Machine Learning Homework 2: Randomized optimization

Nicolas Six

March 10, 2018

Contents

1 Introduction

During this report we will explore four different randomized optimizer to see how well they manage to solve different problems. In that goal, we will compare them in terms of run time and result quality, exploring at the same time multiple meta-parameters combinations.

In a second time we will apply three of those randomized optimizer to try to learn a neural network on the Starcraft classification problem defined in the first homework.

The graph presented in this work follow the same convention as the one of the previous homework. The hard line represent the median value and the hue area fill the space between the minimal and maximal values at the given abscissa.

2 Optimization Problems

2.1 Introduction

In this section we will display the result we get by applying four different randomized optimizer to three optimization problem and draw some conclusions from them. The four algorithm we will apply here are: Randomized Hill Climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA) and MIMIC. Due to the physical limit of this work we will not spend time here to describe how those algorithms work as this is already explained in the lessons and is out of the scope of this document, but will try to link as much as possible their intern work to their apparent results.

The different problem we will use to make this comparison possible are: the Traveling Salesman Problem (TSP), the Four Peak problem and the Flip Flop problem. To perform our tests, we chose a set parameters. for example, each set of meta-parameter was run 10 times to prevent random artifact from misleading our conclusions. To let the time to every algorithm to converge we also let them run during 5 000 iterations. To improve readability of plots, some iterations giving no new information will be removed. The following subsections will cover those well-known optimization problem and some more informations on them.

2.2 Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is probably one of the most famous NP-Hard problem because it's a very simple problem to explain but still a very difficult one to solve. The TSP is given a set of point to visit, find the visiting order that minimize the path distance for visiting every point and going back to the departure point. It is easy to see that the number of possible path is factorial in the number of points to visit, thus exploring all the possibility is not a possibility. We can not say that it does not exist a polynomial algorithm to solve the TSP as it would mean that $P = NP$ which is still one biggest question in computational complexity theory.

For this problem, the accuracy value is defined as:

$$A = \frac{1}{distance}$$

With *distance* being the length of the loop. Thus, bigger is the accuracy better is the result, but the optimal accuracy is unknown.

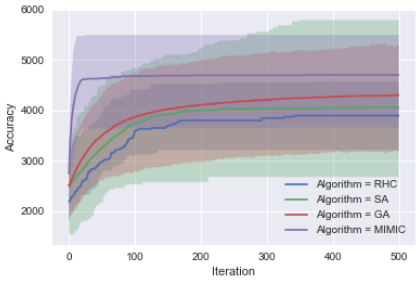


Figure 1. Accuracy on the TSP for different algorithms

Figure 1 display how the different algorithms performed on this problem. The first thing that can be said is that every algorithm as a lot of variation, none of them clearly stand out, with MIMIC being the best median performer with a fair margin. However, it must be noted that the time per iteration greatly vary from an algorithm to the other, going from 16.14s for MIMIC to 0.004s for RHC to perform all the 5000 iterations, in the worst case.

The following sections will give more details per algorithm.

2.2.1 Randomized Hill Climbing

RHC is probably not the most interesting algorithm to discuss here as it does not have a lot of parameters to play with. In our specific case the only parameters available are on the problem side and thus not specific to this algorithm.

As you can see on Figure 1, RHC is the worst performer on the TSP problem. This result is not really surprising as the RHC optimizer expect a sort of locally continuous distribution of the accuracy, so that going in the direction of the higher results mean something and is not the result of noise. The TSP is one of this chaotic problem, if the basin of attraction of a set of good solution is present, this set is full of local optimum where the RHC can get stuck and forbid to get the basin's maximum. Thus, getting a good solution on the TSP with RHC impose to start on one of the few points leading to a good solution. RHC is so a good way to get quickly a good average solution but does not give more chance to get an optimal one than picking one at random. Figure 1 is cropped to 500 iterations as between 500 and 5000 there is no changa in the accuracy, showing that the odd of picking a place to start is very low, as it can be expected when a factorial number of starting position are available.

RHC is the fastest of the four algorithms in terms of run time. It spends only 3,46 ms to run all the 5000 iterations for the slowest run on the TSP problem.

In terms of variation, the RHC is the one with the lowest variation explored here. This is easy to explain by the fact that RHC require the lowest number of run to explore the different parameters, so the optimizer were run on a smaller number of randomly generated TSP and the accuracy measure depend on the shortest possible path. In other words, the fewer instance you try the less chance you have to have one where easy to get a very good / bad accuracy.

In conclusion RHC is not well suited to solve the TSP problem and in the best case only gave us an average solution, but give it quickly.

