Machine Learning
-
Supervised Learning

Brandon Alves

October 16, 2022

Contents

1 Introduction

Optimisation is the goal of finding the best state over an input space, with respect to a fitness function. In this article, I will explore four different Randomised Optimisers on three different problems. I will be using the following optimisers:

- Random Hill Climbing
- Simulated Annealing
- Genetic Algorithm
- MIMIC

I will discuss their performance and compare them to each other. I will also discuss the pros and cons of each optimiser depending on the problem. In that purpose, I will explore multiple meta-parameters for each optimiser. I will use those optimisers on the following problems:

- Traveling Salesman
- Flip Flop
- Four Peaks

I will also use the three first optimisers to find good weights for a neural network on the gender dataset. The graphs presented in this report follow the same format as the one in the previous report. The hard line represents the median value while the hue areas fill the space between the minimal and maximal values.

2 Random Optimisation

2.1 Traveling Salesman Problem

Introduction

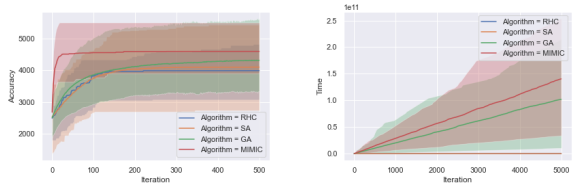
The Traveling Salesman Problem (TSP) is a well-known problem in the field of optimisation. It consists in finding the shortest path that visits all the cities of a given graph. The TSP is a NP-hard problem, which means that it is difficult to solve. In this section, we will present the results of the optimisers on the TSP problem. For this problem, the fitness function to minimise is the length of the path:

$$f(d) = \frac{1}{d} \tag{1}$$

with d the length of the path. We present on figure 1 the results of the four different optimisers on the TSP problem. The first thing to notice is that there is a lot of variation for each algorithm. RHC and SA have an median accuracy which are pretty close. GA does a little bit better while MIMIC gets the best median result. We can also notice that MIMIC gets the best results with a low number of iteration and reaches its bets optimisation only after a few iterations.

Randomised Hill Climbing

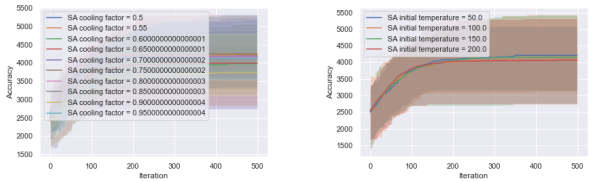
The only parameter we will discuss here for RHC is the number of iterations. We see that the number of iterations has an impact on the performance of the algorithm. The best results are obtained with a high number of iterations but RHC does not success to get a good accuracy even with a high number of iterations as we can see on



(a) Accuracy on TSP Problem for the different algorithms

(b) Time on TSP Problem for the different algorithms

Figure 1: Accuracy and Time on TSP Problem for the different algorithms



(a) Accuracy on TSP with SA, for different cooling factors

(b) Accuracy on TSP with SA, for different initial temperatures

Figure 2: Accuracy of SA on TSP according to Initial Temperature and Cooling Factor

figure 1a. It is not surprising that RHC does not get good results on this problem because it is a local optimiser. It will not be able to find the best solution because it will get stuck most of the time in a local optimum. The only way it can succeed is to start in the basin of attraction of one global optimum.

RHC is the fastest algorithm of the four as we can see on figure 1b. It is not surprising because it is a local optimiser and it does not explore the whole search space. It only exploit the neighbourhood of the current state.

To conclude, RHC is not well suited for this problem even if we appreciate its speed.

Simulated Annealing

SA gives the second worst median results on this problem. For SA, we will discuss the following parameters:

- Temperature
- Cooling Rate

We can see the similarity between the results of RHC and SA. Which seems faire because SA is just an improved way of doing RHC. Another interesting thing to notice is that SA gives the best and worst results in maximum and minimum values respectively. This can be explained by the space exploration of SA where RHC only exploit the neighbourhood of the current state. Let's discuss the impact of the two parameters of SA.

