Wrocław University of Science and Technology

Introduction to Artificial Intelligence

Laboratory 1: Genetic Algorithms

By:

Sindy Kola 245903

1. Analysis of the impact of the crossover probability**

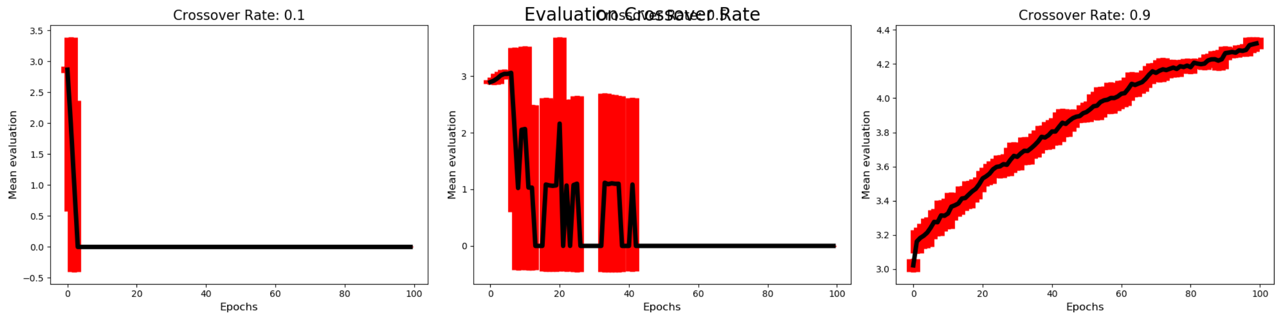

Fig. 1 Evaluation of the Crossover rate

The crossover probability is basically defining what portion of the population of our individuals is allowed to reproduce (swap their genetic information after the crossover point). With a rate of 0.1, not a lot can do so, therefore we can not move to a better solution on the graph. The situation is similar on the plot in the middle. On the third plot we can observe that the algorithm is giving us much better results with a crossover rate/probability of 0.9.

Crossover is a convergence operation which is intended to pull the population towards a local minimum/maximum. Since the end goal is to bring the population to convergence, Crossover needs to happen more frequently. I tested for the values: 0.1, 0.5 and 0.9.

2. Analysis of the impact of the mutation probability**

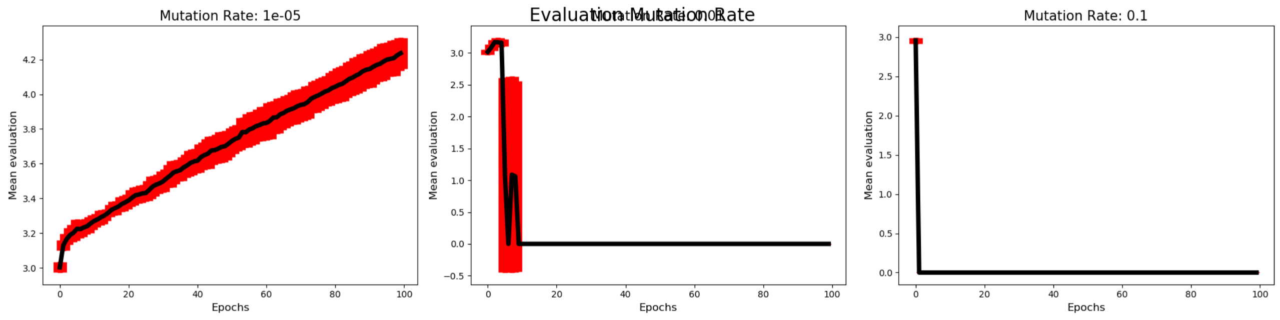

Fig. 2 Evaluation of the Mutation rate

Mutation, as opposed to Crossover, is a divergence operation. It is intended to occasionally break one or more individuals of a population out of a local minimum/maximum space to potentially discover a better minimum/maximum space. As such it needs to happen less frequently, since what we are actually doing is changing the genes of an individual, and it will result in the individual wandering in the opposite direction of the solution.

Mutation maintains genetic diversity in the subsequent generations which means that we avoid premature convergence on a local maximum/minimum. With a mutation probability/rate which is too high, the convergence is slow or it does not occur. Lowering this rate we start achieving better and better results. During the various tests and executions of my program, I found that I got the best results for 1e-05 (0.00001) and 1e-04 (0.0001). The values I tested for were: 1e-05, 0.01 and 0.1.

