Randomized Optimization: Comparison of Four Learning Algorithms

Herbert Ssegane GtID#903755945

1.0 Part 1: Random search on three optimization problem domains

This part of analysis explores four random search algorithm's performance on three selected optimization problem domains. The four algorithms are randomized hill climbing (RHC), simulated annealing (SA), genetic algorithm (GA), and mutual information maximum input clustering (MIMIC). Three criteria were used to measure algorithm performance; 1) convergence to an acceptable solution, 2) number of iterations to converge, and 3) time to converge. The chosen optimization problem domains are 1) N-Queens problem (8-Queens), 2) Knapsack problem (KS), and traveling salesman problem (TSP). The next subsections briefly expound on the three problems and the corresponding performance by each optimization algorithm.

1.1 8-Queens

The 8-queens problem is an example of an optimization problem given some constraints. Thus, the target is not finding a single optimal solution, but to find feasible solutions that meet preset constraints. In the game of chess, a queen can attack other pieces in any direction (horizontally, vertically, and diagonally). The optimization problem is to place 8 queens on an 8x8 chess grid such that no queen can attack another queen? Problem set up is such that *queens[i]* is the row number for a queen in column *i*. The *fitness* is designed as number of diagonal conflicts. Thus, a fitness of zero (0) is optimal. For additional details refer to *Russel & Norvig (2010)*.

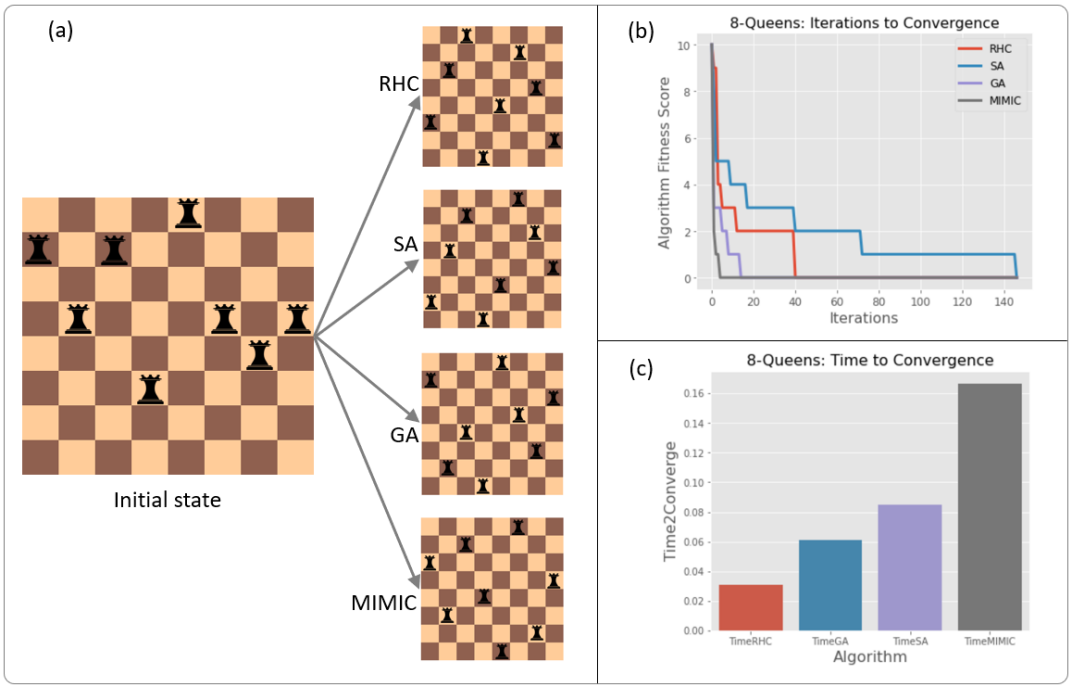

Figure 1: initial state of the 8-queens problem and final states by algorithm (a), algorithm performance based on iterations (b) and time (c) to converge to a solution(s), respectively. Comparison is after parameter tuning

Figure 1a shows solution states whose finesses are zero, the desired optimal fitness. All algorithms converge to a solution in less than 200 iterations. This is partly attributed to the problem size of 8x8 chess grid. MIMIC converges in the least number of iterations (4) while it takes the most computation time, about 5 times the least computation algorithm, the RHC with restarts (figure 1b & figure 1c). The RHC and SA have stepwise decay (figure 1b) compared to GA and MIMIC, an indication of getting stack at a local minimum over several iterations. This is attributed to the inherent nature of the steepest descent from an initial state while the population-based GA and MIMIC start at various possible solutions. And thus, show improvement on subsequent iteration.