We can see the impact of the cooling factor on the accuracy of the algorithm on figure 2a. We notice that the cooling factor does not seem to have a big influence. However it is interesting to notice that the minimum and maximum values are affected. We can can see that a high cooling factor gives smaller minimal accuracy, whereas a low value gives smaller differences between min and max. This can be explained by the fact that the bigger the cooling factor is, the more the temperature will decrease and the less the algorithm will explore the search space, and finally SA will tend to get stuck in a local optimum like for RHC. In the other hand, a low cooling factor will allow SA to explore more the search space. This is why we can see that the minimum accuracy is better with a low cooling factor.

We can see the impact of the initial temperature on the accuracy of the algorithm on figure 2b. Like for the cooling factor, the initial temperature does not impact a lot the accuracy of the algorithm. The best results are obtained with a low initial temperature of 50. The same comments as for the cooling factor can be applied here.

SA is the second fastest algorithm of the four as we can see on figure 1b. It is not surprising because of its similarity with RHC.

To conclude, as RHC, SA is not really well suited for this problem.

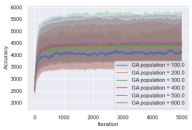
Genetic Algorithm

GA gives the second best median results on this problem. Figure 3 shows the impact of the following parameters:

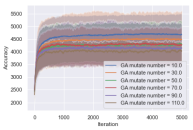
- Population Size
- Mutate Number
- Mate Number

Let's start with the population size. We can see on figure 3a that the accuracy is the highest for high values of population size while it is the lowest for low values of population size. We can see that the best results are obtained with a population size of 500. The fact that the accuracy depends on the population size is not surprising because the population size is the number of individuals in the population. The more individuals we have, the more chances we will have to combine them to find a good solution.

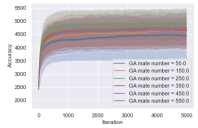
Let's now discuss the impact of the mutate number. We can see on figure 3b that the accuracy is the highest for low values of mutate number while it is the lowest for high values of mutate number. We can see that the best results are obtained with a mutate number of 10. The



(a) Accuracy on TSP with GA, for different population size



(b) Accuracy on TSP with GA, for different mutate number, with population set to 600



(c) Accuracy on TSP with GA, for different mate number, with population set to 600 and mutate number to 10

Figure 3: Accuracy of GA on TSP according to Population Size, Mutate Number and Mate Number

mutation is important in GA as it permits to introduce new variations that were not present in past iterations. The problem is that introducing too much variations can lead to turn a good predictor into a poor one before it has a chance to reproduce.

On figure 3c we can see the impact of the mate number on the optimisation accuracy for a mutate number of 10. On this graphic, we can see that the best median accuracy is given with a mate number of 250. The next best results are given with the highest value of the mate numbers. It is interesting to see that, in TSP, a performant GA is one with few mutations but a lot of mating to try to combine the good solutions. When having few mutations and mating, the population is very stable and only the best estimators are used in the mate process giving us a very slow way to get a result but which is more likely to end up with good results. So we can see that they are here a sort of trade off between trying to get the best result by having a population evolving very quickly in the neighbors of the best solution or one slowly exploring those neighbors and keeping track of the previous generations, kept unmodified.

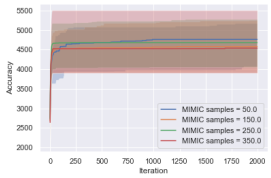
GA is the second slowest algorithm of the four as we can see on figure 1b.

Finally, GA did not give slightly better results than SA or RHC but is much more time consuming.

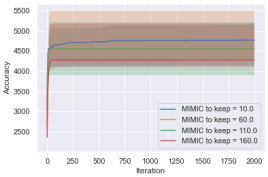
MIMIC

MIMIC gives the best median results on this problem. Figure 4 shows the impact of the following parameters:

- Number of samples generated for each iteration
- Number of samples to keep from an iteration to another



(a) Accuracy on TSP with MIMIC, for different number of samples



(b) Accuracy on TSP with MIMIC, for different number of estimators kept, with a number of samples to 350

Figure 4: Accuracy of MIMIC on TSP according to the Number of Samples and the Number of Estimators Kept

Let's discuss the impact of the number of samples generated for each iteration. We can see on figure 4a that the accuracy is the highest for a number of samples of 50 and the lowest for a number of samples of 350 while a number of samples of 250 gives better results than a number of 150. That result is a little bit surprising as we would have expected the best results to be given with a number of samples of 350.

