Assignment 2

Ventura Lucena Martínez

April, 2021

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In computer science and operations research, a genetic algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover and selection.

In this report we are applying this metaheuristic to a well-known problem: Knapsack problem [3]. The evaluation has been carried out according to a series of parameters, which have been modified to observe how the algorithm behaves, in such a way that it could be possible to identify which configuration obtained the best results. The previous parameters are the following:

- **maxWeight:** maximum storage capacity.
- **nSolutions:** number of solutions in a population.
- **maxGenerations:** maximum number of population generation.
- **k:** tournament selector. Selection of k individuals from a population.
- **cProb:** probability of crossing one individual with another.
- **mProb:** probability of mutating one individual with another.

In addition, the results have been obtained from two points of view according to the type of coding, considering both binary and integer encoding.

Chapter 2

Crossover and Mutation

2.1 Crossover in Genetic Algorithms

Crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. Crossover is sexual reproduction. Two strings are picked from the mating pool at random to crossover in order to produce superior offspring. The method chosen depends on the encoding method.

2.1.1 Types of crossover

Single Point Crossover

A crossover point on the parent organism string is selected. All data beyond that point in the organism string is swapped between the two parent organisms. Strings are characterized by positional bias. This is the method used in this study.

$$Parent_1 = [00|110] \tag{2.1}$$

$$Parent_2 = [10|011] \tag{2.2}$$

$$Child_1 = [00|011] \tag{2.3}$$

$$Child_2 = [10|110] \tag{2.4}$$

Two-Point Crossover

This is a specific case of a N-point Crossover technique. Two random points are chosen on the individual chromosomes (strings) and the genetic material is exchanged at these points.

$$Parent_1 = [00|11|0] \tag{2.5}$$

$$Parent_2 = [10|01|1] \tag{2.6}$$

$$Child_1 = [00|01|0] \tag{2.7}$$

$$Child_2 = [10|11|1] \tag{2.8}$$

Uniform Crossover

Each gene (bit in a binary codification) is selected randomly from one of the corresponding genes of the parent chromosomes.

$$Parent_1 = [00000] \tag{2.9}$$

$$Parent_2 = [11111] \tag{2.10}$$

$$Child_1 = [01101] \tag{2.11}$$

$$Child_2 = [10010] \tag{2.12}$$

2.2 Mutation in Genetic Algorithms

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to a better solution by using mutation. Mutation occurs during evolution according to a user-definable mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

2.2.1 Types of mutation

Some of the mutations that we can find are:

Bit flip mutation

The mutation of bit strings ensue through bit flips at random positions. For instance:

$$[00110] \rightarrow [10110] \tag{2.13}$$

The probability of a mutation of a bit is $1/l$, where "l" is the length of the vector. Thus, a mutation rate of 1 per mutation and individual selected for mutation is reached. Also, we consider to define a static value in range [0.01 - 0.2].

<u>This is the method used in this study</u>.

Random resetting mutation

In random resetting mutation, we select one or more genes (array indices) and replace their values with another random value from their given ranges. For instance, range [0-9]:

$$[73182] \rightarrow [03162] \tag{2.14}$$

Swap mutation

In swap mutation we select two genes from our chromosome and interchange their values.

$$[73182] \rightarrow [83172] \tag{2.15}$$

Chapter 3

Types of encoding

3.1 Binary encoding

On the first hand, we are going to work with a binary representation of the knapsack problem. The notation used is as follow:

$$[[0, 0, 1, 1, 0], 620] \tag{3.1}$$

It indicates that object in position 2 and 3 has been chosen. The last value indicates the profit obtained by the selection of these materials.

On the next subsections we are going to see how the algorithm behaves, first of all, with the default parameters; and then modifying them so that, we can see which configuration works better.

3.1.1 Default dataset

$$Weights = [34, 45, 14, 76, 32] \tag{3.2}$$

$$Prices = [340, 210, 87, 533, 112] \tag{3.3}$$

With this first example, we are explaining the methodology followed when collecting the data and its evaluating:

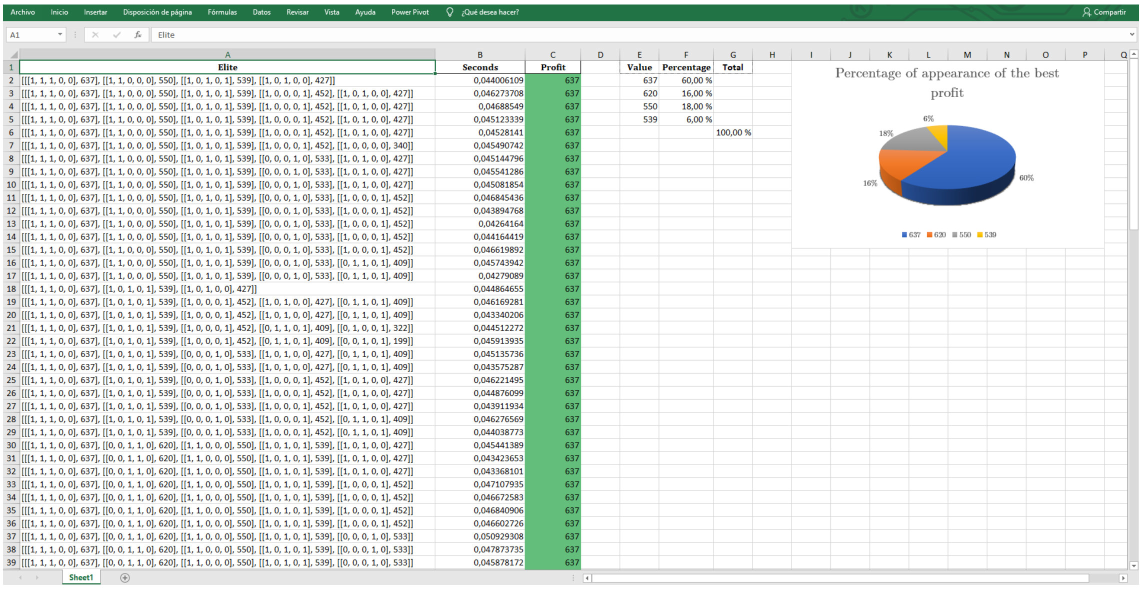
1. Execution of the algorithm (100 times).
2. Save the data in Excel file format. For instance, "100_20_100_3_07_01_default.xlsx". The file name follows this notation:
   (a) maxWeight = 100.
   (b) nSolutions = 20.
   (c) maxGenerations = 100.
   (d) k = 3.
   (e) cProb = 0.7.
   (f) mProb = 0.1.
   (g) Dataset used, either default or created (default/dataset_x).
3. Order the data.
4. Comprehend the results obtained.

