Randomized Optimization Report

Four Optimization Algorithms:

Randomized Hill Climbing:

Randomized hill climbing will run hill climbing algorithm multiple times with randomly different starting points. So in each run, we will end up with different local optima.

This algorithm cannot guarantee to return a global optima but will give a good local optima. The results depends on how many times we rerun the algorithm and whether the starting points can cover a wide range of values.

Simulated Annealing:

Simulated annealing algorithm is a random-search technique and tries to tune the temperature to balance between exploration and exploitation. If temperature is large, the algorithm behaves like random walk. If temperature is small, it behaves like hill climbing.

This algorithm can give a good solution, but it may take long time to run and cannot tell you whether it find a optimal solution or not.

Generic Algorithm:

GA is random optimization technique. It states with a population. Best selected parents will be used to generate high-quality offspring. Random changes will be applied to new generation including crossover and mutation. Crossover is used to randomly take the genes from the parents. Mutation means we will change some values to add new features in the offspring.

This algorithm is easier to understand and works fast. But it may get stuck in local optima which might be solved using crossover and involve more iterations. It is also a little difficult to design a function and representation.

MIMIC:

MIMIC algorithm is designed to model the probability distribution. It searched over solution space and does well with the estimate of structure.

This algorithm can converge faster and need fewer iterations, but one iteration may take far more time. MIMIC can also have less evaluations, so it has lower cost. However, if the fitness function does not have probability distribution, MIMIC may get stuck in local optima.

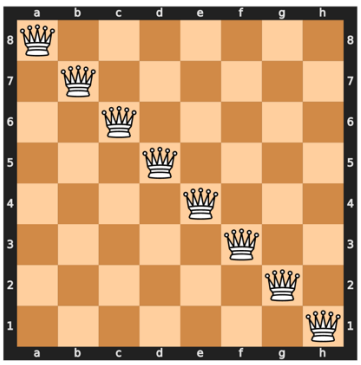
Optimization Problem 1: 8 Queens Problem

Description:

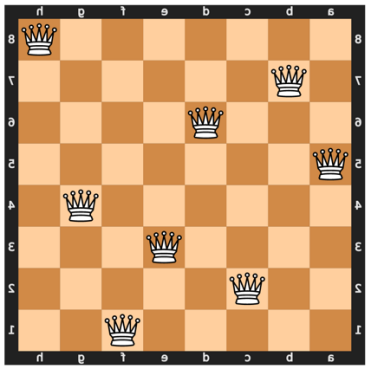
In a 8x8 chessboard, we are given 8 queens. The goal is to place 8 queens in the chessboard and none of them can attach each other in the same row, column and diagonal. In other words, if a queen has been places in a cell, we should not place other queens in the same row, column and diagonal. To denote the position of each queen, we can start from left upper corner and mark the row number starting from 0. And then each column only has 1 queen.

Why interesting?

This optimization problem is interesting because we want to maximize the number of queens who cannot be attached by any other queens. In other word, we can use state vector to count the number of pairs of queens of not being attacked and try our best to maximize this count number.