On figure 4b we can see the impact of the number of samples to keep from an iteration to another with a number of samples of 350. I decided to use a high number of samples at each iteration to make the different number of samples to keep more significants. We can see that the best median accuracy is given with a number of samples to keep of 10. The worse accuracy is given for a number of sample to keep of 160. It is interesting to see that, in TSP, a performant MIMIC is one with few samples generated for each iteration as well as a few samples kept from an iteration to another to try to combine the good solutions.

In terms of computation time, MIMIC gives the worst results compared to the other algorithms.

Finally, MIMIC gave the best results on this problem at the cost of a long computation time.

2.2 Flip Flop Problem

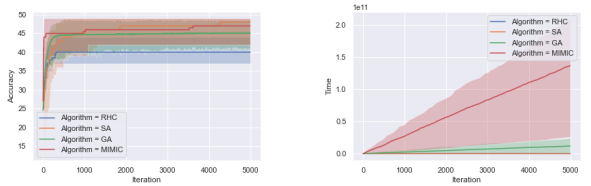
Introduction

The Flip Flop problem is a problem where we have a binary vector of size n and we want to find the vector that

has the most consecutive bit alternations. The fitness function is the number of consecutive bit alternations in the vector.

We present on figure 5 the accuracy and the time for the different algorithms on this problem. We can see that the best median accuracy is given by SA followed by MIMC. We chose a vector of 50 bit which mean that the best possible score is 49. SA and MIMIC manage sometimes to get that maximal score. GA stops at 48 while RHC does not exceed 43.

To be more concise, I will not restate the comments already expressed in the last subsection. Instead I will concentrate myself on what differs.



(a) Accuracy on Flip Flop problem for the different algorithms

(b) Time on Flip Flop problem for the different algorithms

Figure 5: Accuracy and Time on Flip Flop Problem for the different algorithms

Randomised Hill Climbing

Again RHC does not success to perform well. We can see on figure 5a that the best accuracy is given by a number of iterations bigger than 1000. But RHC does not succeed to improve even with much more iteration. It always get blocked on local optimums.

In terms of computation time, RHC is the fastest algorithm of the four as we can see on figure 5b. Again that is explained by the simplicity of the algorithm.

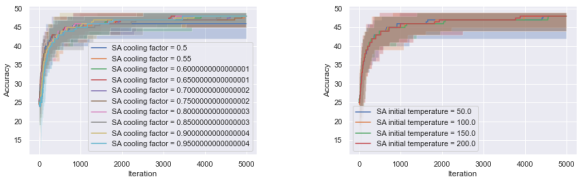
Simulated Annealing

In this problem, SA gives the best results. Figure 6 shows the impact of the following parameters:

- Number of iterations
- Cooling rate

We can see on figure 6a that the best accuracy is given by a cooling factor of 0.95. The more the cooling factor is high, the more the accuracy seems to increase. That means that this problem works well with a temperature decreasing quickly.

As we can wee on figure 6b, the initial temperature does not really affect the median accuracy of the SA optimizer.



(a) Accuracy on Flip Flop problem with SA, for different Cooling Factors

(b) Accuracy on Flip Flop problem with SA, for different Initial Temperature

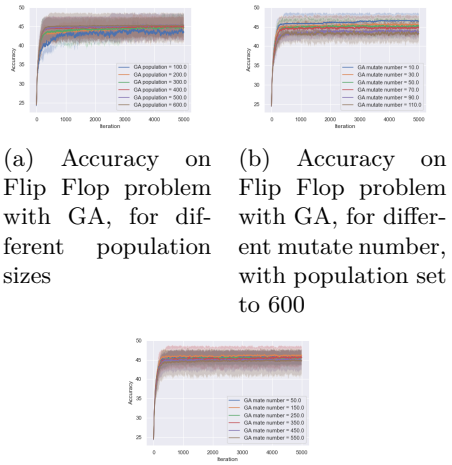
Figure 6: Accuracy of SA on Flip Flop according to Initial Temperature and Cooling Factor

In terms of computation time, SA is the fastest algorithm of the four, with RHC, as we can see on figure 5b.

To conclude, SA gives the best results on this problem with a fast computation time. It is perfectly suited for this problem. It is interesting to see the differencies of results between RHC and SA. Just by introducing some exploration as SA does, we can see that the results are much better.

