Project Report

Evolutionary Computation 2021/22

Miguel Galvão - 2018278986

1 Introduction

The goal of this project is to compare the performance of different algorithms used to solve problems through evolutionary computation techniques. To do so, I will attempt to do a statistical comparison among the various options in order to draw conclusion on which version best works for the problems in question. Throughout this report, I will describe structure of the project and the strategies used, as well as all the work developed and results obtained.

2 Problem Description

The variations of the algorithm to be analyzed come down to the crossover operator used to recombine the genes of two different individuals. More specifically, the two operators to be compared are the Partially Mapped Crossover (PMX) operator and the Order Crossover (OX) operator. These are crossover algorithms based for permutation problems, meaning the genes of a chromosome remain the same, but their order changes. I will shortly describe each one in the following subsections.

2.1 Partially-Mapped Crossover (PMX)

This crossover algorithm maintains a portion of the order of the genes in the chromosome. The parents start by directly exchanging a sub-sequence of their chromosome between them, maintaining the order. Then, for each new number in the chromosome (received by the other parent), the corresponding "lost" number is placed in the position where the new number was before the exchange. If that position is already occupied by a new number, then we find that number's previous position, and repeat that process until an available position is found. If any numbers remain (not added or lost), they stay in the same position. This way we successfully create two new individuals from the two parents, mapping the genes position according to the randomly chosen sub-sequence. We can see a simple example in figure 1:

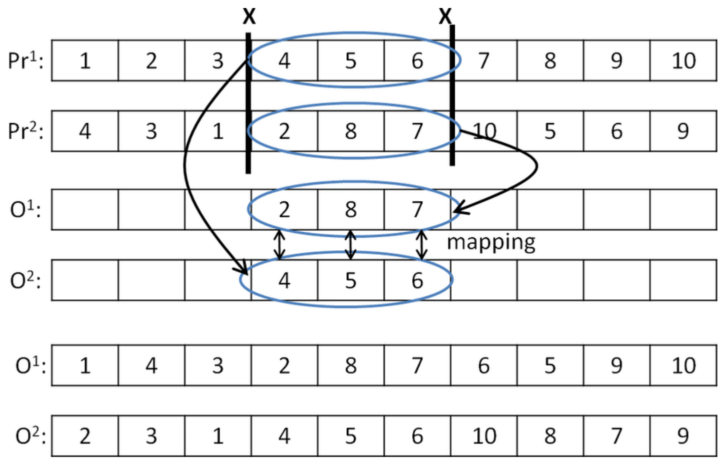

Figure 1: Example of PMX operation. Source

The parents start by exchanging the sub-sequences 4,5,6 and 2,8,7 between them. Then, if we look at the second child, for example, we look for the position where the new number 4 previously was (first position) and place the "lost" number 2 in that position. That process is repeated for every number for both parents

2.2 Order Crossover (OX)

While PMX maintains the order of the genes from one of the parents, this algorithm maintains the order of both. OX starts the same way, by selecting a sub-sequence of one of the parent's chromosome. After that, it takes from the second parent the remaining numbers not yet in the child, placing them in the same relative order as they were in the parent, starting at the position where the sub-sequence ends. The same is done for the second child, switching the "roles" of the parents A simple example is presented in figure 2:

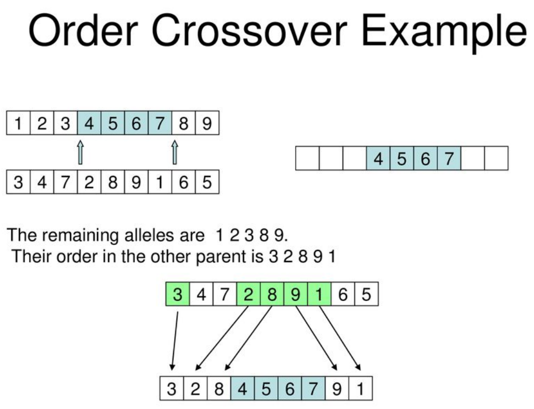

Figure 2: Example of OX operation. Source

From the first parent, a sub-sequence of 4,5,6,7 is randomly selected for the child. Then, the remaining numbers (1, 2, 3, 8 and 9) are added in the same relative order to the new chromosome.