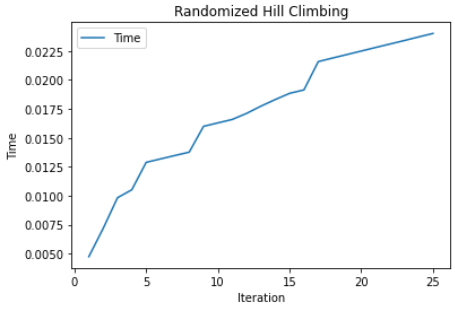


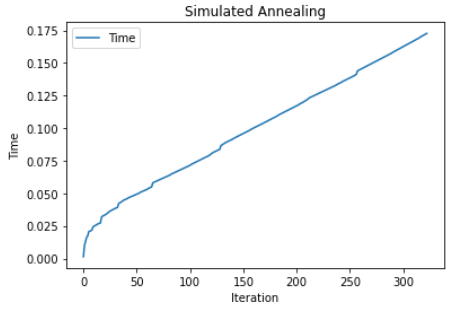
Initial---------->Optimization

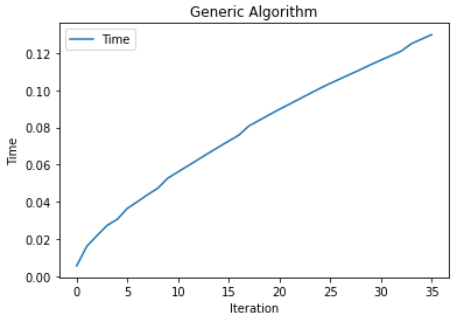


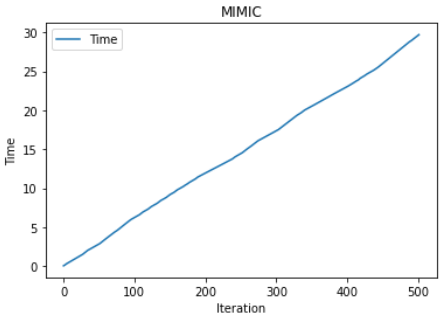
Comparison of Algorithms:

Running Time:

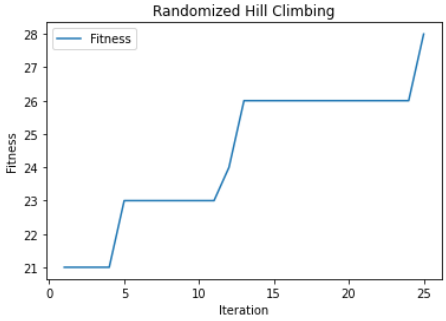


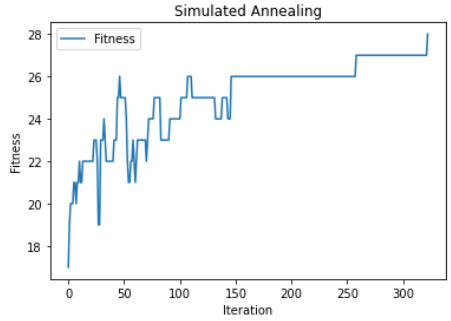


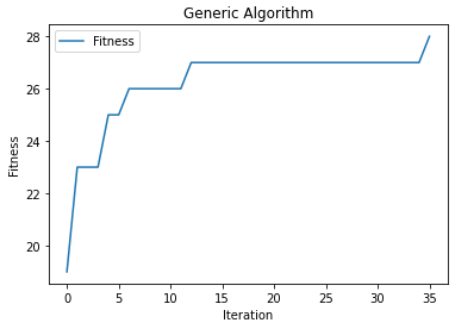


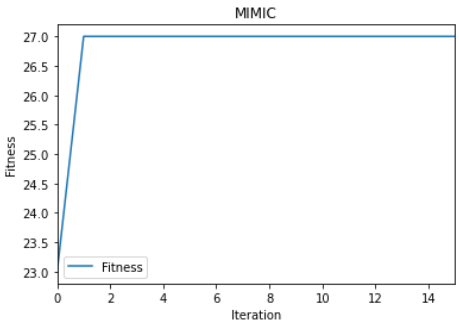


Fitness Score:

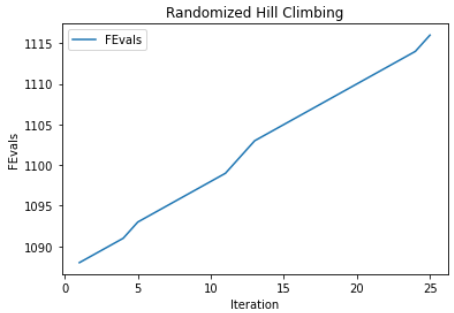


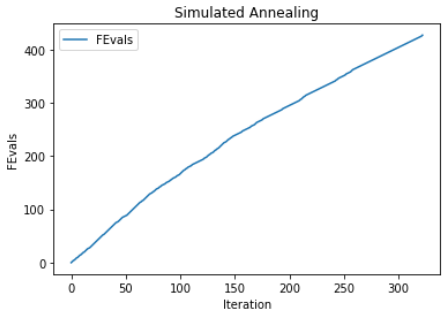


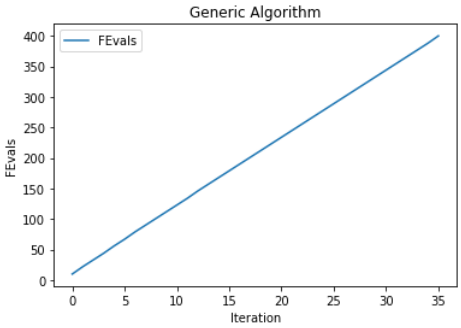


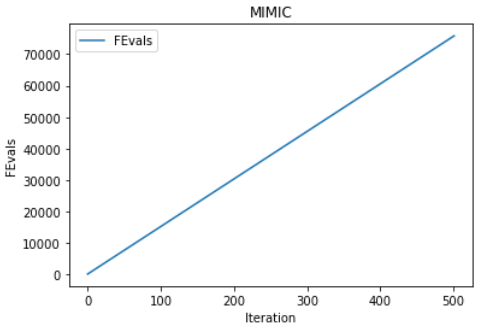


Number of Evaluations:









Summary:

Algorithm	Fitness	FEvals	Time	State
Randomized Hill Climbing	28	1116	0.793033	[5, 2, 0, 7, 3, 1, 6, 4]
Simulated Annealing	28	427	0.163399	[7, 1, 3, 0, 6, 4, 2, 5]
Genetic Algorithm	28	400	0.112018	[5, 2, 4, 7, 0, 3, 1, 6]
MIMIC	27	75802	29.696543	[7, 4, 1, 3, 0, 6, 3, 5]

I think the best approach is to use generic algorithm (shorten as GA) with population of 10 and mutation rate of 0.5. Although generic algorithm has same fitness scores as randomized hill climbing and simulated annealing, it uses least time and evaluations. So, GA works very fast in 8-queen problem. On the other hand, the fitness score from GA increases very fast in the first few iterations based on the graph above. Maximum fitness score is 28 because 8 queens can make up of 28 pairs and we expect all of them are not attacked by others.

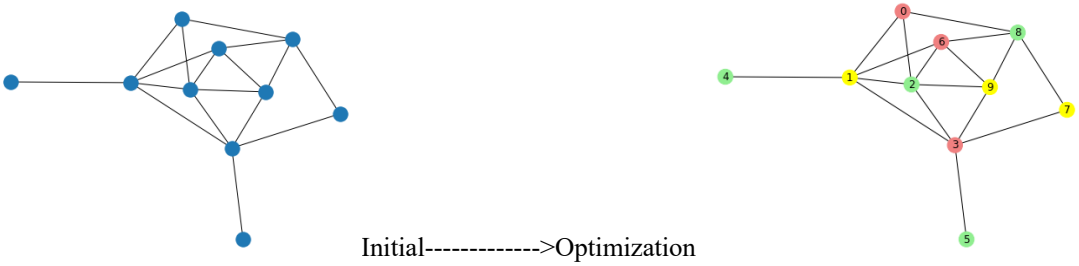
Optimization Problem 2: Max 3 Colors Problem

Description:

This is a problem that we need to find maximum pairs of nodes so that if a node is colored, its neighbors should be colored with different colors. In other words, we need to find a solution to make sure each node should have different colors from the adjacent neighbors. We also define how many colors which can be used in the graph. In this problem, I will solve a maximum 3-colorable problem.

Why interesting?

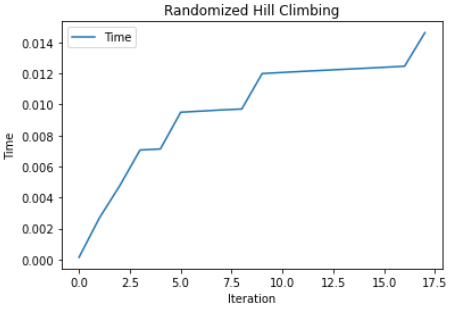
This problem is interesting because max 3 color problem require us to find the maximum pairs of nodes which have different colors from their adjacent neighbors. It is NP-hard for general graph.