Genetic Algorithm

GA gives the second best results on this problem after



(a) Accuracy on Flip Flop problem with GA, for different population sizes

(b) Accuracy on Flip Flop problem with GA, for different mutate number, with population set to 600

(c) Accuracy on Flip Flop problem with GA, for different mate number, with population set to 600

Figure 7: Accuracy of GA on Flip Flop according to Population Size, Mutate Number and Mate Number

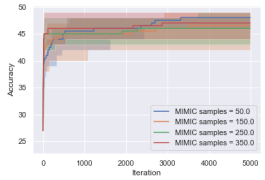
SA and MIMIC. We can see on figure 7 the impact of the different parameters. The best accuracy is given by

a population size of 600. The best accuracy is given by a mutate number of 10 and the worst by a mutate number of 110. Like in the previous problem, the optimiser works well with a small mutate number. Finally the figure 7c shows that the best accuracy is given by a mate number of 250. For that parameter it does not seem to have a linear impact on the accuracy. We can explain that by the fact that mixing two strings together in the crossover process does not really make sens as the flip-flop string must be synchronized to work properly. So finally the best optimisation for that problem is given by a relatively high population size (600), a small mutate number (10) and a mate number of 250.

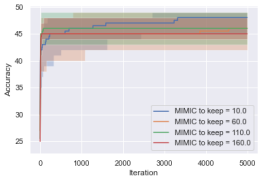
In terms of computation time, GA do not perform very well. It is the slowest algorithm after MIMIC, as we can see on figure 5b.

MIMIC

MIMIC gives one of the best results on this problem with SA



(a) Accuracy on Flip Flop problem with MIMIC, for different number of samples



(b) Accuracy on Flip Flop problem with MIMIC, for different number of estimators kept, with a number of samples set to 250

Figure 8: Accuracy of MIMIC on Flip Flop according to Number of Samples and Number of Estimators Kept

We can see on figure 8a that the best accuracy is given by a number of samples of 50. On figure 8b we can see that the best accuracy is given by a number of estimators kept of 10. As for GA, MIMIC do not take advantage of his complexity to find good solutions quickly. This probably comes from the fact that setting one bit to a particular value is not good or bad, without knowing the value of its neighbor.

In terms of computation time, MIMIC is again the slowest algorithm of the four, as we can see on figure 5b.

2.3 Four Peaks Problem

Introduction

The Four Peaks problem is a problem with two local optima and two global optima. The local optima have wide basins of attraction while the global optima have narrow basins of attraction. The goal is to find the global optima. The function to optimise is defined as follows:

$$f(x,t) = tail(0,x) \uparrow (head(1,x) + r(x,t)) \tag{2}$$

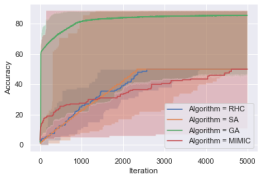
Where:

- x is the input vector, $x \in \{0,1\}^n$
- t is a threshold
- $head(k,x)$ is the number of leading k's in x
- $tail(k,x)$ is the number of trailing k's in x

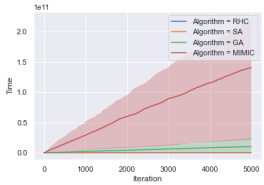
and:

$$r(x,t) = \begin{cases} n & \text{if } tail(0,x) \geq t \wedge head(1,x) > t \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

We present on figure 9 the results of the four algorithms on this problem. We can see that the best accuracy is given by GA. GA is the only algorithm to manage to find the global optima. The other algorithms get stuck in a local optima.



(a) Accuracy on Four Peaks problem for the different algorithms



(b) Time on Four Peaks problem for the different algorithms

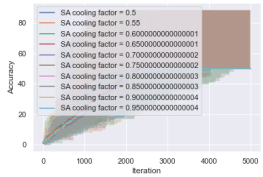
Figure 9: Accuracy and Time on Four Peaks Problem for the different algorithms

Randomised Hill Climbing

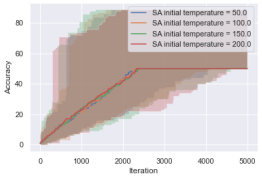
RHC goes not perform well on this problem. We can see on figure 9a that it only finds the local optima. That can be explained by the fact that the two basin of attraction of the two suboptimal peaks are much bigger than the ones of the optimal peak.