2.2.2 Simulated Annealing

SA is the second worst algorithm on the TSP problem. Those results are easy to link with the RHC one of section 2.2.1, as SA is just a smarter way of doing RHC. An interesting fact is that, even if SA is the second worst algorithm in the median case, it's also the best and the worst in the max and min case respectively (with the same prior on result comparison disused on section 2.2.1).

Due to there similarity, most of the distributions around RHC is also applicable here. We will thus spend time here to explore the effect of the two parameters available for SA: initial temperature and cooling factor. The effect of this two parameters are displayed on Figure 2.

As you can see on Figure 2b, initial temperature don't impact the accuracy on this problem in the range we explored, but it's interesting to note that if the median value stay similar, the min and max value are more affected. You can see that a low initial temperature gave smaller minimal accuracy, whereas high value gave smaller difference between min and max. This was to be expected as low temperature are more likely to get stuck in sub optimal attraction basin and high value are more likely to go for a good solution but at the same time may skip small basin in the first iterations and thus skip better solution. This problem is particularly important with the TSP where neighbors are often very different in terms of accuracy.

Figure 2a represent the effect of different cooling values on the accuracy. As for the initial temperature, this parameter do not change much the median accuracy. This can be explained in the same way as we do previously, the TSP is not solvable as a hill climbing problem, the best solution may only have bad solution as neighbors.

In terms of run time, SA is also very close to RHC with the longest run at about 3.62 ms for the slowest run off 5000 iterations, and is so about 10% slower than RHC.

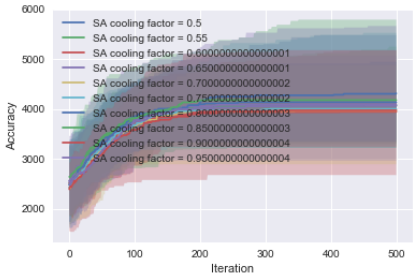
Thus, SA as a particular case of hill climbing, has similar result than RHC. If those results are slightly better than the one of RHC, they show that the TSP do not fulfill the conditions to be solved by that type of optimizer. Even if it must be noted that by including min and max accuracy SA is both the best and the worst performer on this problem.

2.2.3 Genetic Algorithm

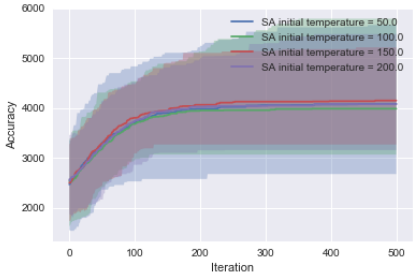
GA is here the second best performer for the TSP and achieve slightly better results than SA as displayed on Figure 1. This algorithm is also the one here with the most parameters to explore.

Figure 3 display a selection of the results obtained for the TSP. The fist thing we can observe is that as displayed on Figure 3a under a certain population size, GA do not perform as well is can, and above a problem dependent threshold the population size have a very little impact on the results.

The mutation number is less important, as you can see on Figure 3b, a small mutation number give in that case



(a) Accuracy on the TSP of SA, for different cooling factor



(b) Accuracy on the TSP of SA, for different initial temperature

Figure 2. Effect of the different parameters for the SA algorithm

better median result and do not imply important change in the variation range. Mutation is important in a GA as it is the way to introduce new comportment which were not present in the first iteration, but introducing too much of them risk muting a good predictor into a bad one before it has time to reproduce.

This explain the importance of a good mate number. A graphical representation of the evolution of accuracy with the iterations for various mate number is displayed on Figure 3c. On this plot you can see that the higher value explored here gave the best median results. It's interesting to note that a mate number of 550 and 450 gave us the best median and max results but that the first gave also the best min accuracy whereas the second gave the worst one. Showing that in the TSP case a good GA is one with few mutations but a lot of mating to try to combine the good solutions. You may have noted that this conclusion do not fit perfectly to our result as the lowest mate number slowly rise to be one of the best performer. Having very few mutations and mating, this population is very stable and only the best estimator are used in the mate process giving us a very slow way to get a result but one which is more likely to end up with good result. So you can see that they are here a sort of thread of 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. Thus, we can see this two option as a SA at high temperature in the first case and at low temperature for the second. If the two solutions perform well here, a compromise between them is not really efficient. In those middle ground case combine the disadvantage of the two: a stable population to small to properly keep track of the previous generations and a too small quantity of mating to properly explore the neighbors of the best solutions.

In terms of running time GA is order of magnitude slower than both RHC and SA. The longer run took 15.07 s to complete all the 5000 iterations which is more than 3000 times the one of SA.

Thus, GA do not present slightly better results than SA and RHC, but at the cost of far longer computation time.

2.2.4 MIMIC

MIMIC is, by a clear margin, the best algorithm tested here (Figure 1). In an iteration point of view it quickly gets to its maximum value and stay there for all the remaining iterations. MIMIC as two main parameters: the number of sample generated for each iteration and the number of sample to keep from an iteration to the other.