1.2 Knapsack problem (KSP)

The Knapsack problem requires to pack a set of items (valuables) into a bag given item weights and corresponding values such that total items do not exceed the bag capacity while maximizing total value. Below is the problem formulation where w_i is the weight of the i^{th} item, v_i is the corresponding value, and W is the sack capacity

$$maximize \quad \sum_{i=1}^{n} v_i w_i \qquad given\ that \quad \sum_{i=1}^{n} w_i \leq W$$

For this analysis, the search space included 50 items with weights varying from 0.5 lbs to 94 lbs and values varying from $10 to $862. The maximum bag capacity is 844 lbs. total available weight and value for all items are 1,876 lbs and $8,604, respectively. The data is modified example from Google's *ortools* python library.

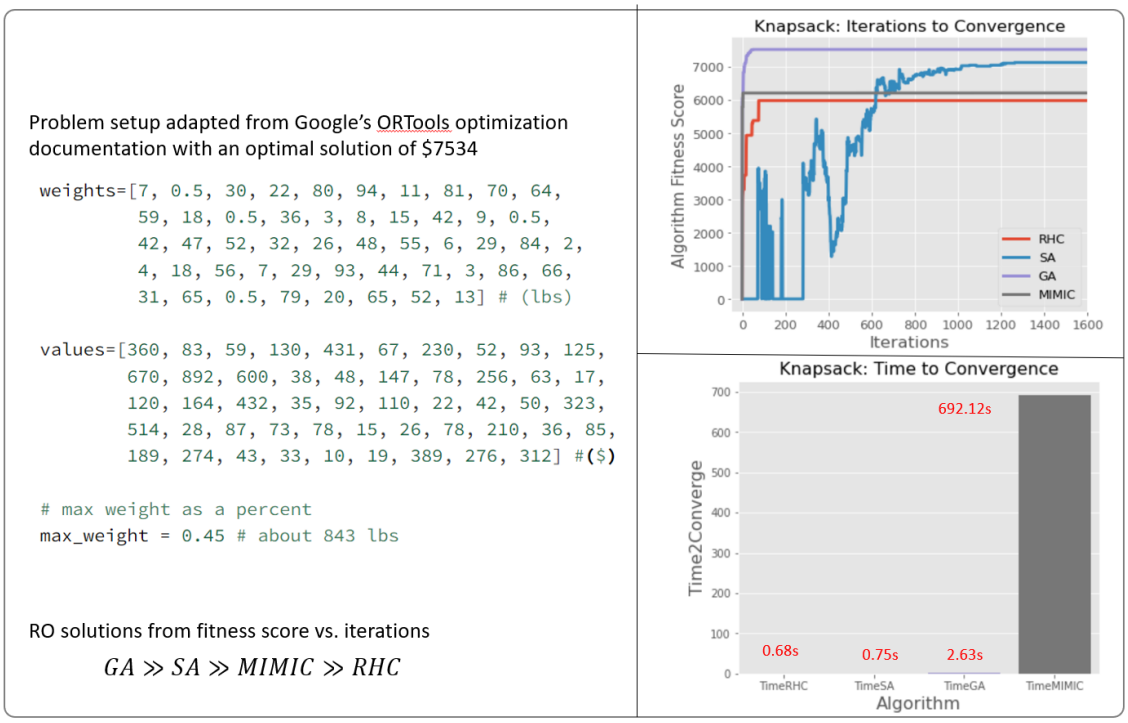

Figure 2: The knapsack problem weights, values, and bag capacity. The figure also captures the number of iterations to converge and the time computational complexity of each algorithm. Comparison is after parameter tuning.

The fitness of the KSP is the total value of the selected items that do not exceed the bag capacity. The GA outperformed all other algorithms with a fitness of $7,512 after converging in about 200 iterations

over 2.63 seconds. This value is the closest to the optimal value given by the *ortools* python package. The *ortools* uses the multidimensional branch and bound search algorithm. The SA would also reach optimal solution given more iterations. Both, the RHC and MIMIC converged to a local optimal in less than 100 iterations. Varying MIMIC *population_size* and *keep_percent* did not improve the results. Similar to 8-Queens, MIMIC had the largest computation complexity, more than 260 times the time requirement for the GA.

1.3 Traveling Salesman problem (TSP)

The TSP is a routing problem, where given a set number of cities with corresponding intercity distance pairs, what is the shortest route for visiting each city only once and return to the origin city. Applications of TSP include planning of activities to achieve a set goal, vehicle routing for package pickup, planning of movements of drills during the manufacture of microchips, and movement of machines on shop floors. For additional details refer to *Russel & Norvig (2010)*.

