University of Córdoba

Computer Science Engineering Degree
3rd course

Metaheuristics

Evolutionary Computation

Valentín Gabriel Avram Aenachioei
Alexandra-Maria Borsan
Aicha Bracken-Soliman
Radu Ciurca
Barbara Kereselidze
Víctor Rojas Muñoz
Davit Tchanturia

Academic year 202-2023
Córdoba, June 9, 2023

Contents

List of Tables

List of Figures

In this paper, we will set out how we have done the second parctical assignment, *Evolutionary Computation*, answering the questions asked in the instructions of the lab assignment.

1 Genetic Algorithm

1.1 Implement different genetic operators (crossover and mutation). Analyze which combination and probabilities produce the best results.

For this analysis, we have used a large number of hyperparameters. In this first basic approach we will analyze the impact of the best combination of genetic operators, crossover an mutation, since they are the most influential factor on the genetic algorithm.

To obtain some visible results, we only compared the obtained results from the **7a.txt** dataset, since the rest of datasets have an accuracy close to 100% or 0%.

As crossover operators, we have used three different:

- **Single point crossover:** We select a random gene from both possible solutions, and we swap the values.
- **Uniform crossover:** For each gene in each possible solution, we randomly select if we keep the value or we swap it.
- **HUX:** Half Uniform Crossover. We combine half of the bits among those where parents differ.

As mutation operators, we only have used two different:

- **Single bit flip:** We select a random gene from each and we flip its value.
- **Random flip:** We select a random gene from each solution and we change its value randomly.

Both Single point and Uniform Crossover operator could be optimized, avoiding the selection of genes close to the limits of the solution, (first and last genes). Moreover, we apply those changes based on a probability for each type of operator.

As a comparative measure, we used accuracy, in terms of how many test obtained the best possible combination, with every posible combination of hyperparameters.

As we can observe in the table 1, we compared every combination of crossover and mutation operator to select the best possible combination of both.

Mutation	**Crossover**	**Accuracy**
Single Bit Flip	HUX	68.518 %
Random Flip	HUX	64.814 %
Single Bit Flip	Single point crossover	64.814 %
Random Flip	Single point crossover	59.259 %
Single Bit Flip	**Uniform crossover**	**72.222 %**
Random Flip	Uniform crossover	68.518 %

Table 1: Genetic operator comparison

The image 1 shows the comparison between all the genetic operators combinations.