As you can see on Figure 4a, the number of sample used greatly influence the convergence speed of MIMIC. A few sample increase the iteration needed to make MIMIC converge on a solution and also slightly decrease the accuracy. While bigger value converge faster on a better solution, probably because it is easier in that case to estimate a meaningful distribution. It must be noted that in this case a number of sample of 250 perform better than 350. To the light of this result we can say that with a number of sample of 350 or above MIMIC will be affected by the noisy distribution of the TSP.

Figure 4b display how the accuracy behave for particular values of number of samples kept from an iteration to the other, in the particular case where the number of sample is 250. On this graph you can see that keeping more sample only decrease the median accuracy, but at the same time the best max value is get for middle ground number of 60 sample kept. Thus, we can suppose that the best possible value lie between 10 and 60 and so is the maximum number we can allow saving computation time without loosing too much on the accuracy.

In terms of running time, as expected, MIMIC get the worst results of all the four algorithms, with 16.14 s, one second more than GA. Making MIMIC iteration costly to run particularly in comparison with RHC and SA.

In conclusion MIMIC perform well on the TSP in comparison to the other algorithms but at the cost of a computation time far greater than the others.

2.3 Four Peak Problem

The Four peak problem is a simple problem designed to test random optimizer. Its name come from the fact that it has four local maximal, with two of them being global one.

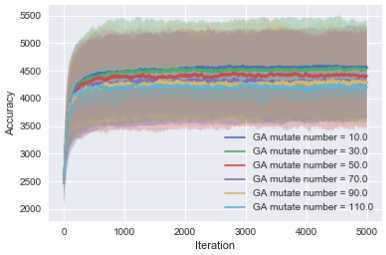
The function to optimize here can be defined as:

$$f(X,T) = max\left[tail(0,X), head(1,X)\right] + R(X,T) \quad (1)$$

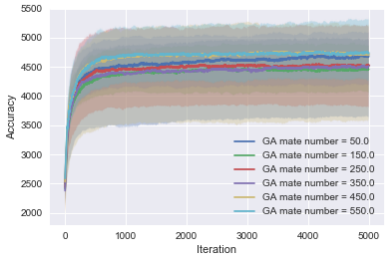
With:

- $X \in \{0,1\}^N$
- $tail(b,X)$ the number of b tailing X
- $head(b,X)$ the number of b leading X

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

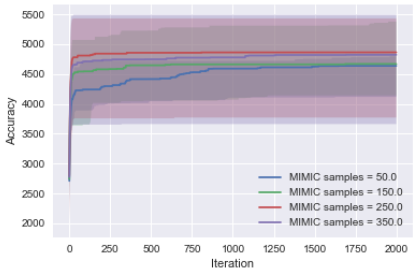


(b) Accuracy on the TSP of GA, for different mutate number, with population set to 600

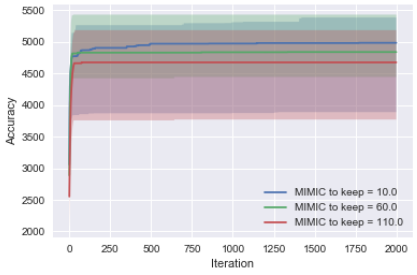


(c) Accuracy on the TSP of GA, for different mate number, with population set to 600 and mutate number to 10

Figure 3. Evolution of the accuracy with iteration for different GA parameters



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



(b) Accuracy on the TSP of MIMIC, for different number of estimator kept, with a number of samples set to 250

Figure 4. Effect of the different parameters for the MIMIC algorithm

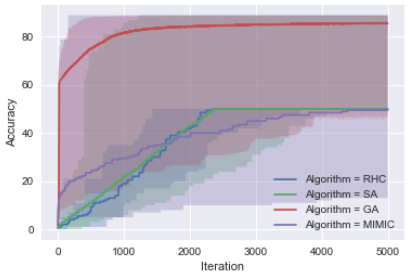


Figure 5. Accuracy on the Four Peak problem for different algorithms

and:

$$R(X,T) = \begin{cases} N & \text{if } (tail(0,X) > T) \wedge (head(1,X) > T) \\ 0 \end{cases} \tag{2}$$

The best solutions are thus get for X heading with $T+1$ 1 or tailing with $T+1$ 0, and the other two local maximal are for vector X full of 1 or 0.

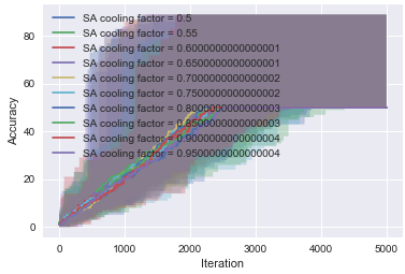
The accuracy that we are going to use in this section is the function f defined in equation 1.

