Randomized Optimization

Younes EL BOUZEKRAOUI
ybouzekraoui3@gatech.edu

***Abstract*—**The Purpose of this project is to compare the performance of four randomized optimization methods (randomized hill climbing, Simulated annealing, Genetic algorithm, MIMIC) in the context of three optimization problems intended to highlight the strengths and weaknesses of each algorithms in different situations.
The randomized hill climbing, Simulated annealing, and Genetic algorithms are then applied to a deep learning problem : finding optimal weights for a neural network.

1 TOOLS AND ENVIRONMENT

The programming language used for this assignment is Python with the Numpy, Matplotlib, Pandas and Sckit learn libraries. In Addition to the MLrose Library which is a Python package that allows using some of the most popular randomized optimization and search methods to solve a variety of optimization problems in both discrete and continuous parameter spaces.

The programming environment is Jupyter Notebook, and the code is available in a jupyter-notebook format and also in pdf format.

2 LOCAL RANDOM SEARCH ALGORITHMS

In This Assignment 4 different algorithms for random optimization are put in place and tested on 3 different optimization problems in order to evaluate the performance of each algorithm in different situations and show their strengths and weaknesses .
Firstly, let's define briefly each one of the algorithms.

2.1 Randomized hill climbing

This algorithm chooses a beginning point at random and then starts moving in the direction of increased fitness with each iteration. This isn't particularly useful because it's so reliant on the beginning point: it's simple to get stuck in a local optimum. We can, however, repeat the algorithm many times to ensure that we don't constantly get stuck in the same one; even if we don't discover the global

optimum, we'll at least get a really good local optimum. Hill climbing with a random restart is known as random-restart hill climbing.

2.2 Simulated annealing

Simulated Annealing is a global search optimization technique with a stochastic component. This means that it uses randomness as part of the search procedure. Unlike the hill climbing algorithm,were we always exploit the best possible direction, now we'll do some additional exploration. it explores the neighborhood and it may accept worse solutions as the current working solution and move on its direction. This helps avoiding getting stuck in local optimums.

2.3 A genetic algorithm

Genetic algorithms are a type of learning algorithm that is inspired by simulated evolution. A population, or collection, of starting hypotheses is used to find the optimum. Random mutation and crossover, which are modeled by biological evolution processes, are used to give rise to the next generation population by members of the present population.

The current population's hypotheses are compared to a specified measure of fitness at each stage, with the most fit hypotheses being chosen problematically.

2.4 MIMIC

Because they always terminate at one point, the algorithms we've looked at so far: hill climbing, simulated annealing, and genetic algorithms all seem relatively primitive. They learn nothing about the space they're exploring, they have no knowledge of where they've been, and they have no memory of the underlying probability distribution they're searching.

The primary idea underlying MIMIC is that it should be possible to directly model a probability distribution, improve and refine it over time, and end up with something that will keep the knowledge of the past iterations

3 OPTIMIZATION PROBLEMS

In order to evaluate and test the performance of the previous algorithms 3 problems are designed to show the strengths and weaknesses of each one of them.

3.1 Problem 1 : Count Ones or One Max

This optimization problem seems simple but it is a good example to show the strengths of the Randomized hill climbing and Simulated annealing methods.

In this problem, given a Bit strings of length n

$$x = [x_0, x_1, \ldots, x_{n-1}]$$

we are trying to maximize the number of ones in this bits string, the x_i can only take the values 0 or 1.

Mathematically, the fitness function for One Max optimization problem.

Evaluates the fitness of an n-dimensional state vector

$$x = [x_0, x_1, \ldots, x_{n-1}]$$

as:

$$\mathtt{Fitness}(x) = \sum_{i=0}^{n-1} x_i$$

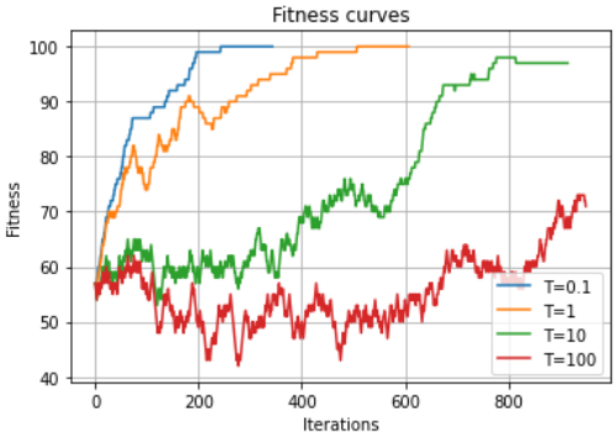
In this context the solution of this problem is the bit string containing only ones [1,1,1,1...,1], and this is the only global maximum, moreover this problem does not contain any local maximums.