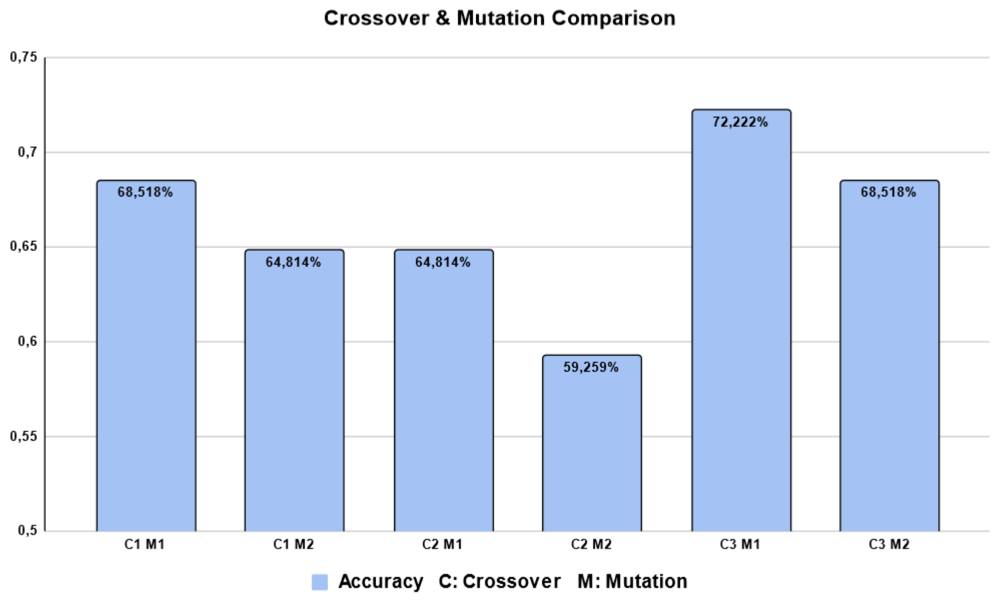

Figure 1: Genetic operator comparison

As we can see, the best combination of hyperparameters, in this dataset, is using Uniform crossover and single bit flip as genetic operators, with an accuracy of 72.22%.

Now we need to prove the importance of probabilities in the performance of the algorithm, based on this best genetic operators combination. The results of these tests can be found in the table 2, where we compared every combination of crossover and mutation probabilities.

The image 2 shows the comparison between all the genetic operators combinations.

As a conclusion, we have determined that the best perfomance is obtained using Uniform Crossover with Single Bit Flip Mutation, using *0.5* as Crossover probability value and *0.05* as Mutation probability value.

Crossover probability	Mutation Probability	Accuracy
0.5	**0.05**	**75,93%**
0.7	0.05	64,81%
0.9	0.05	61,11%
0.5	0.1	73,22%
0.7	0.1	72,22%
0.9	0.1	61,11%
0.5	0.2	68,52%
0.7	0.2	66,36%
0.9	0.2	72,22%

Table 2: Genetic probabilities comparison

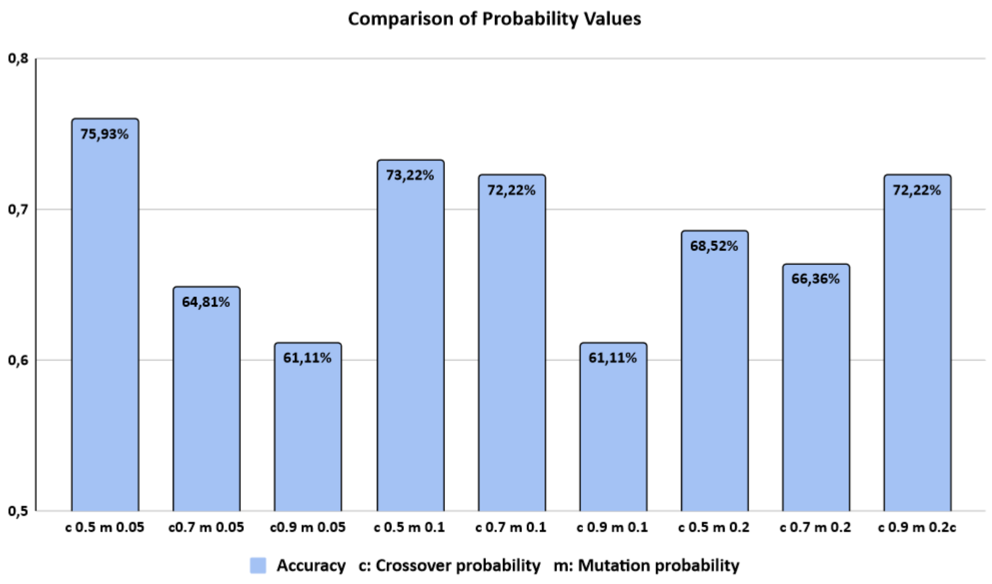

Figure 2: Genetic probabilities comparison

1.2 Do a hyperparameter study: number of solutions (population size), number of generations, tournament size, probability values (genetic operators), etc. Provide the best parameter values.

Once we have the best genetic operators combination and the best genetic propabilities combination, we can analyze the performance of the algorithm based on other hyperparameters, such as Population size or maximun number of Generations.

As in the other question, we only compared the obtained results from the **7a.txt** dataset, since the rest of datasets have an accuracy close to 100% or 0%. We compared different combination of hyperparamets using the best genetic operators combination.