This problem is challenging for hill climbing solutions as the attraction basin of the local maxima grow exponentially this the T parameter while attraction basin of the global one reduce at the same speed.

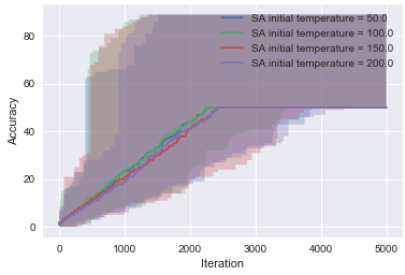
As you can see on Figure 5, GA clearly stand out and surpass all the other algorithms. This is not really a surprise as this problem was introduced as "four peaks: a problem designed to be ga-friendly". All the other algorithm top on the local maximal, with only SA managing to get max value on the global one. In the following sections we will explore with more details all the four algorithms studied here. To be more concise, we will not repeat the conclusion already stated in Section 2.2. Thus, the following sections will concentrate on the particularity of this problem.

2.3.1 Randomized Hill Climbing

As you can see on Figure 5, RHC get here average result. This come from the fact that the attraction basin for the two suboptimal peak is much larger than the one of the optimal one.



(a) Accuracy on the Four Peak problem of SA, for different cooling factor



(b) Accuracy on the Four Peak problem of SA, for different initial temperature

Figure 6. Effect of the different parameters for the SA algorithm

In the timing point of view, RHC is still very fast, with the slowest run of 5000 iteration taking only 4.23 ms to complete.

So we can say that RHC is too basic to overcome the difficulty of surpassing local maximal, with the relative advantage of finding the local maximal quickly. Such result are not surprising as this problem was designed to be nearly impossible for hill-climbing methods.

2.3.2 Simulated Annealing

SA is here the second best performer, as it is the only one able to sometimes go above the local maximal. But the median result on all the run performed here is still blocked by the local maximal.

As you can see on Figure 6, the different parameters have not much importance as in Section 2.2.2. We can add that with all the parameter explored here, given enough time, all the different version has their min value going to the local maximal.

On the timing point of view, SA as still very similar result than RHC, with 4,64 ms to run all the 5000 iteration in the worst case. Which is again, about 10% slower than RHC.

Thus, SA is, as for the TSP, bound to the same constraint as RHC but with slightly better results thanks to its temperature parameter.

2.3.3 Genetic Algorithm

As stated on the introduction, the Four Peaks problem was designed to be GA friendly. It is, thus, not surprising to see it dominating the results here, by a clear margin.

Figure 7 show an aggregation of the results get for this problem by GA. Most of the conclusion here are similar than for the TSP: a too small population do not allow to properly solve the problem while really large population do not really improve the results. At the best result where obtained with a small mate and mutate number, we understand it the same way as we did before, keeping track of the old best solution and moving around with only a small part on the population is more precise but far longer. As you can see on Figures 7b and 7c, this parameter combination is the only one where more than half the run end up with the actual global maximum and not something close to it.

In terms of run time, GA is faster here than on the TSP, and finished in the worst case in 1.48 s. It is still 300 times slower than SA, but 10 times faster than it was on the TSP.

GA really take advantage of this problem design as a large population and the mutation will allow him to find a way to get the R bonus of equation 1, and then explore around to get closer to the global optimum.

2.3.4 MIMIC

MIMIC can be considered as worse than SA as it achieve similar results, but take far more time to complete.

Figure 8 show how MIMIC perform on the Four Peaks problem. As described in section 2.2.4, convergence speed greatly depend on the parameters used. In addition, we can add that here a small mutation number, as displayed on Figure 8a get by far the best results and allow the minimal accuracy to find the sub optimal peak, as well as the median. Showing once again that saving computation time per iteration by using more already computed samples lead to the need of more iteration to let the algorithm converge.

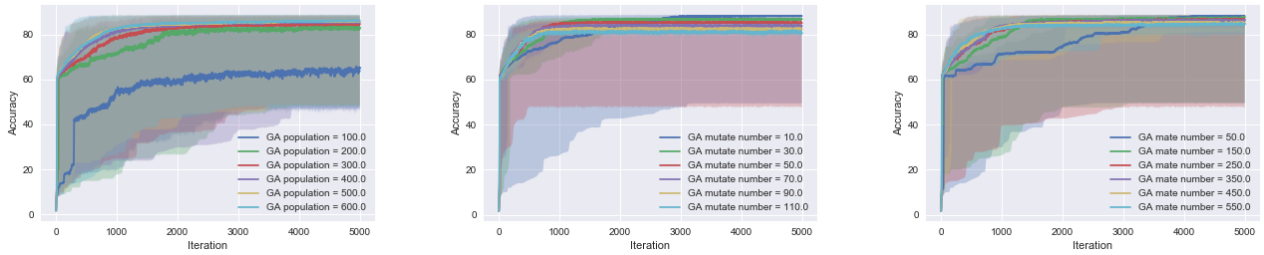
In timing point of view, MIMIC is still the worst performer with 16.73 s needed to perform the 5000 iteration in the worst case. This is about 3000 times more than SA for similar results.