For this analysis, the *mlrose_hiive* python library was used to generate a TSP of 10 cities. The fitness is the route distance, and the shortest route distance is the optimal solution. Below are some observations

1. All algorithms converge to an optimal route distance of 664.96
2. RHC converges within 64 iterations with 20 restarts
3. SA converges at the same 107 iterations for lower temperatures of 0.1, 1.0, and 10.0.
4. GA converges faster with high population sizes. For example, with 500 population size it converges in 8 iterations compared to 32 iterations with population of 50, both at mutation rates of 10%.
5. MIMIC converges within 8 iterations with a higher population size of 1000.
6. Therefore, parameter tuning is key for optimal solution for each method

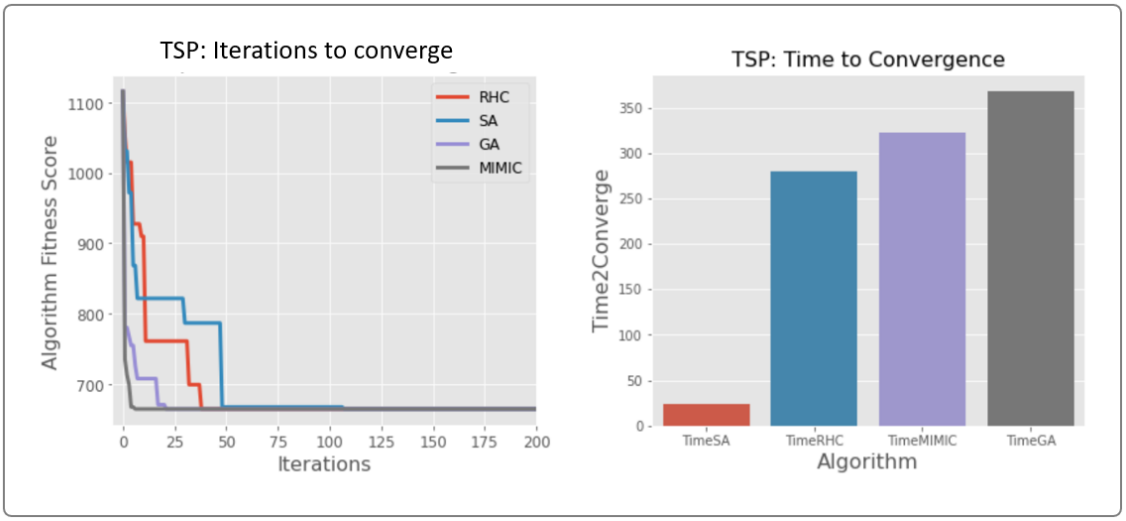

Figure 3: Comparison of performance of four optimization algorithms on the traveling salesman problem (TSP). Performance is captured by the number of iterations to converge and the time computational complexity of each algorithm, under optimal parameters for each algorithm.

2.0 Part 2: Fitting Neural Network weights using random optimization

2.1 Heart failure prediction data (HFP)

The dataset consists of 918 data points with 11 features to predict heart disease (*HeartDisease* column in Table 2). A value of one (1) for heart disease and a value of zero (0) for no heart disease. This is also a two-class classification problem, where the *HeartDisease* feature is the target. For a detailed description of each feature and data sources, refer to *Fedesoriano (2022)*. This data is relevant because it demonstrates application of supervised learning for predictive analytics in the health care industry, knowing that heart failure is the leading cause of deaths, globally.

2.2 Data pre-processing, train, and test data

The supervised learning algorithms expect numeric values. Thus, data preprocessing was implanted on both datasets prior to exploring the learning algorithms. The target feature of the WBCD was the only categorical feature. It was transformed into a binary feature where zero (0) represented benign cancer and one (1) represented malignant cancer. The HFP data has four categorical features (Sex, ChestPainType, RestingECG, ExerciseAngina, and ST_SLope). These features had a maximum cardinality of 4 and thus were transformed into dummy variables to capture the different categorical levels without risk of the curse of dimensionality. Finally train and test data was scaled using a min-max scaler of the train data (refer to equation below).