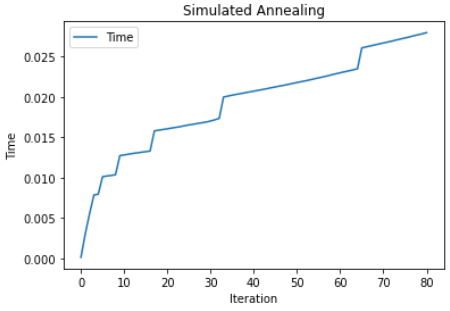


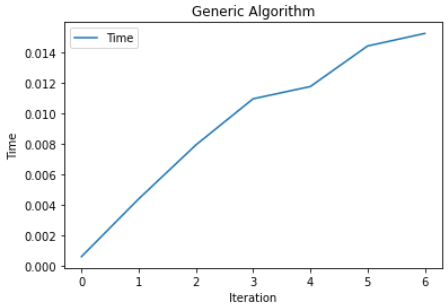
Initial---------->Optimization

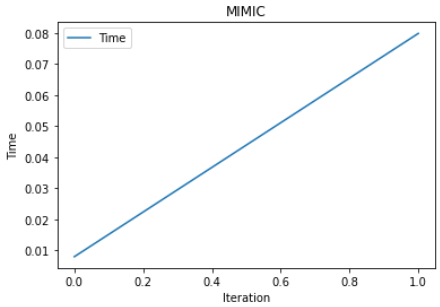
Comparison of Algorithms:

Running Time:

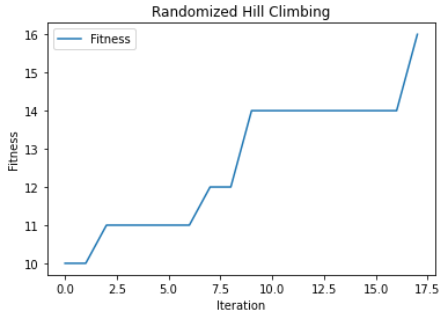


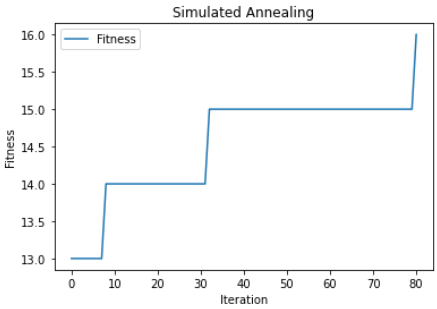


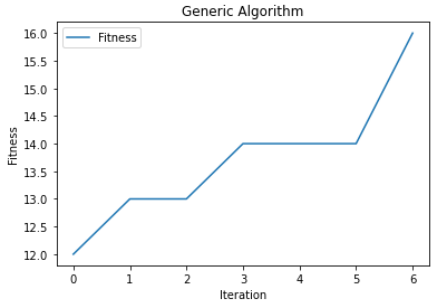


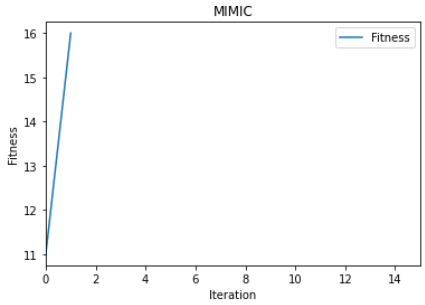


Fitness Score:

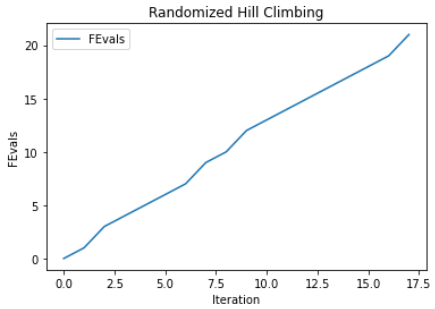


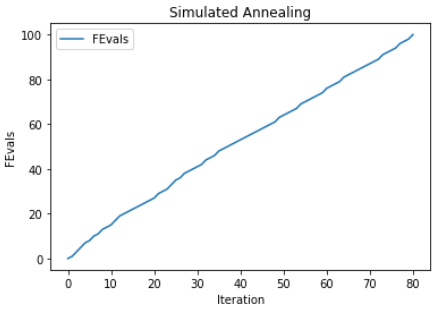


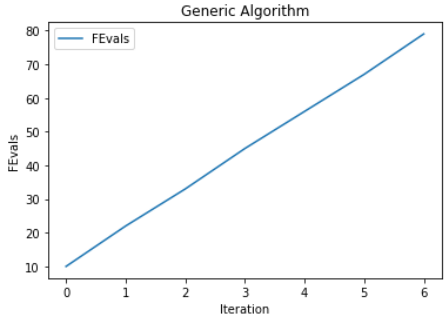


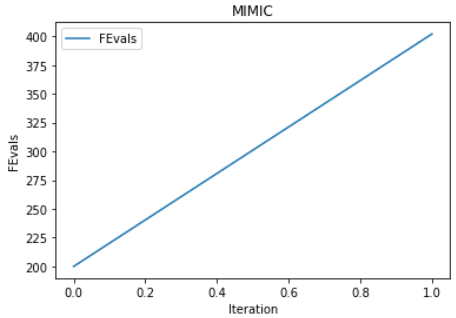


Number of Evaluations:









Summary:

Algorithm	Fitness	FEvals	Time	State
Randomized Hill Climbing	16	21	0.014627	[0, 1, 2, 0, 0, 2, 0, 1, 2, 1]
Simulated Annealing	16	100	0.027435	[1, 2, 0, 1, 0, 0, 1, 2, 0, 2]
Genetic Algorithm	16	79	0.010374	[2, 1, 0, 2, 0, 1, 2, 1, 0, 1]
MIMIC	16	402	0.044397	[0, 1, 1, 2, 0, 1, 0, 2, 2, 1]

I think the best approach is MIMIC algorithm with population of 200 and percent of 0.25. These 4 final fitness scores are same and hit the maximum. But based on the fitness comparison graph, I think MIMIC has significant speed and works fast since it makes the fitness function hit the maximum in first iteration. Even though MIMIC's running time is not the least one, its running time is very close to others' running time. Maximum fitness is 16 because there are 16 pairs of nodes in the graph.

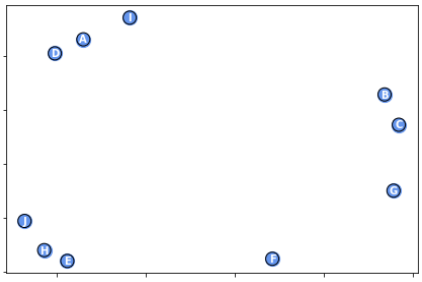
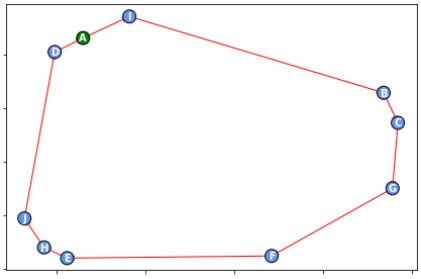
Optimization Problem 3: Travelling Salesperson Problem

Description:

Suppose a salesperson wants to visit multiple cities to sell the products. We want to design a trip for the salesperson so that he/she can visit all cities. It is required the salesperson should start and come back to the same city. In this problem, I assume there are 10 cities that the salesperson will visit.

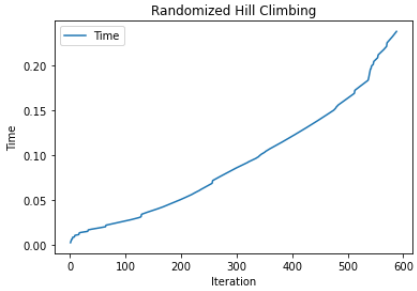
Why interesting?

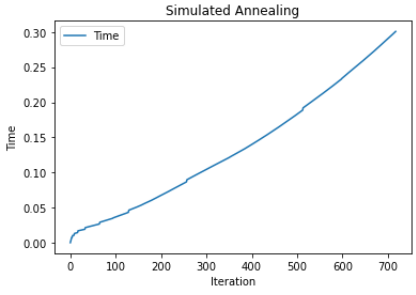
This is a very interesting problem because it is easy to compute the total length of the trip and we can use optimization algorithm to maximize the distance he can save if we design a good road trip.

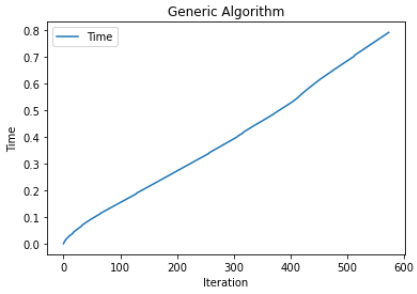

Initial--------->Optimization


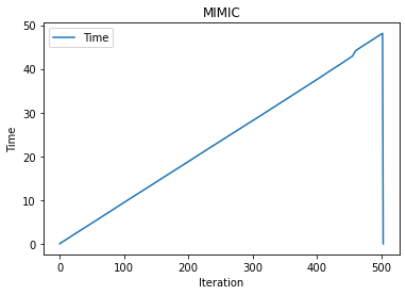
Comparison of Algorithms:

Running Time:

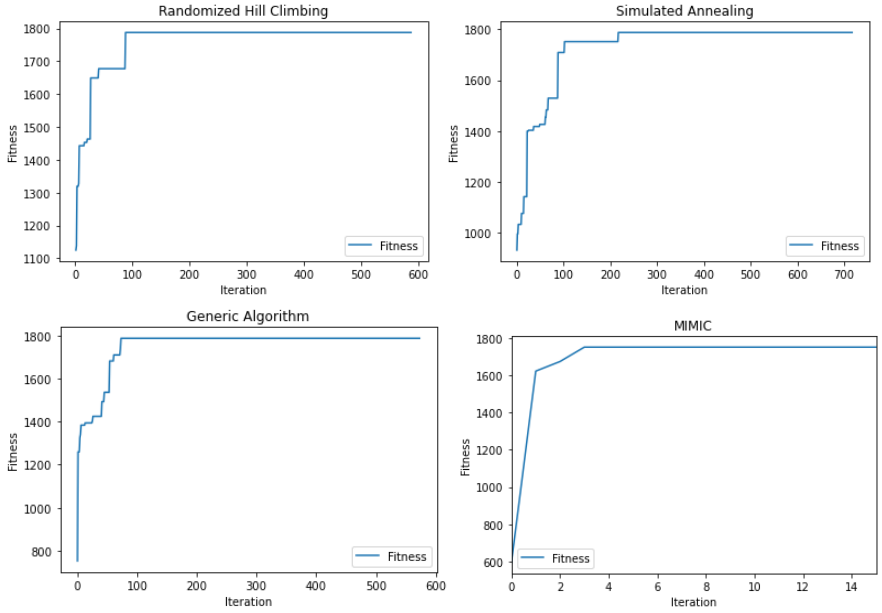




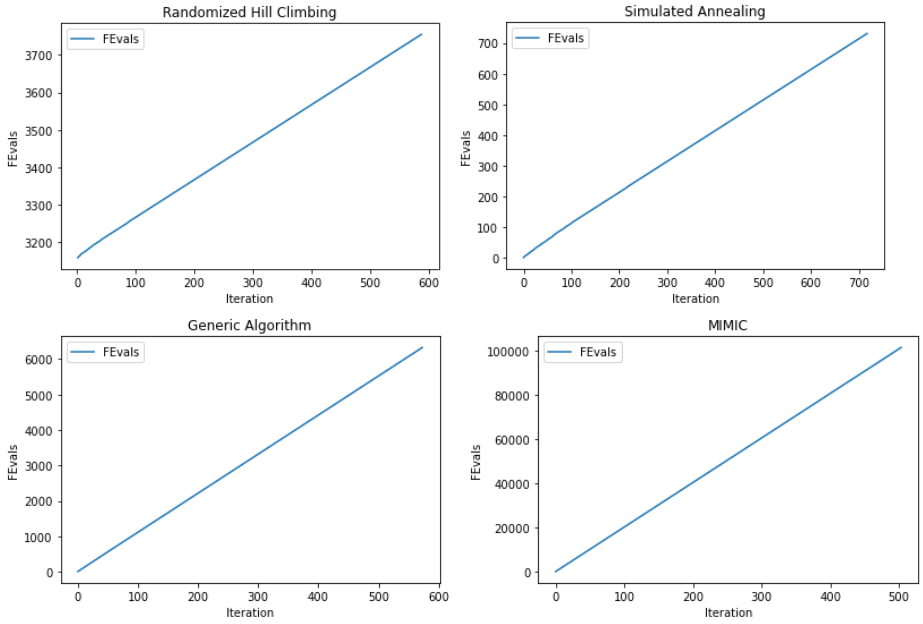




Fitness Score:



Number of Evaluations:



Summary:

Algorithm	Fitness	FEvals	Time	State
Randomized Hill Climbing	1787.530712	3755	5.429274	[9, 3, 0, 8, 1, 2, 6, 5, 4, 7]
Simulated Annealing	1787.530712	731	0.328342	[0, 3, 9, 7, 4, 5, 6, 2, 1, 8]
Genetic Algorithm	1787.530712	6326	0.771824	[7, 9, 3, 0, 8, 1, 2, 6, 5, 4]
MIMIC	1751.638743	101306	61.51314	[6, 5, 4, 7, 9, 0, 3, 8, 1, 2]