Default parameters

Parameter	Value
maxWeight	100
nSolutions	20
maxGenerations	100
k	3
cProb	0.7
mProb	0.1

Table 3.1: Default dataset with default parameters.

Once we execute the algorithm, we obtain the following results[1] (already sorted):



	Elite	Seconds	Profit
2	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 1, 0, 0], 427]]	0,044006109	637
3	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452], [[1, 0, 1, 0, 0], 427]]	0,046273708	637
4	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452], [[1, 0, 1, 0, 0], 427]]	0,04688549	637
5	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452], [[1, 0, 1, 0, 0], 427]]	0,045123339	637
6	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452], [[1, 0, 1, 0, 0], 427]]	0,04528141	637
7	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452], [[1, 0, 0, 0, 0], 340]]	0,045490742	637
8	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 1, 0, 0], 427]]	0,045144796	637
9	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 1, 0, 0], 427]]	0,045541286	637
10	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 1, 0, 0], 427]]	0,045081854	637
11	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 0, 0, 1], 452]]	0,046845436	637
12	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 0, 0, 1], 452]]	0,043894768	637
13	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 0, 0, 1], 452]]	0,04264164	637
14	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 0, 0, 1], 452]]	0,044164419	637
15	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 0, 0, 1], 452]]	0,046619892	637
16	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[0, 1, 1, 0, 1], 409]]	0,045743942	637
17	[[[1, 1, 1, 0, 0], 637], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[0, 1, 1, 0, 1], 409]]	0,04279089	637
18	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[1, 0, 1, 0, 0], 427]]	0,044864655	637
19	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452], [[1, 0, 1, 0, 0], 427], [[0, 1, 1, 0, 1], 409]]	0,046169281	637
20	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452], [[1, 0, 1, 0, 0], 427], [[0, 1, 1, 0, 1], 409]]	0,043340206	637
21	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452], [[0, 1, 1, 0, 1], 409], [[0, 1, 0, 0, 1], 322]]	0,044512272	637
22	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452], [[0, 1, 1, 0, 1], 409], [[0, 0, 1, 0, 1], 199]]	0,045913935	637
23	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 1, 0, 0], 427], [[0, 1, 1, 0, 1], 409]]	0,045135736	637
24	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 1, 0, 0], 427], [[0, 1, 1, 0, 1], 409]]	0,043575287	637
25	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 0, 0, 1], 452], [[1, 0, 1, 0, 0], 427]]	0,046221495	637
26	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 0, 0, 1], 452], [[1, 0, 1, 0, 0], 427]]	0,044876099	637
27	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 0, 0, 1], 452], [[1, 0, 1, 0, 0], 427]]	0,043911934	637
28	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 0, 0, 1], 452], [[0, 1, 1, 0, 1], 409]]	0,046276569	637
29	[[[1, 1, 1, 0, 0], 637], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533], [[1, 0, 0, 0, 1], 452], [[0, 1, 1, 0, 1], 409]]	0,044038773	637
30	[[[1, 1, 1, 0, 0], 637], [[0, 0, 1, 1, 0], 620], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 1, 0, 0], 427]]	0,045441389	637
31	[[[1, 1, 1, 0, 0], 637], [[0, 0, 1, 1, 0], 620], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 1, 0, 0], 427]]	0,043423653	637
32	[[[1, 1, 1, 0, 0], 637], [[0, 0, 1, 1, 0], 620], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 1, 0, 0], 427]]	0,043368101	637
33	[[[1, 1, 1, 0, 0], 637], [[0, 0, 1, 1, 0], 620], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452]]	0,047107935	637
34	[[[1, 1, 1, 0, 0], 637], [[0, 0, 1, 1, 0], 620], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452]]	0,046672583	637
35	[[[1, 1, 1, 0, 0], 637], [[0, 0, 1, 1, 0], 620], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452]]	0,046840906	637
36	[[[1, 1, 1, 0, 0], 637], [[0, 0, 1, 1, 0], 620], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[1, 0, 0, 0, 1], 452]]	0,046602726	637
37	[[[1, 1, 1, 0, 0], 637], [[0, 0, 1, 1, 0], 620], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533]]	0,050929308	637
38	[[[1, 1, 1, 0, 0], 637], [[0, 0, 1, 1, 0], 620], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533]]	0,047873735	637
39	[[[1, 1, 1, 0, 0], 637], [[0, 0, 1, 1, 0], 620], [[1, 1, 0, 0, 0], 550], [[1, 0, 1, 0, 1], 539], [[0, 0, 0, 1, 0], 533]]	0,045878172	637

Value	Percentage	Total
637	60,00 %	
620	16,00 %	
550	18,00 %	
539	6,00 %	
		100,00 %

Figure 3.1: Default dataset execution with default parameters (Excel view).

We can see 3 different columns:

1. **Elite:** represents the best results (unique parents) obtained in each iteration of the algorithm.
2. **Seconds:** time spent in each iteration.
3. **Profit:** value of the highest profit obtained in that iteration.

[1] File available in the ".zip" provided: /binary_encoding_data/100_20_100_3_07_01_default.xlsx

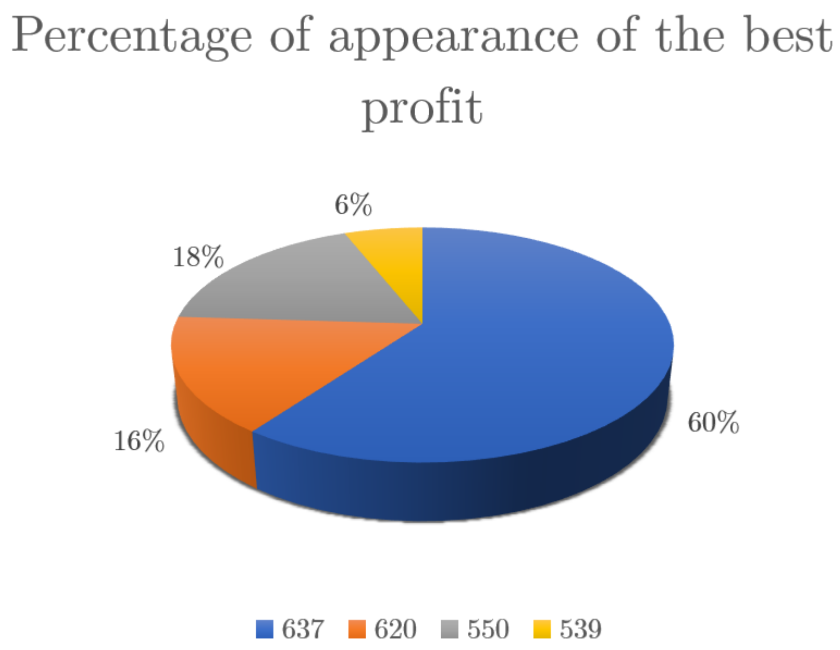


Figure 3.2: Default dataset execution with default parameters.