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \; where \; 0 \leq x' \leq 1$$

Each dataset was split into 70% and 30% for train and test datasets, respectively. The splitting was stratified using the target to maintain proportionality within the two data splits. The train data is used to generate the log-loss curve as a function of number of iterations. For this analysis, the maximum iterations were set to 5000 for all algorithms. The test data is used to assess generalization of the final weights by each algorithm on the same neural network architecture. Below are the hyper-parameters used for the network on all weight optimization algorithms (RHC, SA, and GA).

```
(hidden_nodes=[200],  activation='relu', max_iters=5000,
bias=True, is_classifier=True, learning_rate=0.001,
early_stopping=True, clip_max=5, max_attempts=1000,
random_state=42,curve=True)
```

2.3 Baseline performance

The baseline model performance is the multi-layer perceptron network with a single hidden layer of 200 neurons, trained using the *adam* stochastic gradient-based optimizer.

2.4 Comparison of performance of the algorithms

The smoothness of all three curves depicts the impact of a very low learning rate of 0.001. The GA converges in fewer iterations compared to RHC and SA. However, the corresponding time to converge is exponentially higher for the GA compared to other two (30s vs, 2,340s).

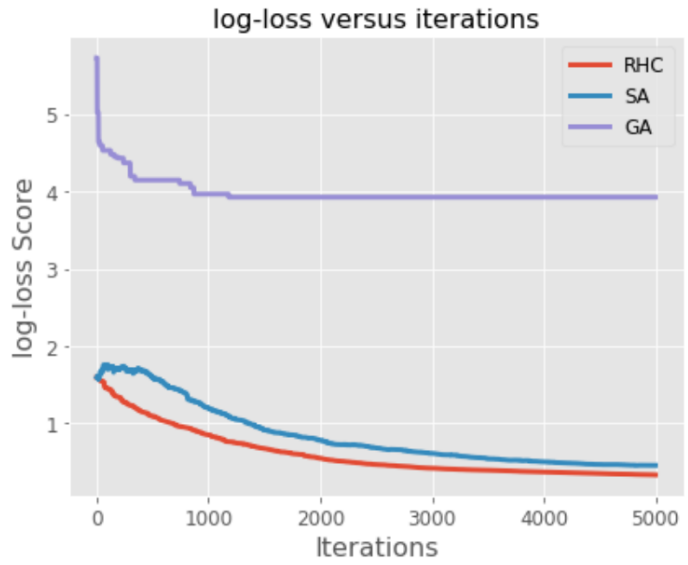

Figure 4: Number of iterations to model convergence during training for three optimization algorithms

Table 1 compares performance of the three random optimization algorithms to a baseline stochastic gradient descent (adam). The GA outperforms RHC and SA on four of the six model performance metrics. Its metrics are the closest to the stochastic gradient descent. Therefore, for this data, the GA is a better random optimization algorithm (Figure 4 and Table 1). However, it has high computation complexity. The high computational complexity is related to differences on how these algorithms scale with problem complexity. The higher the search space, the higher the time effort for population-based methods compared to steepest ascent or descent algorithms.

Table 1: Comparison of optimal weights by each algorithm on a test heart failure data

Optimization Algorithm	Accuracy	AUC	Precision	Recall	F1	Train time (s)	Prediction time (s)
Stochastic gradient descent (adam)	0.88	0.88	0.88	0.92	0.90	0.60	0.03
Random Hill Climbing (RHC)	0.84	0.84	0.87	0.84	0.85	30.60	0.03
Simulated Annealing (SA)	0.84	0.84	0.88	0.83	0.85	37.60	0.03
Genetic Algorithm (GA)	0.85	0.85	0.84	0.90	0.87	2340.00	0.03

5.0 Conclusion

Below are general remarks on all algorithms across the various optimization problem domains.

1. Hyper-parameter tuning is key to all algorithms to get results close to the global optimal. For RHC, the number of restarts while for SA the cooling parameter (temperature) are the key parameters. For 8-Queens only 2 restarts were required by RHC while SA preferred relatively higher temperature values.
2. The RHC and SA are computationally simple (fast) and can approximate global solutions.
3. The GA and MIMIC are computationally complex (slow) but require less iterations to converge to an approximate global optimal. Their disadvantage is that they require more parameter tuning compared to RHC and SA.
4. For this analysis, the GA was the best algorithm at approximating global optima across all problem domains.
5. The MIMIC required the most computation time effort

References

1. Fedesoriano 2022. https://www.kaggle.com/fedesoriano/heart-failure-prediction

2. OR-Tools 7.2. Laurent Perron and Vincent Furnon. https://developers.google.com/optimization/

3. Hayes, G. (2019). *mlrose: Machine Learning, Randomized Optimization and SEarch package for Python*. https://github.com/gkhayes/mlrose. Accessed: *03-01-2022*.