I think the best approach is simulated annealing algorithm with temperature of 0.1 because it uses the least time and least number of evaluations to get the maximum fitness scores although simulated algorithm uses a lot of iterations because it tries to explore and then exploit the path to the optima. I tried the temperature from 5 to 0.1. As we all

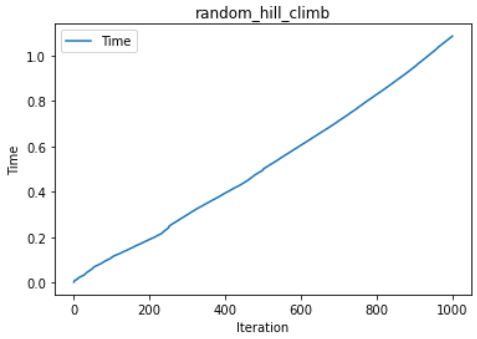
known, larger temperature will be like a random walk to do the exploration work. And smaller temperature will force the model to do the exploitation like hill climbing to find the optima.

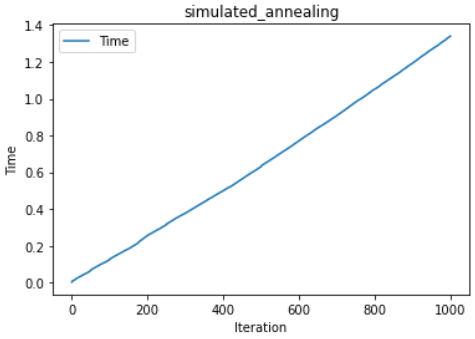
Neural Network Model with Optimization Algorithms

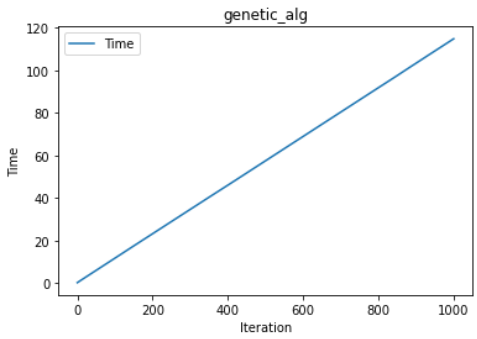
Dataset: Breast Cancer
Data (Breast_cancer_ta.csv) can be found in the link:
https://github.com/qingninglily/CS7641/tree/main/Randomized_Optimization

Comparison:

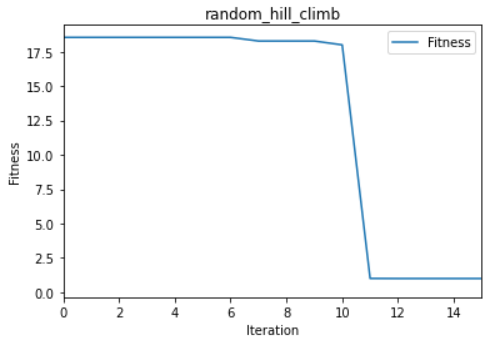
Running Time:

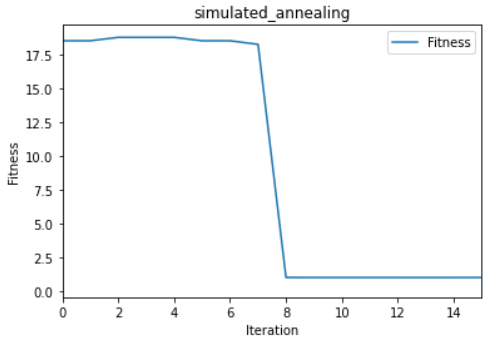


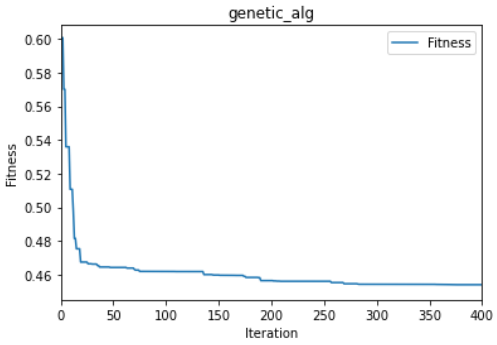




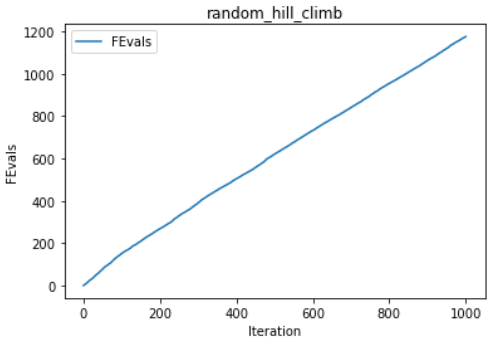
Fitness: (Lower fitness is better)

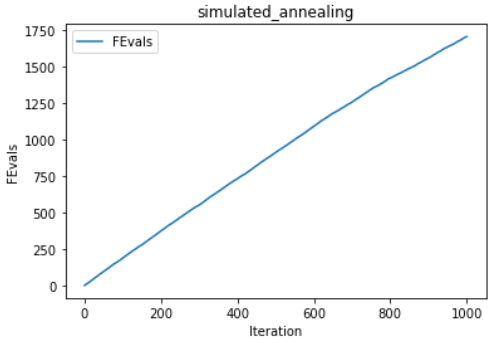


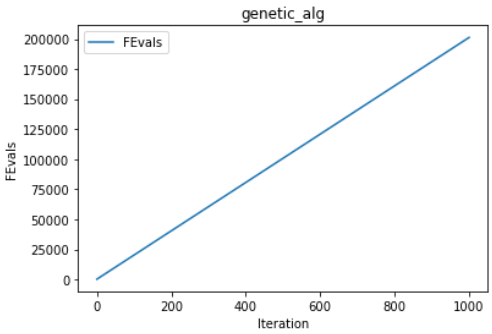




Number of Evaluations:







Summary:

Algorithm	Training Accuracy	Testing Accuracy	Precision	Recall	F1 score	AUC ROC	Learning Rate	Activation
Randomized Hill Climbing	0.373	0.368	0	0	0	0.5	0.001	Relu
Simulated Annealing	0.868	0.877	1	0.83	0.91	0.93	0.001	Relu
Generic Algorithm	0.874	0.885	0.95	0.87	0.91	0.94	0.001	Sigmoid

Algorithm	Fitness	FEvals	Time	activation	learning_rate
Randomized Hill Climbing	0.542501	1176	1.181279	relu	0.001
Simulated Annealing	0.468102	1705	2.053323	relu	0.001
Generic Algorithm	0.452513	201279	114.867163	sigmoid	0.001

Analysis:
For neural network model with randomized hill climbing algorithm, it performs worst compared to other algorithms. It uses 11 iterations to converge. The model's accuracy rate, precision, recall and AUC ROC are pretty low. I think this algorithm might get stuck in local optima so that it is hard to find global optima. To improve this model, I suggest using more random restarting points which will get a higher chance to hit the global optima. Also, we can try to let restarting points cover a wide range to make sure we did a good exploration.

For neural network model with simulated annealing algorithm, it performs best. The accuracy rate is 0.877 in testing dataset. Even though the fitness score is not the lowest one, the running time and number of evaluations are much less compared to the model with generic algorithm. It only takes 8 iterations to converge so this model works very fast. F1 score and AUC ROC are over 0.9 which are very high. To improve this model, we can play around the temperature parameter to balance the exploitation and exploration.

For neural network model with generic algorithm, it performs second best because it only takes few iterations to converge to a low fitness score. But one iteration takes very long time and uses a lot of evaluations. To improve the model, we can tune the mutation rate and population size to let the "parents" generate better "offspring" and apply mutation to add new features on offspring.

Reference

[1]Hiive. (2021, March 12). *Mlrose/problem_examples.ipynb at master · hiive/MLROSE*. GitHub. Retrieved October 18, 2021, from https://github.com/hiive/mlrose/blob/master/problem_examples.ipynb.

[2]Hayes, G. (2019, December 24). *Getting started with randomized optimization in python*. Medium. Retrieved October 18, 2021, from https://towardsdatascience.com/getting-started-with-randomized-optimization-in-python-f7df46babff0.