MIMIC does not perform here as well as on the TSP, it manages to get to the global maximum in some case, but even with the best set of parameters most of the times MIMIC only find one of the local optimum.

2.4 Flip Flop problem

The last problem explored here is Flip Flop. This problem is very basic, the object to optimize is a string of bit where the value function is defined as the number of bit which are different from their preceding bit in the string. The optimum is easy for a human to find: a string of alternating 0 and 1. This answer is quite easy to find for a human being once the rule is given but finding it without knowledge is far harder.

In our case the string was 50 bits long, allowing a maximum score of 49. This score was obtained by two of the four algorithms: SA and MIMIC. While GA managed to get 48 and RHC stopped at 43. However, SA find one of the two best solution in far more case than MIMIC and in addition doing so quicker in terms of wall time as well as iteration, as you can see on Figure 9.

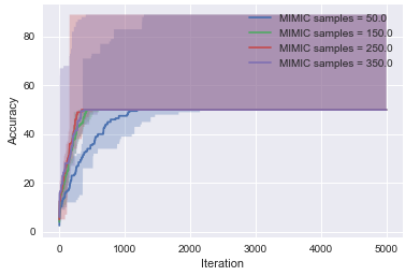


(a) Accuracy on the Four Peaks problem of GA, for different population size

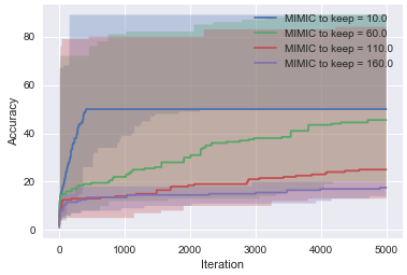
(b) Accuracy on the Four Peaks problem of GA, for different mutate number, with population set to 600

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

Figure 7. Evolution of the accuracy with iteration for different GA parameters



(a) Accuracy on the Four Peaks problem of MIMIC, for different number of samples, number of sample to keep set to 10



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

Figure 8. Effect of the different parameters for the MIMIC algorithm

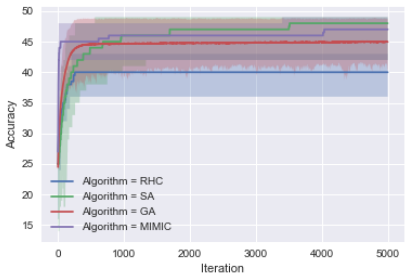


Figure 9. Accuracy on the Flip Flop problem for different algorithms

As in the previous sections, the following parts will cover more in depth the results on an algorithm basis, without repeating uselessly what was said previously in this report.

2.4.1 Randomized Hill Climbing

RHC is as usual the poorest performer here. The best result it managed to get was only 43 before getting stuck into a local maximum. Those results show again the importance of being able to allow a little accuracy loss to overcome those little griping points.

Here we can assume that RHC managed to grow some flip-flop string inside the main string, but that those strings were not synchronized and thus cannot be merged into a single one without reversing some of them. But reversing a flip-flop string, is at best done without accuracy loss, such plateau are very hard to overcome for RHC.

The results of RHC here are not surprising, in terms of wall time is again one of the fastest, but only have here the second position with 2.75 ms for the longest run of the 5000 iterations.

Once again RHC show here the strength and mostly weakness of its basic conception.

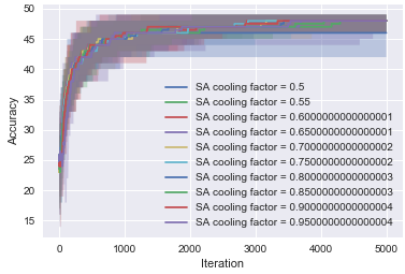
2.4.2 Simulated Annealing

SA is for this problem the best performer. As already stated in the introduction, SA manage to get the best possible score far more often and quicker than other algorithm, such as MIMIC.

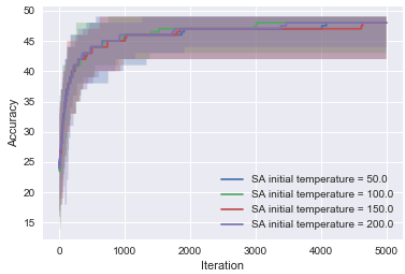
Figure 10 display the impact of the different parameters on the result. They are not much to say here that was not already said previously. We can note that a cooling factor of 0.5 is far too small to properly converge on the good result and that some initial temperature gave better convergence speed than others, but median end result stay the same.