As we can observe in the table 3, we compared every combination of population size and number of generations to select the best possible combination of both. We consider as best options those which obtain the optimal solution.

Population Size	**Maximum Generations**	**Best Solution**
10	10	94
10	50	96
10	**100**	**107**
25	10	105
25	50	105
25	100	103
50	**10**	**107**
50	50	106
50	**100**	**107**

Table 3: Solutions obtained by hyperparameter

The image 3 shows this comparison in a graphical way.

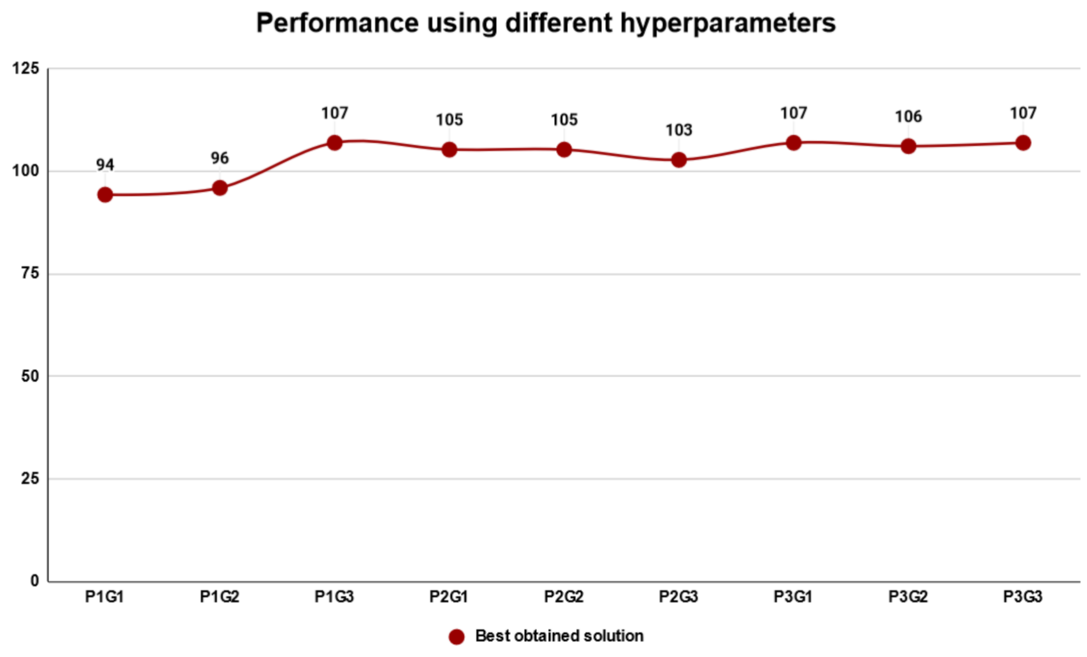

Figure 3: Genetic probabilities comparison

As it can be seen, we improve the quality of the solutions as we increase the population size. This allow us to diversify more, since we have more possible solutions.
Furthermore, this improvement in the quality of the solutions can be caused by an increase in the number of generations, for the same reason, as more iterations, the algorithm can explore a bigger portion of the search space.

As a supplement to the previous data, the table 5 shows the cost difference between the optimal solution and best obtained solution with each combination. We can see which combinations achieve the optimal solution, and we can see the quality of the solutions in the case of not finding the optimal solution.

The image 4 shows this comparison in a graphical way.

