Assignment 2: Randomized Optimization

Haoran Yang – ID: hyang412 – Email: hyang412@gatech.edu

The algorithms used in this project:

In this project, we will explore following RO algorithms.

1. randomized hill climbing
2. simulated annealing
3. a genetic algorithm
4. MIMIC

All of the mentioned local random search algorithms will be implemented by python package mlrose.

Three optimization problem domains:

- n-queens
- 4-peaks
- flipflop

This assignment plans to apply all four search techniques to these three optimization problems. According to the problem structure and different performance, we try to illustrate relative strengths for each algorithm.

In the beginning, we should define how we compare four algorithms' performance for each problem.

The same part includes

1. max_attempts: maximum number of attempts to find a better state at each iteration
2. max_iters: maximum number of iterations

Some specific parameters for each algorithm:

1. RCH – number of random restarts [restarts]
2. SA – schedule used to determine the value of the temperature parameters [schedule]
3. GA – size of population to be used in genetic algorithm [pop_size], probability of a mutation in each element of the state vector during reproduction [mutation_prob]
4. MIMIC – size of population to be used in algorithm [pop_size], proportion of samples to keep at each iteration of the algorithm [keep_pct]

In this research, we decide to fix most parameters and check the mean of fitness score with maximum 1000 iterations in 1000 independent runs.

Table 1 Parameter Setting

Algorithm	max_attempts	max_iters	Others

RHC	10	1000	[restarts]: 10
SA	10	1000	[schedule]: exponentially decaying from 1.0 to 0.001
GA	10	1000	[pop_size]: 200; [mutation_prob]: 0.1
MIMIC	10	1000	[pop_size]: 200; [mutation_prob]: 0.2

1. N-queen

In chess, the queen can attack others in the same row, column, or diagonal. In n-queens problem, the aim is to place the queens on the n*n board so that none of them can attack each other.

Since any two queens should not in the same column in optimal situation, we can model this optimization problem with a n-element state vector $[x_0, x_1, \ldots\ldots, x_{n-1}]$ where each element x_i denotes the row position of the queen position in column i.

We will implement this 8-queen optimization problem in python.

For the fitness function, let's define it as evaluating the number of pairs of attacking queens for a given state and try to minimize this function.

The results after 1000 independent runs with different random initial states are listed in table 2,

Table 2 8-queen optimization results

Algorithm	Average Score
RHC	1.001
SA	0.003
GA	2.0
MIMIC	2.0

Since the objective function is minimizing, I saw SA and RHC perform much better than GA and MIMIC. SA got the best results, reaching the optimal results within 1000 iterations almost all the time.

2. 4-peak problems

A 4-peak optimization problem can be described as follow. Given parameter T, find a n-dimensional state vector x

$$Maximize\ f(x,T) = max(tail(0,x), head(1,x)) + R(x,T)$$

Where:

- $tail(b,x)$ is the number of trailing b's in x
- $head(b,x)$ is the number of leading b's in x
- $R(x,T) = n$, if $tail(0,x) > T$ and $head(1,x) > T$
- $R(x,T) = 0$, otherwise

We are running a 4-peak optimization on a 12-dimentional binary vector with T as 2. The optimal fitness score should be 21.

The results after 1000 independent runs with different random initial states are listed in table 3,

Table 3 4-peak optimization results

Algorithm	Average Score
RHC	8.013
SA	20.001
GA	21
MIMIC	21

Different with the results of 8-queen optimization, GA and MIMIC perform much better here.

3. Flipflop Problem

This problem domain is to find the n-size binary bit string x, maximizing the total number of pairs of consecutive elements.

Take following table as an example:

Table 4 flipflop fitness evaluation example

Vector	Fitness Score
(1,0,1,0,1,0)	5
(1,1,1,1,1,1)	1
(1,1,1,0,0,0)	2
(0,0,1,0,1,1)	3
……	……

We are running a flipflop optimization on a 30-dimentional binary vector. The optimal fitness score should be 29.

The results after 1000 independent runs with different random initial states are listed in table 5,

Table 5 flipflop optimization result

Algorithm	Average Score
RHC	24.997
SA	26.0
GA	24.0
MIMIC	24.0

Summary

According the 3 selected optimization domains, we can see that each algorithm has its own advantages and disadvantages. RHC and SA works well in 8-queen problem, which have more

reachable global optimal. For 4-peak problem, it has 2 global optimal points and 2 local optimal points, models with more structure information such as GA and MIMIC work better.

Optimizing Neural Network Weights by RO

Combined with what we have learnt in SL part, training a ML model sometimes involves finding the weights that minimize a pre-specified loss function for a given training dataset. Some examples include neural networks, linear regression models and logistic regression models. Previously, we usually rely on gradient descent to calculate weights.

However, this training process can also be viewed as a continuous-state optimization problem, where the loss function takes the role of the fitness function, and the goal is to minimize this function.

In this assignment, we will use mlrose to fit the model by both gradient descent and selected RO algorithms. Since the problem is a continuous-state one, we will not apply MIMIC here.

Consist with assignment 1, the Iris dataset will be used to build a single-layer neural network with 100 nodes.

We set the max_attempts as 100 and max_iters as 10000 since a machine learning weights optimization problem is more complex than the problems we selected in previous part.

Table 6 neural network weights optimization results

Algorithm	Training Accuracy	Test Accuracy
RHC	0.34	0.40
SA	0.33	0.36
GA	0.89	0.93
Gradient Descent	0.98	1.00

According to the results, applying randomized optimization algorithms to the machine learning weight optimization problem is most certainly not the most common approach to solving this problem. However, it serves to demonstrate the versatility of randomized optimization algorithms in general.