time, simulated annealing may not be stable in performance.

3. Travelling Salesman Problem (TSP) with size = 10 (Upper) and 16 (Lower), Figure 7: Convergency for four algorithms (Left), hyperparameter pop size tuning in the best algorithm mimic (Right)

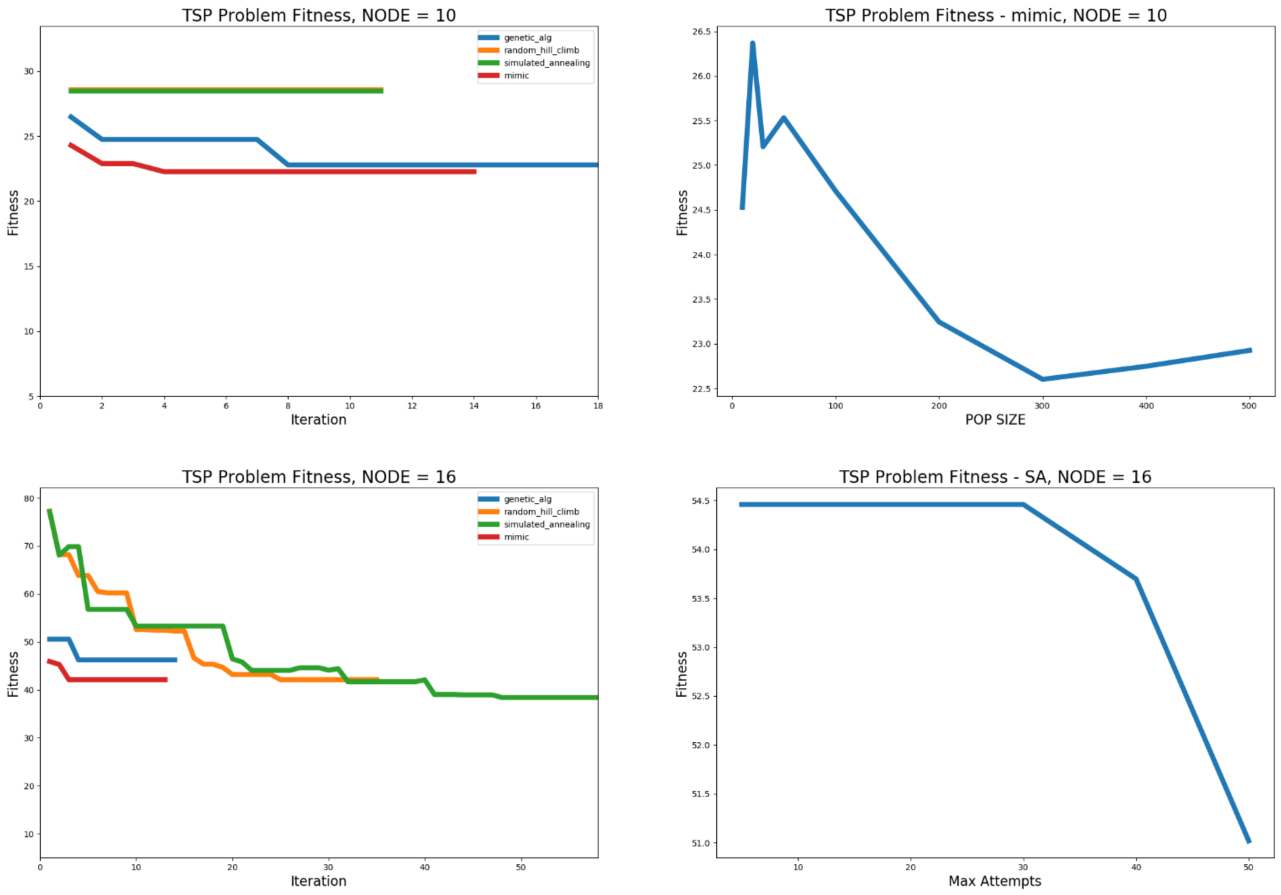


Figure 7: Travelling Salesman Problem (TSP) with size of node = 10 (Upper), 16 (Lower). The running time for randomized hill climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA), mimic is **[0.002, 0.0015, 0.806, 2.179],** node = 10. The running time for randomized hill climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA), mimic is **[0.008, 0.014, 0.881, 3.787],** node = 16. All running parameters are the default values (e.g., max_attempts = 10), except for hyperparameter tuning.