We can observe that lower population size values cannot achieve the optimal solution, in this case, only using a bigger number of generations. For the number of generations, in the same way, lower values leads to a worst perfomance. We can see that using a population size of 50 individuals, we

Population Size	Maximum generations	CostDifference
10	10	13
10	50	11
10	**100**	**0**
25	10	2
25	50	2
25	100	4
50	**10**	**0**
50	50	1
50	**100**	**0**

Table 4: Cost differences by hyperparameter

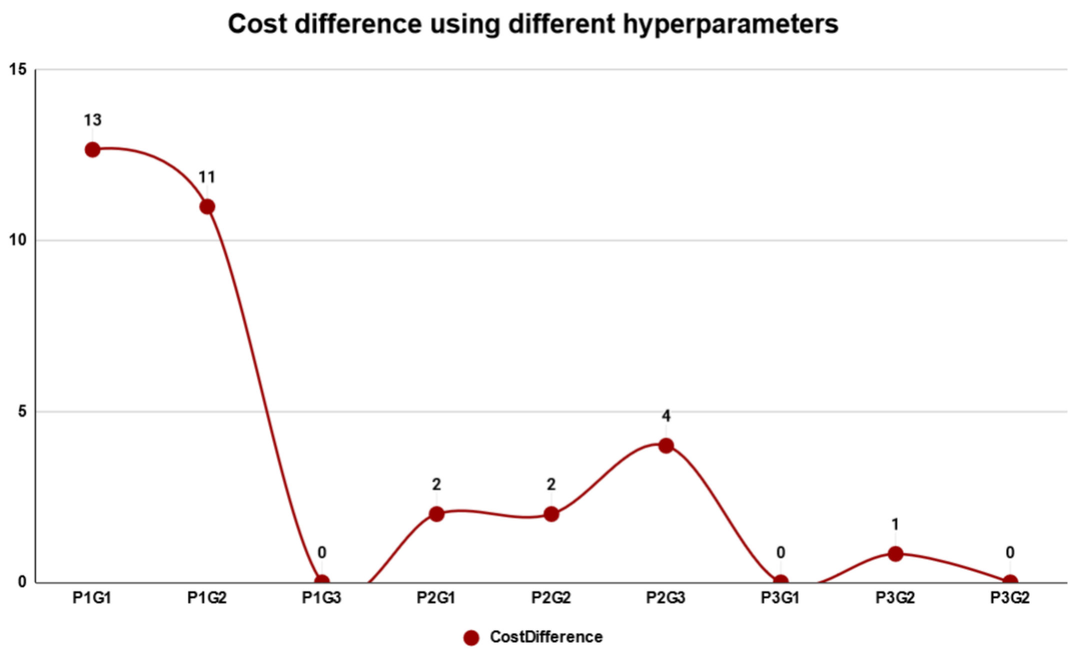

Figure 4: Genetic probabilities comparison

easily achieve the optimal solution, and when not, we get close to it.

1.3 Modify the code to include elitism. The best solution is kept and never lost. This is the solution that will be returned once the algorithm ends. Have you managed to improve? Why?

Since we have already the best combination of hyperparameters, we only need to test the importance of the elitism in the average accuracy. These result can be seen in the table 5.

Elitism	**Accuracy**
False	72 %
True	**78 %**

Table 5: Genetic probabilities comparison

As it is expected, using elitism significantly improves the performance of the algorithm. This is easily explained, because by applying elitism and always keeping the best result so far, when the algorithm diversifies the possible solutions and moves away from the current best solution, we won't mind being trapped in a local optima.
This will also allow us to diversify the solutions, allowing us to find new and possibly better solutions.

1.4 We have started so far from valid solutions when starting the algorithm. Change it so that any solution can be generated (invalid ones are possible). Does it affect the final performance?

For this analysis, we will study the results obtained with every combination of hyperparameters in every dataset, just comparing the performance evaluating the solutions or not.

We can see this comparison in the table 6

Solution Evaluation	**Accuracy**
False	75.925 %
True	**77.777 %**
True and False included	76.851 %

Table 6: Solution Evaluation effect

Even if we generate inavlid solutions in the first generation, as the algorithm converges, we will reach similar results. The only difference is that the algorithm will need more generations / iterations to converge into valid solutions, but once we are working with valid solutions, the solutions will converge in the same way.