3. Analysis of the impact of the tournament size**

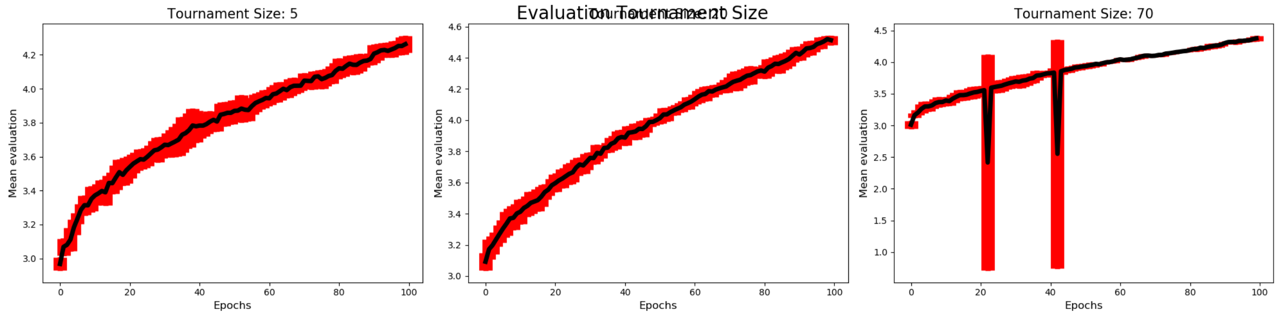

Fig. 3 Evaluation of the Tournament size

In this part of the report, using different values of Tournament sizes (5, 20 and 70), we try to compare how it affects the end results. In short, the bigger the tournament size, the more fitter individuals participate in reproduction.

In Tournament, the higher the tournament size, the harder the selection is. In hard selection, we give preference to the better individuals to have children and do not allow the others to reproduce. In soft selection, we give preference to the better individuals to reproduce, but still allow the other individuals to have children at a very low rate. Depending on the problem at hand we might prefer one method or the other. However, generally, we say that soft selection is better since it focuses the individuals/population on a local optimum.

In the third graph, one might say that the algorithm is not working properly, however that is not the case at all. On the contrary, the third graph is a perfect depiction of a working algorithm. We can observe this easily. It can change the way that it works by jumping out of the local minima when it hits them.

4. Analysis of the impact of the population size**

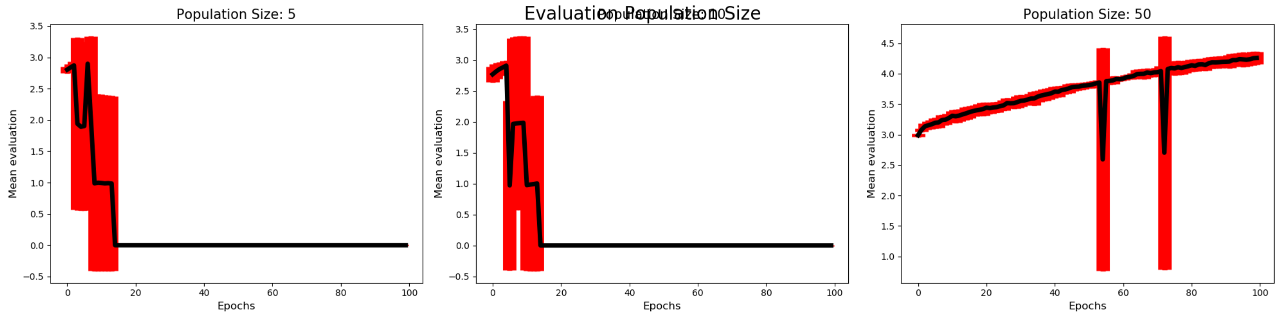

Fig. 4 Evaluation of the Population size

The tests were conducted on population sizes of 5, 10 and 50. We can see that for a small population size, our individuals were not able to mutate to better individuals. The population, in both cases where the population size was 5 and 10, moved very quickly in the wrong direction, thus not obtaining a good solution. With the increase of the population size, we also increase diversity, allowing for a better exploration and exploitation, thus achieving better results.

5. Comparison of the best solution with any of the non-evolutionary methods

Since for every new run, my program generates different tasks, the results slightly differ from one execution to another even though the parameters remain the same. With this in mind I tried different implementations to this solution to compare the results I obtain with this evolutionary approach.

For starters, a random working algorithm would not be a good idea in order to come to a good solution of our problem. If we were to choose between this approach and the genetic approach, we should definitely opt for the latter. The reasons for this are simple. Firstly, while using the genetic approach we do not forget previous points, and furthermore the fittest deemed individuals will produce more children. Moreover, while using the GA approach we do not only use the information for a specific point in the population, but we can also tell how much better or worse this point is compared to other individuals.

As for other implementations of the solution of this problem, I attempted to compare my results with other known solutions to this problem. Namely, I tried adapting and modifying solutions offered by Google OR - Tools. However, even though at first they seemed to be working fine, with the increase of the number of tasks, it appeared to be a difficult to solve problem, even using approaches such as Branch and Bound and Dynamic Programming. For a small set of tasks, the example found on the GitHub repository* of Google OR-Tools seemed to work fine, however with a large dataset as input, it just kept executing for large amounts of time with no results.

** - The values on the y axis of the chart are the mean evaluation. They were reduced by a line in the code which I found to be redundant and later removed (it was used just for illustrative purposes of the charts). However I kept the results yielded by that line (param_history /=1e05), because they provided a very good representation of all the elements that needed to be showcased in the analysis given in this report.

* - https://github.com/google/or-tools