Under all these circumstances, the problem seems to be more suitable to be solved with the Randomized hill climbing and Simulated annealing methods because they are relatively fast, inexpensive and because there are no local maximum and only one single global maximum with a large basin of attraction.

This is what we will confirm by testing all the algorithms with different parameters.

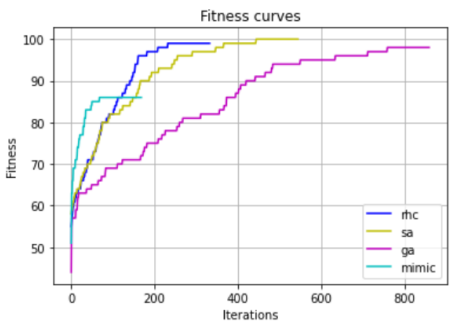
3.1.1 *Experiment Results*

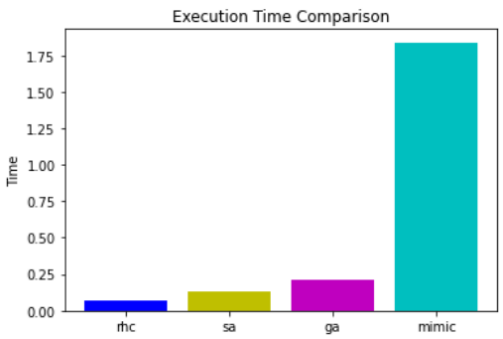
To evaluate the performance of the Simulated annealing algorithms , we tried different values of the temperature to see the number of iterations needed to reach the maximum



As we can see for higher temperatures the algorithm is taking too many iterations to reach the maximum , this is due to the fact that for the higher temperature the algorithm behaves like a random walk and try to explore the neighborhood but for lower temperature it behaves like the standard hill climbing without exploration. Because we only have one global maximum, with lower temperatures, the algorithms will go straightforward to the maximum in less iterations compared to higher temperatures.

In order to compare the performance of all the algorithms (rhc = Random Hill Climbing, sa = Simulated annealing, ga = Genetic algorithms, mimic) we plotted their fitness curves with respect to the number of iterations and also their performance in term of execution time. We used the best temperature for the Simulated annealing algorithm (T=0.1)





As we can see in the previous figures, the Random Hill Climbing and the Simulated annealing algorithms performed well in this case, they reached the maximum (fitness =100) in less iterations than the others, and also in terms of time

performance the Random Hill Climbing and the Simulated annealing are performing better with less time compared to the others.

We can say the for this type of simple optimisation problems with no local maximum it is better to use the Random Hill Climbing or the Simulated annealing algorithms because they are more suitable to this type of problems , and even with local maximum but with small basin of attraction the Simulated annealing method is expected to perform well by tuning the Temperature parameter to avoid getting stuck in local maximum

3.2 Problem 2 : Four Peaks

The second optimization problem , given a bit string x of length N and a threshold T, the fitness function is

$$\begin{aligned} z(x) &= \text{Number of contiguous Zeros ending in Position 100} \\ o(x) &= \text{Number of contiguous Ones starting in Position 1} \end{aligned}$$

$$REWARD = \begin{cases} 100 & \text{if } o(x) > T \wedge z(x) > T \\ 0 & \text{else} \end{cases}$$

$$f(x) = MAX(o(x), z(x)) + REWARD$$

The expression of the fitness function above is for an example of N =100.

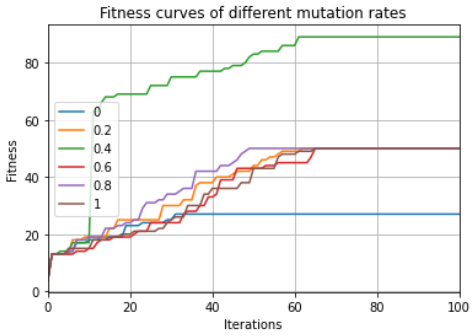
If we take the example of N = 50 , the bit strings contains 50 bits, and the Fitness function reaches it's maximum if a string is able to get both the REWARD of 50 and if one of O(X) or Z(X) is as large as possible.