1.5 Brief Conclusion

In previous questions, we have already analysed the effect of hyperparameters in the performance of the dataset, based in the dataset **7a.txt**. We can also compare the average performance of the algorithm per dataset. We can see this comparison in the table 7.

Dataset	**Optimal Solution**	**Solution Found (Average)**
5.txt	51	50.52777778
6.txt	150	146.1018519
7a.txt	107	104.5277778
7b.txt	1735	1718.842593
8.txt	837	825.1251852
10.txt	309	270.1481481
15.txt	1458	1435.351852
24.txt	13549094	12918749.74

Table 7: Performance per dataset

In the same way, the image 5 shows this comparison in a graphical way.

As expected with this analysis, with bigger datasets, which means bigger search spaces, the performance is worse, since the algorithm can get lost while diversifying solutions. Even so, we observe that the obtained results are close in average to the optimal solution, so we can affirm that the algorithm provides very good results, even if it not finds the optimal solutions.

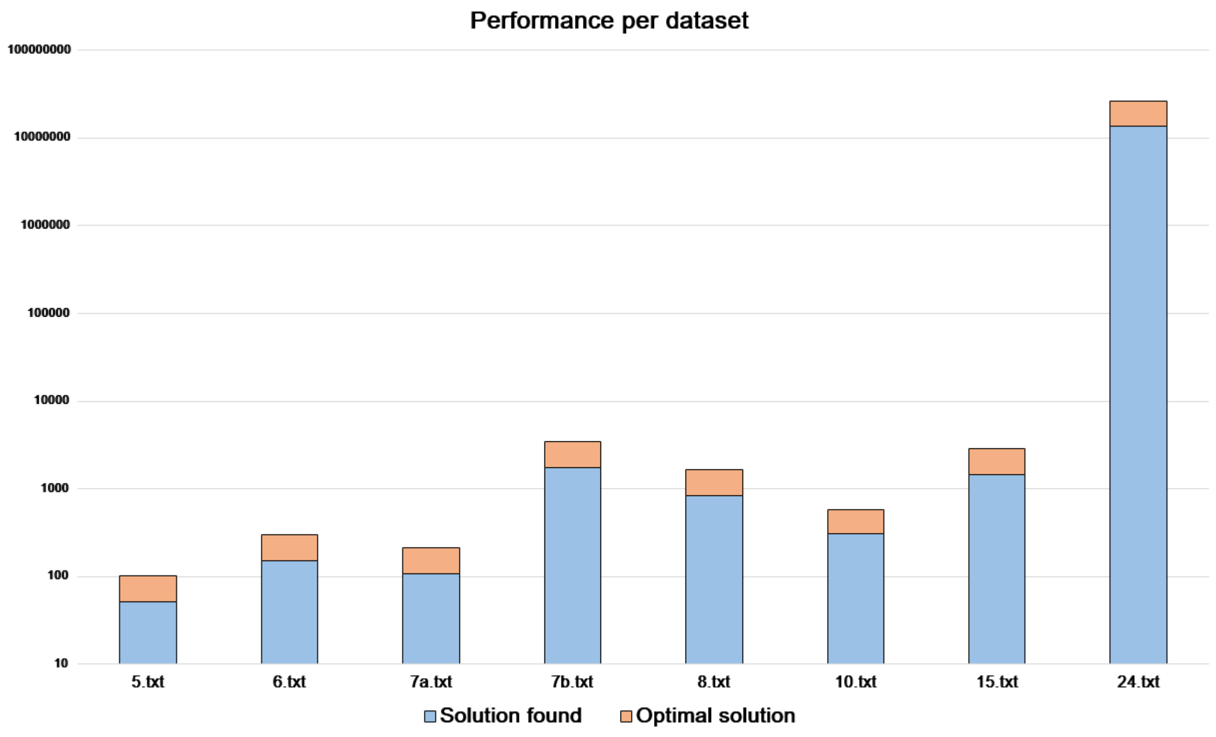

Figure 5: Performance per dataset