Simulated Annealing

SA does not do much better that RHC. It sometimes manage to find the global optimum, but fails the most of the time. On figure 10, we can see that the colling factor as well as the initial temperature do not have a real impact on the accurace of SA. Unfortunately, even if SA introduce exploration compared to RHC, it does not manage to find the global optima most of the time. We will see that the crossover introduce by GA will give us better results.



(a) Accuracy on Four Peaks problem with SA, for different Cooling Factors



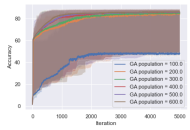
(b) Accuracy on Four Peaks problem with SA, for different Initial temperatures

Figure 10: Accuracy of SA on Four Peaks according to Cooling Factor and Initial Temperature

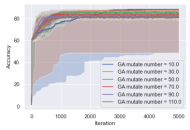
Genetic Algorithm

GA is the only algorithm to manage to find the global optima. Figure 11 shows the impact of the following parameters on the accuracy of GA:

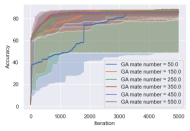
- Population Size
- Mutate Number
- Mate Number



(a) Accuracy on Four Peaks problem with GA, for different population sizes



(b) Accuracy on Four Peaks problem with GA, for different mutate numbers, with population set to 600



(c) Accuracy on Four Peaks problem with GA, for different mate numbers, with population set to 600

Figure 11: Accuracy of GA on Four Peaks according to Population Size, Mutate Number and Mate Number

We can see on figure 11a that the accuracy is heavely impacted by the population size. With a population size of 100, GA does not manage to find the global optima. With a population size above 200, it manages. But there is no real improvement with the different population sizes above 200. We can explain the fact that a small population size does not get good results by the fact that the population is not big enough to explore the space of solution and quickly get stuck on local optima.

Let's see the importance of the mutate number. We present the results on figure 11b with a population size of 600. We can see as before that the best results are obtained with a small mutate number. We can explain this by the fact that a small mutate number will introduce more exploration in the population and will allow the population to get out of local optima.

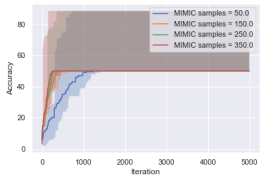
Finally, let's see the importance of the mate number. We present the results on figure 11c with a population size of 600. We can see that the mate number only impact the speed of convergence.

In terms of time, we can see on figure 9b that GA is the slower than RHC and SA as always.

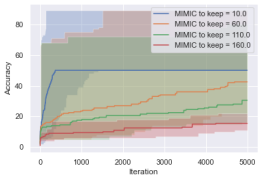
In conclusion, GA is really well suited for this problem. It is the only algorithm to manage to find the global optima.

MIMIC

MIMIC does not have better median result than RHC or SA. We can see the results of MIMIC on figure 5. In this problem, The number of sample does not seem to have an impact on the accuracy of the optimiser. On the other side, the number of samples to keep plays a role. We can see the smallest one gives the better results while the larger one gives the worst results. We can explain this by the fact that the larger the number of samples to keep, the more the algorithm will be biased towards the best samples of the last iteration. In terms of computation time, MIMIC is the slowest algorithm of the four. Which makes it the worse algorithm for this problem.



(a) Accuracy on Four Peaks problem with MIMIC, for different number of samples



(b) Accuracy on Four Peaks problem with MIMIC, for different number of estimator kept

Figure 12: Accuracy of MIMIC on Four Peaks according to Number of Samples and Number of Estimators Kept

Conclusion

To conclude we saw the behaviour of the different algorithms on the different problems. We saw that RHC is the basic one. It gives poor results but is fast. SA is an improvement of RHC. It gives better results in general and is still as fast as RHC. GA and MIMIC are often the best optimisers. GA introduces the concept of population and crossover between iterations while MIMIC uses a probabilistic model to estimate the probability of a solution to be the best one which makes it the slower optimiser considered here.

3 Neural Networks

Introduction

In this section we will discuss the results of the randomised optimisation algorithms for training a neural network. There is some difference from the problems treated in the last section, mainly due to the supposed continuity of the weights in an artificial neural network. This can lead to issues when we want to to define a neighbor. The chosen solution adopted here is to update one weight by a random factor.

The problem I have chosen to explore is the Gender dataset from the previous homework. The purpose of this dataset is to find the gender given some appearence characteritics. A more complete description of this dataset is available in homework 1.