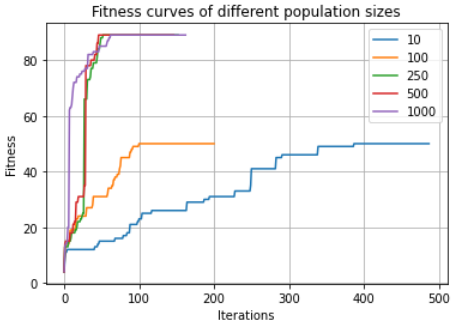
As an example, for T = 10 The optimal fitness of 89 is obtained by bit strings containing either thirty-nine 1's followed by eleven 0's or eleven 1's followed by thirty-nine 0's. These are the two global maximum.

There exist also two local maximum with strings of all 0's and all 1's with a fitness function of 50. These local maximum have large basins of attractions so we expect the genetic algorithms and the MIMIC to excel here since the Randomized hill climbing and Simulated annealing methods are more likely to get stuck on the local maximums.

3.2.1 *Experiment Results*

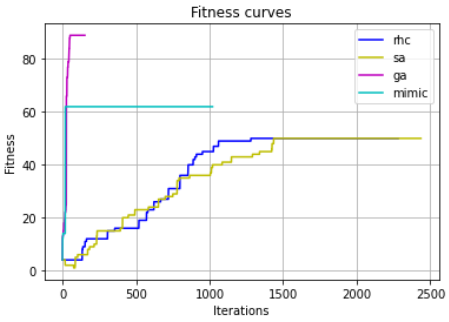
To evaluate the performance of the Genetic algorithms , we tried different values of the mutation rate and the population size to see how they affect the results

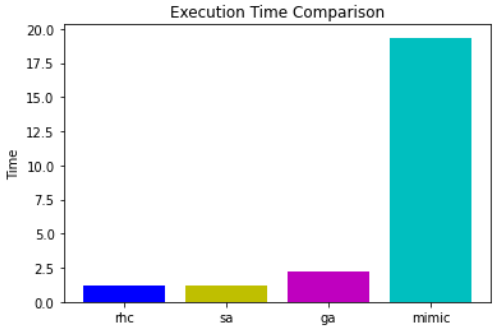




As we can see on the figure on the left, with a fixed population size of 200 mutation we tried different values of mutation rates and with a value of 0.4 the algorithm was able to avoid the local maximum and reach the global maximum.

On the figure on the right , for a mutation rate of 0.4 we tried multiple values of population size, and we can see that for lower population sizes (10 and 100) the algorithms got stuck on the local maximum (50) but for big population he was able to reach the global maximum (89), this makes sense, because with bigger populations we will have more cross overs and mutations and the algorithm is more likely to avoid the local maximum and converge quickly to the solution





By comparing the performance of the genetic algorithm (using it's better parameters mutation-rate = and population-size = 250) we can see that it is performing very well compared to the others, the global maximum is reached after less than 200 iterations, this was expected because there exist local maximums that will

catch the other algorithms and because the genetic algorithms are performing parallel hill climbing thanks to the mutations and cross over they were able to out-perform the other and avoid getting stuck on the local maximums

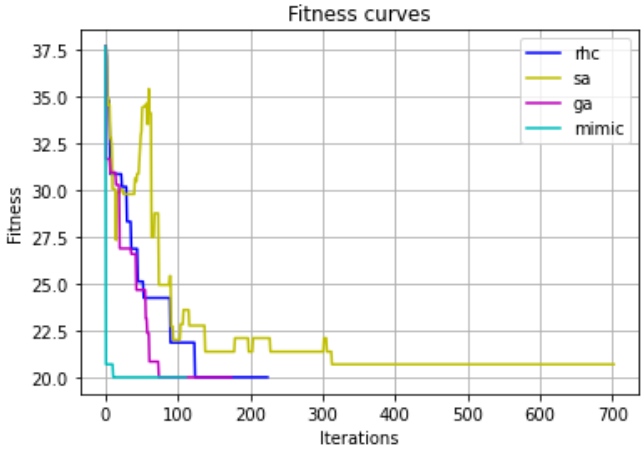
From the time perspective it's performing better than MIMIC and a little less than RHS and SA, but in general the execution time is reasonable .

3.3 Problem 3: The travelling salesman problem (TSP)

The travelling salesman problem (TSP) is a classic optimization problem in which the goal is to find the shortest tour of a set of cities that starts and ends in the same city and visits all of the other cities exactly once.