We can observe that the 5 best different values obtained in a total of 100 iterations are the ones present on the last figure. Let's see what information we get by altering the parameters in "Modification 1", where the only same parameter is the maximum weight:

Modification 1

Parameter	Value
maxWeight	100
nSolutions	50
maxGenerations	300
k	5
cProb	0.9
mProb	0.2

Table 3.2: Default dataset with modification 1 of the parameters.

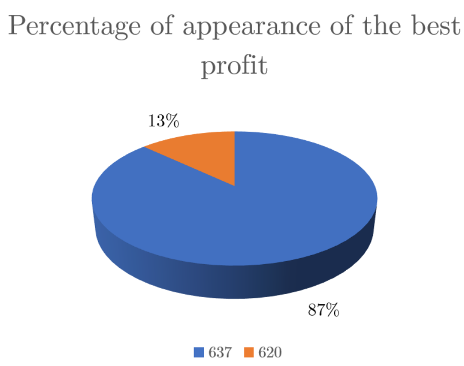


Figure 3.3: Default dataset execution with first modification of parameters.

As we can see, we have increased the frequency of occurrence of the best result, from 60% to 87%, which is a big amount. We can see that the increase in this percentage has been at the cost of time:

	Time (s)
Default	0,045128771
Modification 1	0,384681329

Table 3.3: Timetable for best result of default dataset with default parameters and modification 1.

Modification 2

In this part, we have increased the maximum weight, so that, it is no longer 100, but 200:

Parameter	**Value**
maxWeight	200
nSolutions	20
maxGenerations	100
k	3
cProb	0.7
mProb	0.1

Table 3.4: Default dataset with modification 2.1 of the parameters.

Parameter	**Value**
maxWeight	200
nSolutions	50
maxGenerations	300
k	5
cProb	0.9
mProb	0.2

Table 3.5: Default dataset with modification 2.2 of the parameters.

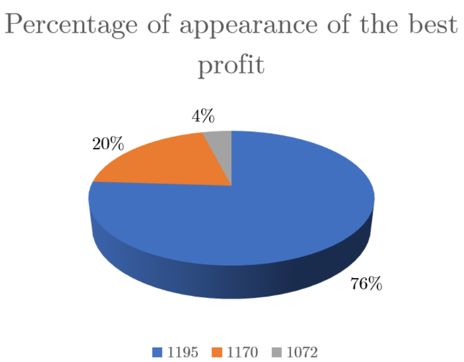


Figure 3.4: Default dataset modification 2.1.

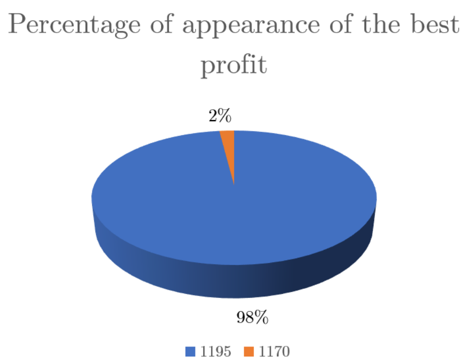


Figure 3.5: Default dataset modification 2.2.

	Time (s)
Modification 2.1	0,045870163
Modification 2.2	0,400479784

Table 3.6: Timetable for best result of default dataset in modification 2.1 and 2.2.

Notice that it has the same effect as in the previous example. To conclude with the initial dataset, we will observe the latest changes by increasing the maximum weight again, from 200 to 300:

Modification 3

Parameter	Value
maxWeight	300
nSolutions	20
maxGenerations	100
k	3
cProb	0.7
mProb	0.1

Table 3.7: Default dataset with modification 3.1 of the parameters.

Parameter	Value
maxWeight	300
nSolutions	50
maxGenerations	300
k	5
cProb	0.9
mProb	0.2

Table 3.8: Default dataset with modification 3.2 of the parameters.

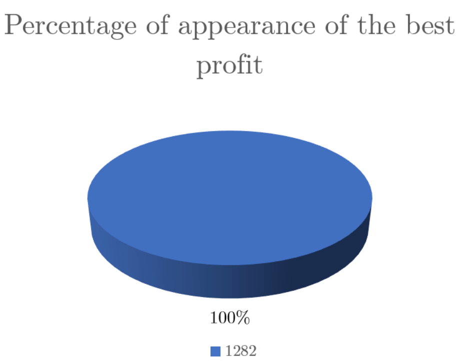


Figure 3.6: Default dataset modification 3.1.

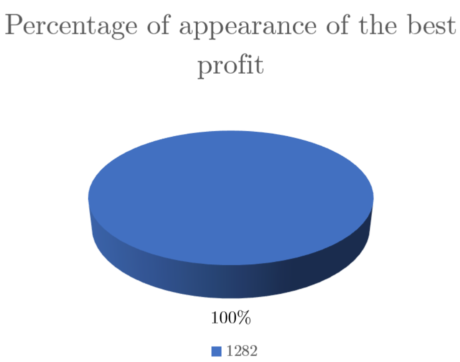


Figure 3.7: Default dataset modification 3.2.

	Time (s)
Modification 2.1	0,045621877
Modification 2.2	0,409299922

Table 3.9: Timetable for best result of default dataset in modification 3.1 and 3.2.

In the latter case, it seems that both options always achieve the best value, regardless of the parameters used.

3.2 Integer encoding

On the second part, we are going to work with an integer representation. In this case, the elements are not represented with binary notation (0, 1), but with integer notation, specifically in range [0-5][2]. For instance:

$$[[5, 3, 3, 0, 4], 3039] \tag{3.4}$$

The interpretation is analogous to the view in binary encoding.