On a timing point of view, SA is here also the best, taking only 1.61 ms to run all the 5000 iterations in the worst case. This is 40% better than RHC, which is surprising giving that is slightly more complicated in term computation per iteration. We suspect here that something must have slowed down the RHC process on our test machine, as the final run time are very small.

SA is here by far the best performer showing that in such a problem being able to go through little loss can



(a) Accuracy on the Flip Flop problem of SA, for different cooling factor



(b) Accuracy on the Flip Flop problem of SA, for different initial temperature

Figure 10. Effect of the different parameters for the SA algorithm

help a lot. It's also important to note the implementation we used (ABAGAIL) allow SA to move from a neighbor to the other if they have the same value, which is not the case for RHC. Thus, SA can pass the plateau mentioned on section 2.4.1.

2.4.3 Genetic Algorithm

GA is here in third position for the first time. If the overall comportment of GA according to its parameters is very similar than the ones observed previously, this relative loss may come from the fact that mixing to string together as in the mating process do not really made sens here as the flip-flop string must be synchronized for this to work properly.

The time spends by GA to run all the iterations in the worst case was 1.40 s, which stay on the average time noted on the previous problems.

In conclusion, we can say that here GA is too complicated for the task at hand. The result given are correct but do not represent the time spend to find them.

2.4.4 MIMIC

MIMIC is here one of the two algorithms which manage to get the maximum possible score. But in this case it's still hard to consider that as good performance.

In terms of parameters effect on the score, the plots on Figure 12 are very similar to the one of the others problems. For example, here, as before, a few sample kept from an iteration to the other perform better.

The run time of MIMIC is very similar to the one of the others problems with 15.70 s.

As for GA, MIMIC do not take advantage of his complexity to find good solutions quickly. This probably due to the fact that setting one bit to a particular value is not good nor bad, without knowing his neighbors. Thus, the structure MIMIC is looking for is not stable enough.

2.5 Generalization

As you must have noted in the three previous sections, all the algorithms behave in a similar way from a problem to the other. We will here try to summarize the benefit and drawback of each algorithm and display direction for further investigation which we were not able to fit on this report.

RHC is a very basic algorithm and thus has very basic results, but has the advantage of running quickly.

SA is a little more complete than RHC and allow small loss of score to when it start and then cool down to RHC to find the local maximum where it ended up. It is thus quick and often get correct results. The two parameters are easy to deal with do not need to be fine-tuned to give correct results.

GA is often the second best solution, but is slower than the two previous one by some order of magnitude. We have seen here two ways of using GA, a first one where you use a large population to find as fast as possible a good score and a second one where most of the population act as a memory and small changes allow a slow exploration.

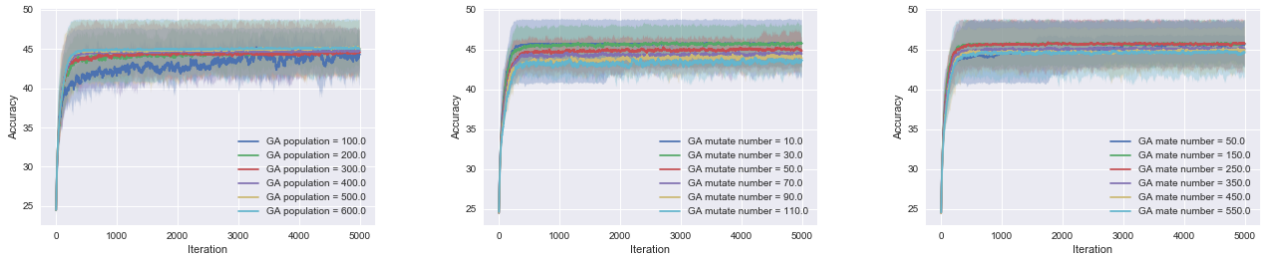
MIMIC appear here as a binary algorithm, on some problem as TSP or Flip Flop, it obtains very good results while on the Four Peaks problems, it is the poorest performer. In terms of parameter, below a problem dependent number of sample MIMIC is not able to understand the problem correctly, but giving more sample nether speed up nor improve further the results. The number of sample to keep is more a way to save computation time than improve the results, thus this parameter must be tunned with time in mind to be useful.

Time is obviously the weakness of this report. A lot of thinks can be said on the different algorithm and problems, as for example studying the time effect of each parameter, but we were not able to fit such study into this report. Figure 13 is for example a graphical representation of the times reported in the previous sections, showing that much can be said about it, but requiring far more space than available here.