We will use the following notation for hidden layers: n_1, n_2, n_3 for a network with 3 hidden layers where n_1, n_2 and n_3 are the number of neurons in each layer respectively. The input and output are imposed by the dataset. The input layer is composed of 8 neurons. The output layer is composed of 1 neuron. The different networks tested are the following:

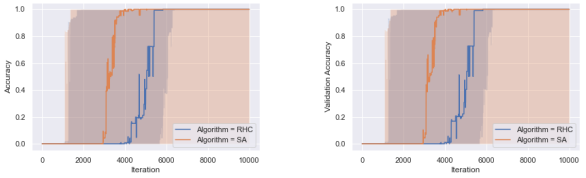
- (5): 46 weights
- (15): 163 weights
- (35): 316 weights
- $(30, 10)$: 561 weights

We can see on figure 13 there is no need to seek for more complex networks. Adding new weight would only increase the computation time without improving the accuracy, and also may lead to overfitting.

As an overview of the results, we can see that the best accuracy of 1.0 is find by each optimiser. One thing that is noticeable on figure 13 is the fast variation on the curves. Those variations are not easy to explain as they are present with RHC algorithm which does not permit the result to decrease. One possible explaination would be that the plotted metric is not the same as the one used for training. In fact, the first is sum of squared errors while the second is the ratio between the number of correctly classified sample on the total number of sample. Thus, the accuracy probably fell down when the network learns that it's better not to be a bit wrong on every thing rather than being correct on the wanted value.

Another thing to notice is that the training and the validation accuracy are very close. It could mean that the training and testing datasets are not independent. But I checked that it was not the case. So this means that the neural network is able to generalize to new data as well as it does not overfit. From now, we will only display the validation accuracy on the figures as the curves are very similar to the training accuracy.

In the next sections we will study more in depth the training process for each algorithm.



(a) Training accuracy on the Gender dataset for different algorithms

(b) Validation accuracy on the Gender dataset for different algorithms

Figure 13: Training and Validation accuracy for the different training algorithms

3.1 Randomized Hill Climbing

We present on figure 14 the results obtained with RHC. RHC succeeds to get the best accuracy on the validation set with all the different networks tested. The difference is the number of iteration required to converge. The best result is obtained with the network (35) with an accuracy of 1.0 after only 1800 iterations. The worst result is obtained with the network $(30, 10)$ with an accuracy of 1.0 after 6200 iterations. The network $(30, 10)$ is too complex here while the network (5) is too simple. An optimised network for that problem that gives a perfect iteration in the most efficient way is therefore the network (35).

RHC is again the fastest algorithm. It took approximately 1h to train all the different networks.

3.2 Simulated Annealing

We present on figure 15 the results obtained with SA. SA also succeeds to get the best accuracy on the validation set. But that time not for all the networks. The

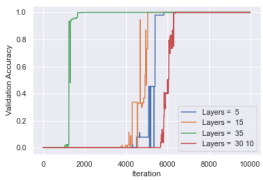


Figure 14: Accuracy with RHC for different layers

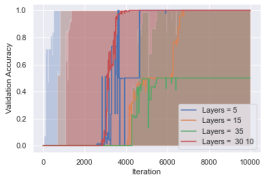


Figure 15: Accuracy with SA for different layers

best network with RHC is the worst with SA. The network (35) only manage to get an accuracy of 0.5 after 7000 iterations. The best result here is obtained with the network $(30, 10)$ with an accuracy of 1.0 after 4000 iterations. The other networks converge slower. I cannot really explain why the network (35) is so bad with SA and $(30, 10)$ suddently become the best network.

It took approximately 2h to train all the different networks with SA.

3.3 Genetic Algorithm

We present on figure 16 the results obtained with GA.

Conclusion

To conclude we can say that randomised algorithms can be helpful to optimise the weights of a neural network. But I guess this heavely depends on the dataset we are considering. Here, the gender dataset turned to be really well suited for neural networks.

Figure 16: Accuracy with GA for different layers

4 Conclusion

During this work I learned in a practical way the characteritics of different randomised optimisers. I saw that the fine tuning of the parameters were not the most important thing as it can be in supervised learning. But randomised optimisation and supervised learning do not have the same objective. The last part of this report showed that the first one can be used for optimising the second one. As always each optimiser has its specificities and it is important to know them to be able to use in a specific problem the most appropriate one.