3.2.1 Default parameters

Parameter	Value
maxWeight	100
nSolutions	20
maxGenerations	100
k	3
cProb	0.7
mProb	0.1

Table 3.10: Default dataset with default parameters.

Parameter	Value
maxWeight	100
nSolutions	50
maxGenerations	300
k	5
cProb	0.9
mProb	0.2

Table 3.11: Default dataset modification 1.

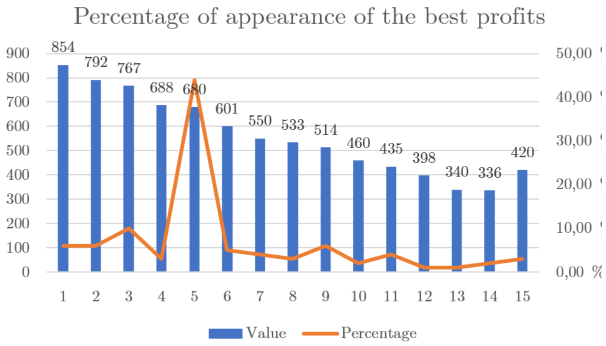


Figure 3.8: Default dataset execution with default parameters.

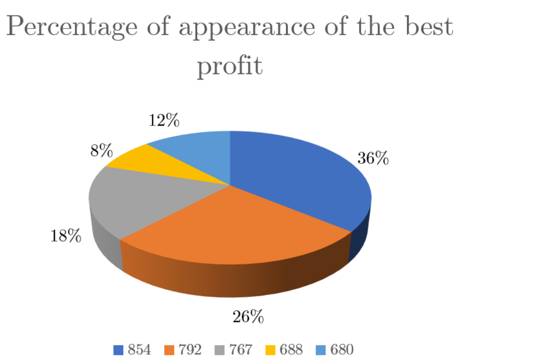


Figure 3.9: Default dataset modification 1.

	Time (s)
Default	0,041646097
Modification 1	0,41397121

Table 3.12: Timetable for best result of default dataset with default parameters and modification1.

Modification 2

In this part, we have increased the maximum weight, so that, it is no longer 100, but 200:

[2]This range has been selected to avoid that, if an item is over weighted and is selected several times, finally a null result is left in a knapsack with a limit weight.

Parameter	Value
maxWeight	200
nSolutions	20
maxGenerations	100
k	3
cProb	0.7
mProb	0.1

Table 3.13: Default dataset with modification 2.1 of the parameters.

Parameter	Value
maxWeight	200
nSolutions	50
maxGenerations	300
k	5
cProb	0.9
mProb	0.2

Table 3.14: Default dataset with modification 2.2 of the parameters.

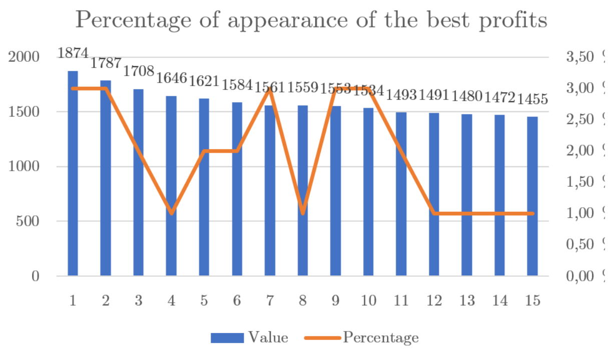


Figure 3.10: Default dataset modification 2.1 (First 15 of 65).

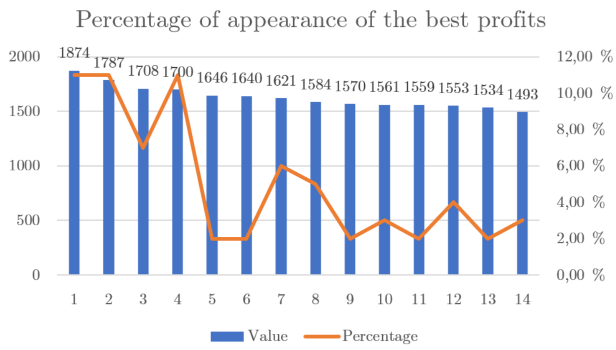


Figure 3.11: Default dataset modification 2.2 (First 15 of 35).

	Time (s)
Modification 2.1	0,046114445
Modification 2.2	0,408910188

Table 3.15: Timetable for best result of default dataset in modification 2.1 and 2.2.

Modification 3

Parameter	Value
maxWeight	300
nSolutions	20
maxGenerations	100
k	3
cProb	0.7
mProb	0.1

Table 3.16: Default dataset with modification 3.1 of the parameters.

Parameter	Value
maxWeight	300
nSolutions	50
maxGenerations	300
k	5
cProb	0.9
mProb	0.2

Table 3.17: Default dataset with modification 3.2 of the parameters.

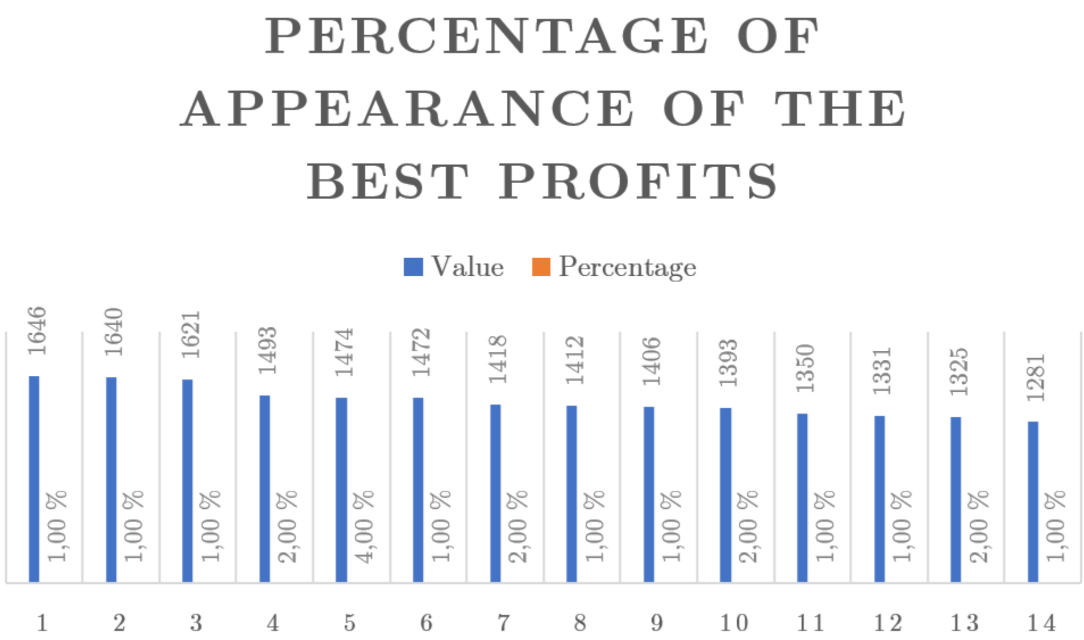


Figure 3.12: Default dataset modification 3.1 (First 15 of 65).