This is considered as an NP-hard combinatorial optimization problem and we expect that MIMIC algorithm excel in this type of problem because it's the only algorithm that keeps track of the information from the past iterations.

3.3.1 *Experiment Results*

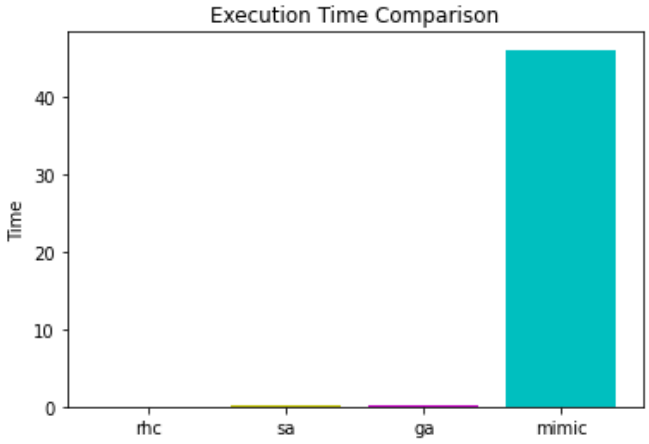


The MIMIC algorithm is doing well in this optimization problem after tuning it's parameters (population size of 1000 and 'keep percentage' of 0.5).

Note that this is a travelling salesman problem (TSP) with 10 cities which is relatively small, and we can see that MIMIC is performing well compared to the others.

I wanted to test the algorithms on bigger dimension problem 100 cities but i realised that using my personal laptop it will take a very long time.

Normally in a more complex problems (100 cities) the MIMIC is expected to show a great out-performance in comparison to the others for the same reason which is that it keep information from the past unlike the others .



In term of time MIMIC algorithm is taking a very long time compared to others, and in very complex problems (many inputs) this may be a problem if we didn't prepare and optimize the infrastructure for the algorithm to work in (Parallel computing, powerful computers). But at least we can say that it's guaranteed with high probability to perform better than the previous algorithms and finally reach the optimum even after a long time.

4 NEURAL NETWORK WEIGHT OPTIMIZATION

In this part we tried to optimize the weights of a neural network used in a previous supervised learning assignment about Classification the data-set is about churn prediction for a Telecommunication company, this is a binary classification problem.

The first step is preparing the data set (one-hot encoding, label encoding ...) and then fitting the model, and finally evaluating it's performance using cross validation and testing on unseen data

In order to evaluate the performance of randomized optimization in this type of problem , we tried to do the fitting, which the optimization of the weight using the methods (randomized hill climbing, Simulated annealing, Genetic algorithm, MIMIC) and comparing them to the default gradient decent method.

The neural network contains 3 hidden layers of 64 nodes each with the relu activation function. We need to know that here we are dealing with a continuous optimisation problem since the weights are floats contrarily to the previous optimization problems .

4.1 Experiment Results

After fitting the neural network using the different algorithms we got the following results for the testing and training accuracy :

```
---------- Gradient Descent :
training data Accuracy: 0.854
Testing data Accuracy: 0.858
```

```
---------- Random Hill Climb :
training data Accuracy: 0.854
Testing data Accuracy: 0.142
```

```
---------- Simulated annealing :
training data Accuracy: 0.146
Testing data Accuracy: 0.142
```

```
---------- Genetic algorithms :
training data Accuracy: 0.869
Testing data Accuracy: 0.877
```

As we can see Random Hill Climb and Simulated Annealing did not perform well in the optimisation problem, and we got bad accuracy in both for the testing data and even bad accuracy for the SA in the training accuracy. These results are taken even after tuning and trying multiple parameters for the algorithms (population size, restarts)

However the genetic algorithms did a very good work in this optimization problem , we even got a training and testing accuracy better than gradient decent.

This may be explained by the possibility of presence of a local optimum with large basin that keeps getting the Random Hill Climb and Simulated Annealing algorithms stuck in each time even with multiple restarts.

But for the Genetic algorithms it was able to avoid the local maximum thanks to it's propriety of parallel search due to mutations and cross over.

5 CONCLUSION

In conclusion we can say that randomized optimization is really useful in certain problems where it's nearly impossible to apply classic methods of optimization (gradient descent) and where we cannot evaluate all the possibilities to get the optimum especially Discrete complex problems or when we are dealing with complex functions with massive input spaces and many local optimums.

" People believe the only alternative to randomness is intelligent design "
–Richard Dawkins–