3 Experimental Setup

In order to analyze the performance of both operators, I will make use of the N Queens problem and construct two versions of the same evolutionary algorithm, each with one of the different crossover operator previously mentioned.
Given that these are stochastic algorithms, with random input regarding the initial population and factors such as mutation and crossover probability, all performance tests will be performed 30 times, with a population of 150 individuals and a length of 300 generations. For each of the 30 runs, the overall best individual's fitness is stored, as well as the generation where it was found. In addition to that, the same random seed is used for both operators (one for each run) in order to make a fair comparison. At the end, we will obtain four distributions of 30 values, two for each crossover operator, one for the value of the best fitness and on for the generation where the best individual was found.
More details are described in the following subsections.

3.1 N Queens problem

The goal of this problem is to display a certain number of queens in a chess board, one for each column, in such a way that no queen is attacking another queen, through chess rules. That means no pair of queens should be in contact with another diagonally, vertically or horizontally. In Figure 3 we can see an example of a valid solution of the N Queens problem with an N value of 8, which means laying out 8 queens in a 8 by 8 chess board.

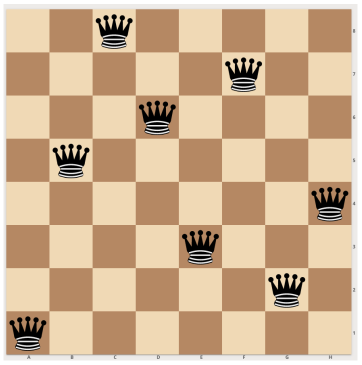

Figure 3: Example of correct solution for N Queens problem. Source

3.2 Fitness and Initialization

To add a desirable complexity level to the problem, we'll use the N Queens problems with an N value of 64, meaning each individual is a possible solution do display 64 queens in a 64 by 64 chess board. To calculate the fitness of an individual, we start by initializing it according to the size of the board, meaning, in this case, each individual starts with a fitness of 64, and deduct 1 for each violation, previously described. This means the maximum fitness an individual can have is 64.

3.3 Parameterization

In order to figure out appropriate values to successfully compare the different crossover algorithms, I ran several tests with various combinations of values for the probability of mutation and probability of crossover. The values tested are the following:

- Probability of mutation: 0.1%, 1% and 5%
- Probability of crossover: 70% and 80%

For each combination (6 in total), 5 tests were run with the N Queens problem with a N value of 32, calculating, for each generation, the best and average fitness of the population, as a mean of the 5 runs. Given the reduced complexity of the task (compared to the vale 64 for N), only 200 generations are ran, with a population size of 75. The results obtained from the different combinations are shown in figure 4:

3.4 Final Parameters

As we can observe, the set of parameters that obtained the best overall results are a probability of mutation of 5% and a probability of crossover of 80%, so those are the values used for the main analysis. Regarding the remaining parameters of the evolution algorithm, a tournament selection of 3 elements was used for the parents selection, as well as an elitism of 2% of the size of the population. A swap mutation was used as the variation operator.

4 Result Analysis

As previously mentioned, each crossover operator generated two distributions of 30 numbers corresponding, respectively, to the best fitness value and the generation in which it was found in each run. Before applying any other statistical tests to the data, we must first figure out whether or not it follows a normal distribution. To do so, we'll apply the Shapiro-Wilk test, with a confidence level of 0.05. In this test the null hypothesis states that the data is normally distributed, which means that if the

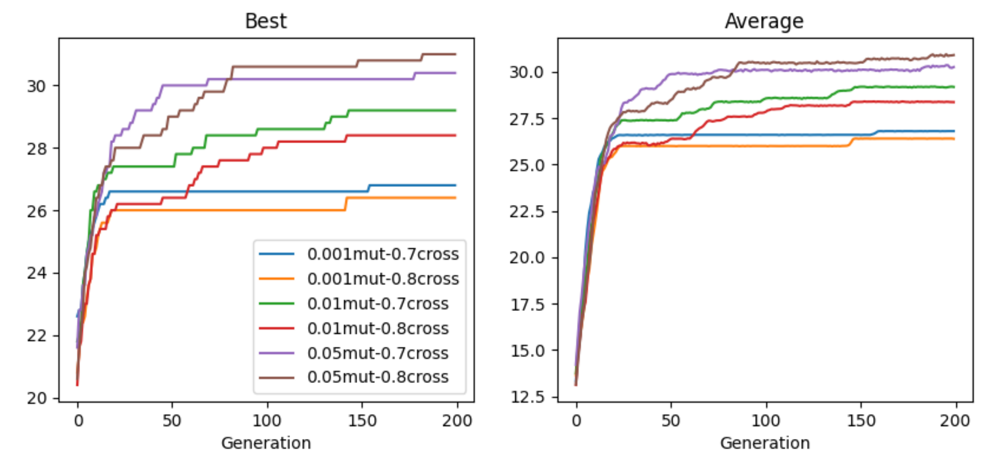

Figure 4: Parameterization results

P-value resulting from the test is a value under our chosen confidence level, then it means the null hypothesis is rejected and we assume the data does not follow a normal distribution. In figure 5 we can observe the results of the test for each distribution.