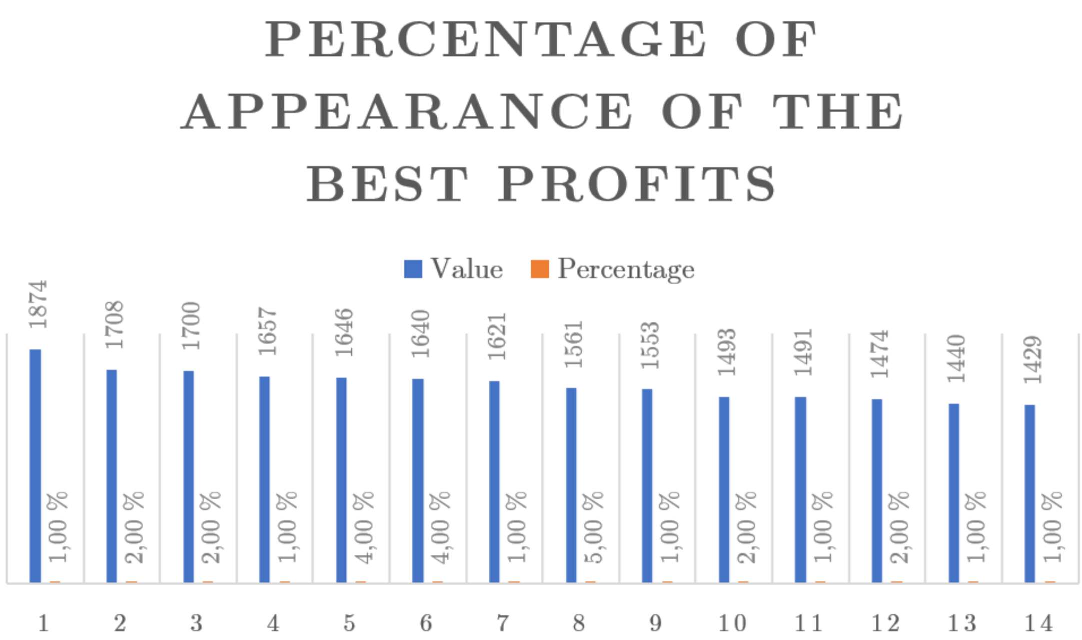


Figure 3.13: Default dataset modification 3.2 (First 15 of 35).

	Time (s)	**Best value**
Modification 3.1	0,04670167	1646
Modification 3.2	0,417586565	1874

Table 3.18: Timetable for best result of default dataset in modification 3.1 and 3.2.

Chapter 4

Conclusions

The difference in the number of solutions between both types of coding is very remarkable: in integer coding it increases considerably. In addition, the time spent in the integer encoding execution is also higher than in binary encoding, almost 10 times higher.

However, the time wasted in execution is not the only thing remarkable; the probability to reach to the best solution increases as the value of the parameters also increases. We can see that modification 3 of all, or almost all experiments obtains the best solution in terms of profit, but at the cost of time.

Appendix A

"Invalid/Forbidden" solutions

So far, we have tested what happens with valid solutions. In this section we have removed the maximum weight, so that, any solution can be generated. These are the results:

A.1 Binary encoding

Parameter	Value
maxWeight	-
nSolutions	20
maxGenerations	100
k	3
cProb	0.7
mProb	0.1

Table A.1: Default dataset with modification 1 of the parameters.

Parameter	Value
maxWeight	-
nSolutions	50
maxGenerations	300
k	5
cProb	0.9
mProb	0.2

Table A.2: Default dataset with modification 2 of the parameters.

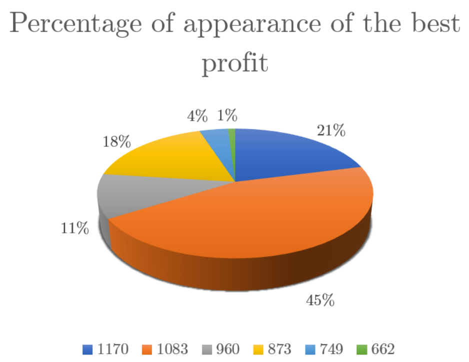


Figure A.1: Default dataset modification 1.

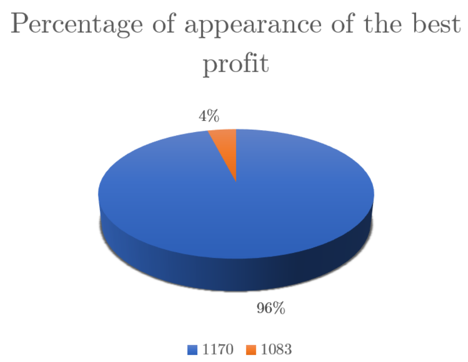


Figure A.2: Default dataset modification 2.

	Time (s)
Modification 1	0,045153357
Modification 2	0,389577416

Table A.3: Timetable for best result of default dataset in modification 1 and 2.

The data obtained seems to be quite similar to the one collected in the previous 3.1 section. It is remarkable that the best solution obtained is not better that the one in section 3.1.1, even though we have no knapsack capacity limit. Times were again quite similar.

A.2 Integer encoding

Parameter	Value
maxWeight	-
nSolutions	20
maxGenerations	100
k	3
cProb	0.7
mProb	0.1

Table A.4: Default dataset with modification 1 of the parameters.

Parameter	Value
maxWeight	-
nSolutions	50
maxGenerations	300
k	5
cProb	0.9
mProb	0.2

Table A.5: Default dataset with modification 2 of the parameters.