3 Training a Neural Network

Using random optimizer to train a neural network (NN) is another potential application. There are however some little difference from the problems treated in section 2, mainly due to supposed continuity of weight in an artificial neural network. This can pose problem when you want to define a neighbor, the solution chosen here is to change one weight by a random factor.

The problem we choose to explore here is the Stracraft 2 data set used in homework 1. The goal of this dataset is to

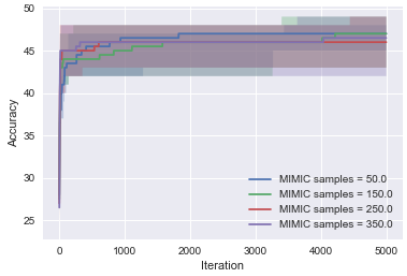


(a) Accuracy on the Flip Flop problem of GA, for different population size

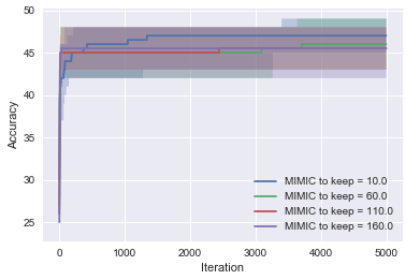
(b) Accuracy on the Flip Flop problem of GA, for different mutate number, with population set to 600

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

Figure 11. Evolution of the accuracy with iteration for different GA parameters



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



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

Figure 12. Effect of the different parameters for the MIMIC algorithm

find which player was on the computer given only a set of statistic on the game (APM, first use of a key...). This task is far from being straightforward as the dimensionality of the state space is very important and all the data are noisy. A more complete description of this dataset is available in homework 1.

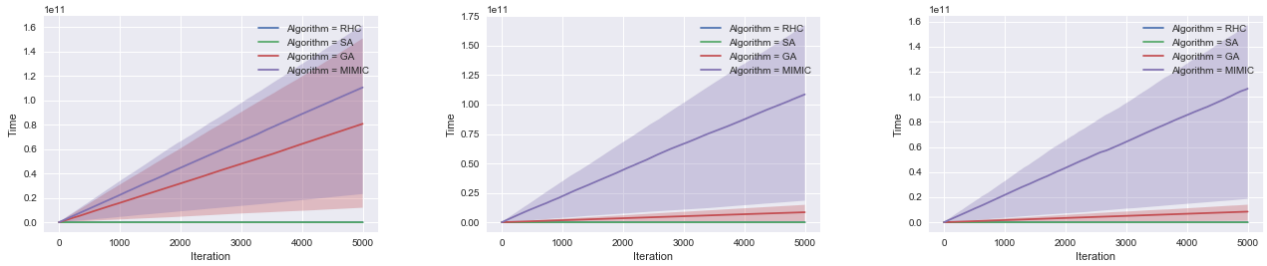
We used here the same neural network structure than in homework 1. We only had one or two hidden layer with from 5 to 35 neurones. In this section we will use the following notation for hidden layers: (a, b, c), for a network with three hidden layer and a, b, c neurones on the first, second and third hidden layers respectively. The first and last layers of the network are fixed by the problem studied here: 72 input neurones and 200 output. For a better understanding of the results we provide you this list of the tested layers and theirs characteristics.

- (5): 1 565 weights
- (15): 4 295 weights
- (35): 9 755 weights
- $(30, 10)$: 4 700 weights

With only 1 950 training example and a limit of 10 000 iteration, you can see that there are no reason to seek for more complex network. Adding new weight will only worsen the training time and if better result might be expected the training time required to get there forgive us to try. In addition, most of the neural network used here will overfit the training data given enough time, seeking higher complexity will only worsen that problem.

As an overview of the results, we can say that they are no comparison possible with the results of homework 1. As you can see on Figure 14 the global result are far from good. During training the best classifier managed to correctly classify 35 % of the training set and 33% of the validation one, which is quite low in comparison to the 80% get on homework 1.

The most notable information of this graph is the sudden variation in it. Those variations are difficult to explain as they append on the training dataset with algorithm such as RHC that does not allow a result to decrease. The only possible explanation for that is that he plotted metric is not the same as the one used for training, in fact

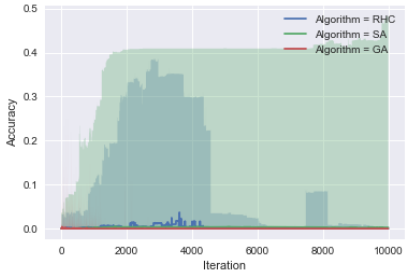


(a) Evolution of time according to the iterations, for the TSP

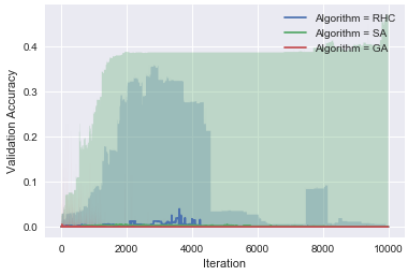
(b) Evolution of time according to the iterations, for the Four Peaks problem