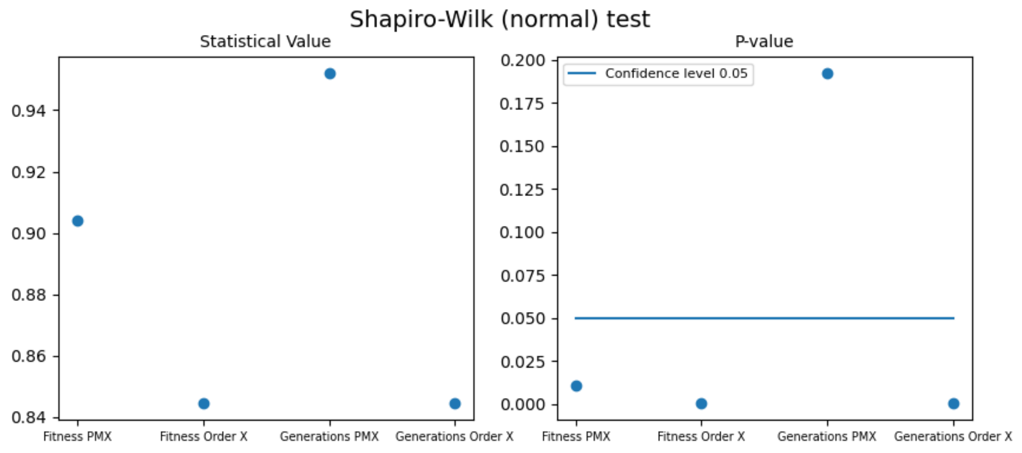

Figure 5: Shapiro-Wilk test results

From the resulting P-values obtained, apart from the distribution of the generations of the PMX operator, all the other values are below the confidence level, meaning that the null hypothesis is rejected. Since the distributions don't follow a normal distribution, we must apply non-parametric statistical tests in order to properly analyse the data. In the following subsections that analysis is done through the Wilcoxon signed rank test, with a null hypothesis stating that there is no statistical difference between the two distributions. This means that if the P-value obtained is lower than our confidence level, we reject that hypothesis, indicating that conclusions can be draw from the data collected.

4.1 Fitness analysis

After applying the Wilcoxon signed rank test from the distributions of the best fitness value, we obtain the following results:

- Statistical value: 128.0
- P-value: 0.498860913135171

As we can see, the P-value is well over the confidence level of 0.05, meaning we can't reject the null hypothesis, thus concluding that there is no statistical difference between the distributions. Therefore, no conclusions can be drawn from this data.

4.2 Generations analysis

After applying the Wilcoxon signed rank test from the distributions of the generations where the best fitness value is found, we obtain the following results:

- Statistical value: 120.5
- P-value: 0.03594287252780737

In this case the P-value is under the confidence level, which means we reject the null hypothesis and conclude that there is statistical difference between the two distributions.
By taking a look at the mean of the values in each distribution, we find that for PMX, the best individual is first found by the 200th generation (200.(3)), whereas for OX that value rounds the 233rd generation (232.7(3)).

5 Conclusion

In this project I attempted to compare the performance of two different crossover operators in the context of the N Queens problem. This means that even though one of the operators might work better than the other in this particular problem, we can't generalize that comparison for all permutation problems. However, for the N Queens problem we can draw some conclusions.
First, there doesn't seem to be a difference between the final results obtained by the operators. As we saw in the result analysis, the distributions of the fitness are statistically indifferent, meaning we can't claim that one of the operators yields a better individual than the other at the end of all generations.
On the other hand, looking into the values of the generations where the best fitness was first found, we find a clear difference. The mean of the distribution is lower for PMX than for OX, meaning that PMX reaches the best results in less generations or, in other words, faster.
Combining these results, we can conclude that for the N Queens problem, the PMX and OX operators reach similar results, but PMX does so faster.