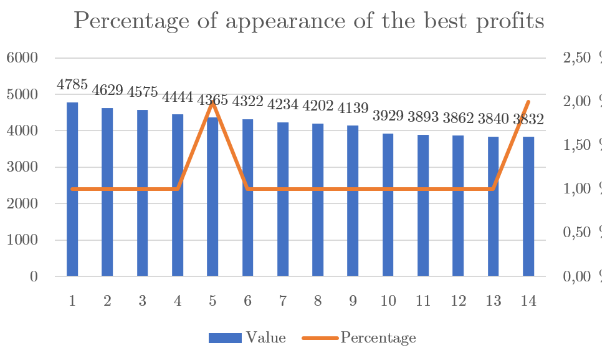


Figure A.3: Default dataset modification 1 (First 15 of 82).

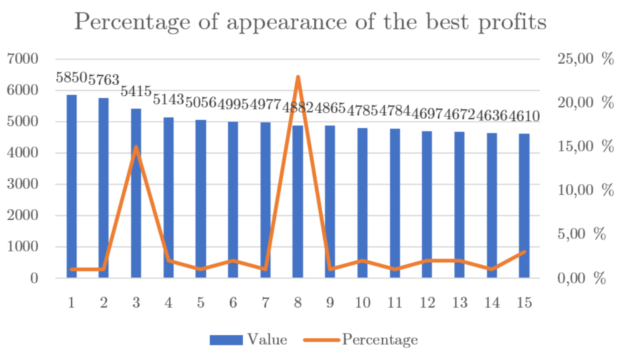


Figure A.4: Default dataset modification 2 (First 15 of 38).

	Time (s)
Modification 1	0,044138193
Modification 2	0,413819313

Table A.6: Timetable for best result of default dataset in modification 1 and 2.

On the contrary, we have exceeded, by far, the results obtained in the section 3.2.1 in this integer encoding with no weight limit.

Appendix B

Source Files

These are the source files used in this project:

- **knapsack.py:** computes the GA previously exposed.
- **knapsack_datasets_generator.py:** generates different datasets of both types, weights and prices of a given size.
- **sort_txt_data.py:** sorts the data generated with "knapsack.py" from a text file.

The following directories contain the ".xlsx" files used in all the tests carried out:

- **binary_encoding_data**.
- **integer_encoding_data**.
- **binary_encoding_data_no_constraints**.
- **integer_encoding_data_no_constraints**.

Appendix C

Extra datasets

The experiment has been carried out with a total of 5 different datasets with different modifications. This extra information is saved in its proper Excel file:

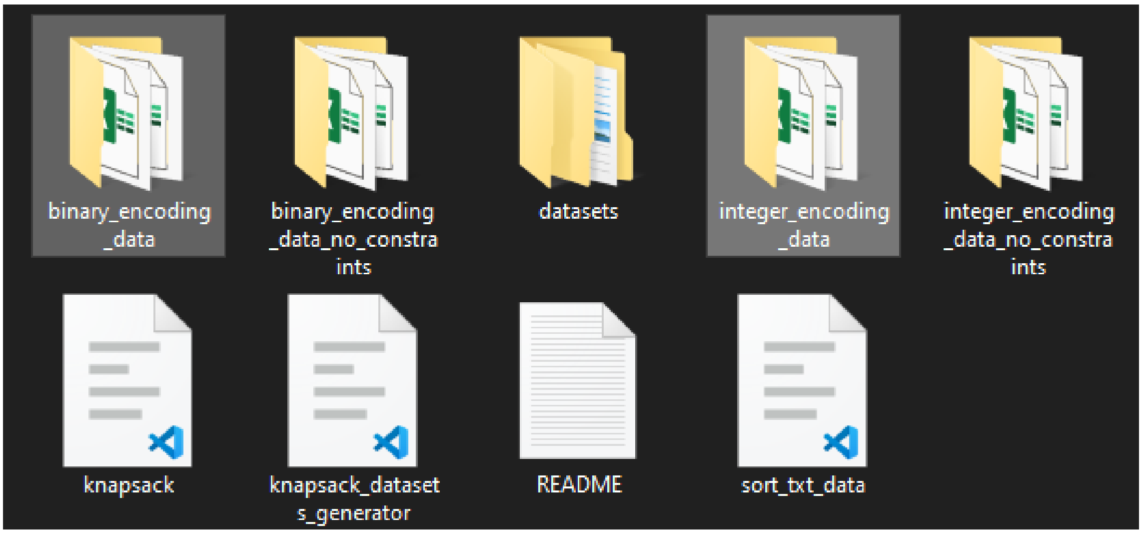
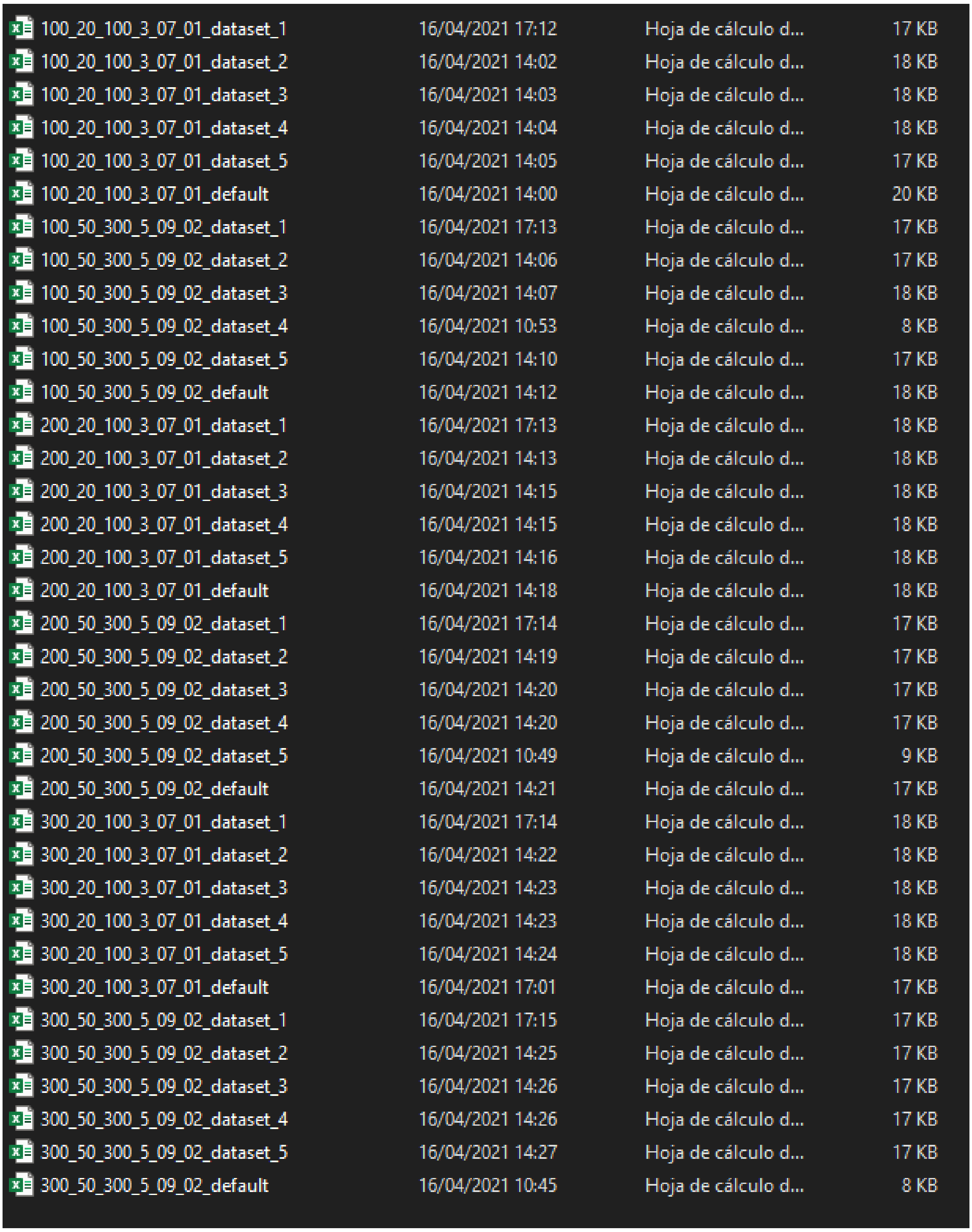
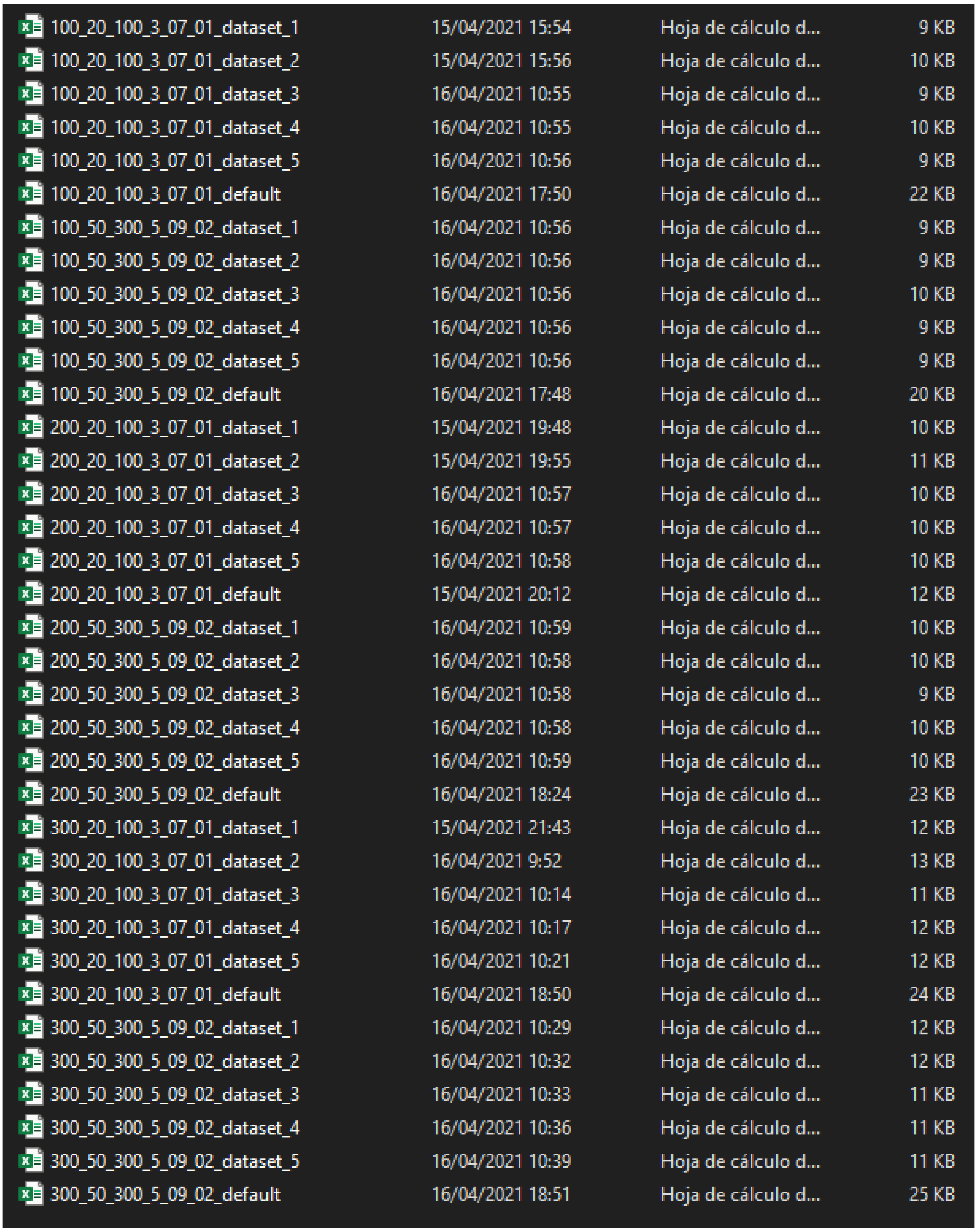


Figure C.1: Extra experiments folders.