- Discussion

Travelling Salesman Problem (TSP) is a minimization optimization problem. The same strategy was used as two previous problems, where problem size was set to be 10 and 16 (node number).

1) **Convergency:** mimic algorithm performed well in both small and large problem sets, it converged at ~15 iterations. In node = 10, genetic algorithm also achieved similar optimum at iterations 20. In node = 16, simulated annealing algorithm performed best and converged at iteration 60.
2) **Iteration:** In small problem set, all four methods can converge before 20 iterations. In large problem set, genetic algorithm and mimic can converge within 10-20 iterations.
3) **Time:** RHC and SA algorithms ran much faster than other two even they needed more iterations to converge, just like two previous problems.
4) **Problem Size:** larger set had longer running time as expected, simulated annealing performed much better in large set (16) than small set (10).
5) **Hyperparameter Tuning:** In small problem set, mimic has better performance when pop size is larger (~300). In large problem set, simulated annealing seems to perform better when max_attempts increase and got the best fitness when max_attempts = 50.
6) **Conclusion:** mimic looks like the best algorithm not considering running time, even in large problem set it looks a little bit inferior to simulated annealing. Simulated annealing may not be stable in performance. The possible reason for simulated annealing is that problem set of large size could have more complex object function, and if max_attempts are fixed for all four algorithms, simulated annealing may perform better than other methods.

DISCUSSTION

The performances of four algorithms: RHC, SA, GA, mimic and comparisons can be summarized as:

1) Genetic Algorithm (GA) has the best performance in red wine dataset, but its performance is not as goods as mimic in OMP, CPP, TSP problems.
2) RHC, SA, GA can finally converge to the similar error rate in red wine dataset if enough iterations are allowed, but in OMP, CPP, TSP problems, these three methods did not converge at the similar value. Possible reason is that simulated problems have different fitness function as in ANN.

3) RHC and SA converged slowly in red wine dataset at ~2000 iterations, but in simulated problem, they normally ran much faster and converged at several hundred iterations. Possible reason is that ANN fitness function with large dataset may be more complex than simulated problems.
4) RHC algorithm performed not well in most cases, either converged at an inferior local optimum or needed long iterations to converge. RHC method is straightforward in mathematics, may only be good at limited optimization problems, and that is why other randomized optimization methods were developed.
5) Hyperparameters, such as max_attempts and pop size, were carefully investigated in this study. Both can sufficiently differentiate performances of candidate algorithms.
6) The intrinsic property of mimic method is its advantages on complex fitness function and longer running time in each iteration, although normally it will have much less iteration number. OMP, CPP and TSP problems seem to have complex enough fitness function, since the initial settings of problem size is not small. Therefore, the fact that mimic performed the best among four algorithms is reasonable.

CONCLUSIONS

In this assignment, 4 randomized optimization algorithms were investigated (Table 1):

1. MIMIC algorithm had the best performance if running time was not considered.
2. Genetic Algorithm worked the best in real red wine dataset.
3. Population-based or distribution-based algorithm is superior to traditional algorithms, such as randomized hill climbing and simulated annealing.

Table 1: Comparison

	Time of Convergence	Iteration of Convergence	Overall Performance
Randomized Hill Climbing	Fast	Long	Moderate
Simulated Annealing	Fast	Long	Moderate
Genetic Algorithm	Slow	Short	Good
mimic	Very Slow	Short	Best

REFERENCES

[1] Comparison of Four Randomized Optimization Methods, Feb 23, 2020, retrieved from https://bambielli.com/posts/2018-07-22-comparison-of-four-randomized-optimization-methods/

[2] Red Wine Quality, Feb 1, 2020, retrieved from https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

[3] mlrose. https://mlrose.readthedocs.io/en/stable/source/opt_probs.html