(c) Evolution of time according to the iterations, for the Flip Flop problem

Figure 13. Evolution of time according to the iterations



(a) Accuracy on the Starcraft problem for different algorithms



(b) Validation accuracy on the Starcraft problem for different algorithms

Figure 14. Test and validation accuracy for all the training algorithm

the first is SSE while the second is the ratio between the number of correctly classified sample on the total number of sample. Thus, the accuracy probably fell when the network learns that it's better not to be a bit wrong on every thing, saying 0.1 instead of 0.0, than being correct on the wanted value.

The other thing to note is that training accuracy is suspiciously close to the training one. This made us double check our train and test data set to ensure that they are independent. As we found no problem there, we suppose that the neural network has a good generalization, and do not overfit. As it is generalizable to the others figures, to save some precious space on report we will display only the validation accuracy on the next figures.

In the following section we will study more in depth the training process algorithm by algorithm.

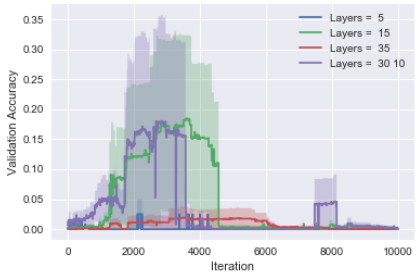


Figure 15. Accuracy on the RHC for different layers

3.1 Randomized Hill Climbing

RHC achieve here correct result, in referential of this section. If it did not achieve to get the best results, we have to highlight that it was able to give correct results for two different trained networks while tested on only 8. As you can see on Figure 15, the layers (5) and (35) do not achieve to get correct results. This can be explained by chance, but with the weight in mind one is probably too complex and the other too basic for random optimization to work properly.

RHC is again one of the fastest algorithm and took only 1 min 27 sec to perform the full training of 10 000 iterations in the worst case, with an average of 1 min.

3.2 Simulated Annealing

SA is here the best performer getting close to 45% of accuracy, in one case. Figure 16 display those results, which are hard to interpret as only 2 NN out of 140 converged toward correct value. It is so hard to distinguish the chance from the effect of parameters, as the initial random network probably as a bigger importance than the parameters of training. We can note that again the (35) network is the poorest performer probably due to its complexity while $(30, 10)$ and (5) get very good results.

In terms of wall time SA is very close to RHC with 1 min 29 s in the worst case and also 1 min in average.

Thus considering the difference in result and timing, if one has to chose between RHC or SA to train a neural network, the good chose is definitively SA.

3.3 Genetic Algorithm

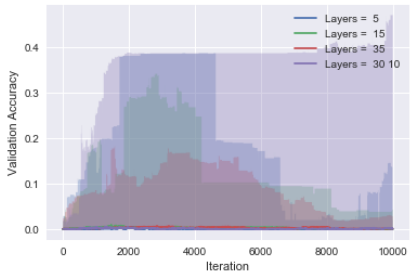


Figure 16. Accuracy on the SA for different layers

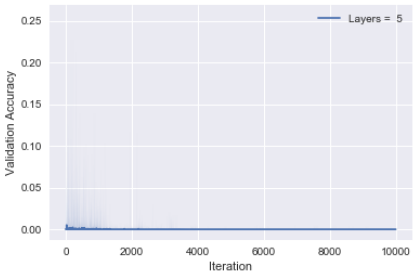


Figure 17. Accuracy on the GA for different layers

3.4 Generalization

From the results above, I think that we can say that random algorithm are not the best way to train a NN. NN are black boxes with a lot of weight that may appear to be random, but trying to change them randomly is not the more efficient way to train one.

The efficiency of this training process probably explain why so few library implement it. But such an exploration of these techniques is good in both a research and learning point of view.

4 Conclusion

The study carried here allowed us to understand more in depth random algorithms by applying them in different context. If the results where never outstanding, they were good most of the time. Random algorithm will not be the first things we will use to solve a given problem but probably the second as they are easy to implement and give some insight on the problem complexity.

The study of the effect of the algorithm's parameters showed that, while they are important, the fine-tuning is not required to get correct results, which is completely different from the supervised learning case. But randomized optimization do not have the same objective as supervised learning, has the Section 3 show, random optimization can be part of a supervised algorithm, with more or less success.

In conclusion, we have seen here that as often there is no perfect solution or universal solver. Some randomized algorithm will perform well one some case but badly on others. Without practice of these algorithms it would have been hard to determine given a problem which algorithm chose to get the best results. Thanks to this work, randomized optimization is now an additional tool we are susceptible to use on other problem.