100_20_100_3_07_01_dataset_1	16/04/2021 17:12	Hoja de cálculo d...	17 KB
100_20_100_3_07_01_dataset_2	16/04/2021 14:02	Hoja de cálculo d...	18 KB
100_20_100_3_07_01_dataset_3	16/04/2021 14:03	Hoja de cálculo d...	18 KB
100_20_100_3_07_01_dataset_4	16/04/2021 14:04	Hoja de cálculo d...	18 KB
100_20_100_3_07_01_dataset_5	16/04/2021 14:05	Hoja de cálculo d...	17 KB
100_20_100_3_07_01_default	16/04/2021 14:00	Hoja de cálculo d...	20 KB
100_50_300_5_09_02_dataset_1	16/04/2021 17:13	Hoja de cálculo d...	17 KB
100_50_300_5_09_02_dataset_2	16/04/2021 14:06	Hoja de cálculo d...	17 KB
100_50_300_5_09_02_dataset_3	16/04/2021 14:07	Hoja de cálculo d...	18 KB
100_50_300_5_09_02_dataset_4	16/04/2021 10:53	Hoja de cálculo d...	8 KB
100_50_300_5_09_02_dataset_5	16/04/2021 14:10	Hoja de cálculo d...	17 KB
100_50_300_5_09_02_default	16/04/2021 14:12	Hoja de cálculo d...	18 KB
200_20_100_3_07_01_dataset_1	16/04/2021 17:13	Hoja de cálculo d...	18 KB
200_20_100_3_07_01_dataset_2	16/04/2021 14:13	Hoja de cálculo d...	18 KB
200_20_100_3_07_01_dataset_3	16/04/2021 14:15	Hoja de cálculo d...	18 KB
200_20_100_3_07_01_dataset_4	16/04/2021 14:15	Hoja de cálculo d...	18 KB
200_20_100_3_07_01_dataset_5	16/04/2021 14:16	Hoja de cálculo d...	18 KB
200_20_100_3_07_01_default	16/04/2021 14:18	Hoja de cálculo d...	18 KB
200_50_300_5_09_02_dataset_1	16/04/2021 17:14	Hoja de cálculo d...	17 KB
200_50_300_5_09_02_dataset_2	16/04/2021 14:19	Hoja de cálculo d...	17 KB
200_50_300_5_09_02_dataset_3	16/04/2021 14:20	Hoja de cálculo d...	17 KB
200_50_300_5_09_02_dataset_4	16/04/2021 14:20	Hoja de cálculo d...	17 KB
200_50_300_5_09_02_dataset_5	16/04/2021 10:49	Hoja de cálculo d...	9 KB
200_50_300_5_09_02_default	16/04/2021 14:21	Hoja de cálculo d...	17 KB
300_20_100_3_07_01_dataset_1	16/04/2021 17:14	Hoja de cálculo d...	18 KB
300_20_100_3_07_01_dataset_2	16/04/2021 14:22	Hoja de cálculo d...	18 KB
300_20_100_3_07_01_dataset_3	16/04/2021 14:23	Hoja de cálculo d...	18 KB
300_20_100_3_07_01_dataset_4	16/04/2021 14:23	Hoja de cálculo d...	18 KB
300_20_100_3_07_01_dataset_5	16/04/2021 14:24	Hoja de cálculo d...	18 KB
300_20_100_3_07_01_default	16/04/2021 17:01	Hoja de cálculo d...	17 KB
300_50_300_5_09_02_dataset_1	16/04/2021 17:15	Hoja de cálculo d...	17 KB
300_50_300_5_09_02_dataset_2	16/04/2021 14:25	Hoja de cálculo d...	17 KB
300_50_300_5_09_02_dataset_3	16/04/2021 14:26	Hoja de cálculo d...	17 KB
300_50_300_5_09_02_dataset_4	16/04/2021 14:26	Hoja de cálculo d...	17 KB
300_50_300_5_09_02_dataset_5	16/04/2021 14:27	Hoja de cálculo d...	17 KB
300_50_300_5_09_02_default	16/04/2021 10:45	Hoja de cálculo d...	8 KB

Figure C.2: Binary encoding experiments.



100_20_100_3_07_01_dataset_1	15/04/2021 15:54	Hoja de cálculo d...	9 KB
100_20_100_3_07_01_dataset_2	15/04/2021 15:56	Hoja de cálculo d...	10 KB
100_20_100_3_07_01_dataset_3	16/04/2021 10:55	Hoja de cálculo d...	9 KB
100_20_100_3_07_01_dataset_4	16/04/2021 10:55	Hoja de cálculo d...	10 KB
100_20_100_3_07_01_dataset_5	16/04/2021 10:56	Hoja de cálculo d...	9 KB
100_20_100_3_07_01_default	16/04/2021 17:50	Hoja de cálculo d...	22 KB
100_50_300_5_09_02_dataset_1	16/04/2021 10:56	Hoja de cálculo d...	9 KB
100_50_300_5_09_02_dataset_2	16/04/2021 10:56	Hoja de cálculo d...	9 KB
100_50_300_5_09_02_dataset_3	16/04/2021 10:56	Hoja de cálculo d...	10 KB
100_50_300_5_09_02_dataset_4	16/04/2021 10:56	Hoja de cálculo d...	9 KB
100_50_300_5_09_02_dataset_5	16/04/2021 10:56	Hoja de cálculo d...	9 KB
100_50_300_5_09_02_default	16/04/2021 17:48	Hoja de cálculo d...	20 KB
200_20_100_3_07_01_dataset_1	15/04/2021 19:48	Hoja de cálculo d...	10 KB
200_20_100_3_07_01_dataset_2	15/04/2021 19:55	Hoja de cálculo d...	11 KB
200_20_100_3_07_01_dataset_3	16/04/2021 10:57	Hoja de cálculo d...	10 KB
200_20_100_3_07_01_dataset_4	16/04/2021 10:57	Hoja de cálculo d...	10 KB
200_20_100_3_07_01_dataset_5	16/04/2021 10:58	Hoja de cálculo d...	10 KB
200_20_100_3_07_01_default	15/04/2021 20:12	Hoja de cálculo d...	12 KB
200_50_300_5_09_02_dataset_1	16/04/2021 10:59	Hoja de cálculo d...	10 KB
200_50_300_5_09_02_dataset_2	16/04/2021 10:58	Hoja de cálculo d...	10 KB
200_50_300_5_09_02_dataset_3	16/04/2021 10:58	Hoja de cálculo d...	9 KB
200_50_300_5_09_02_dataset_4	16/04/2021 10:58	Hoja de cálculo d...	10 KB
200_50_300_5_09_02_dataset_5	16/04/2021 10:59	Hoja de cálculo d...	10 KB
200_50_300_5_09_02_default	16/04/2021 18:24	Hoja de cálculo d...	23 KB
300_20_100_3_07_01_dataset_1	15/04/2021 21:43	Hoja de cálculo d...	12 KB
300_20_100_3_07_01_dataset_2	16/04/2021 9:52	Hoja de cálculo d...	13 KB
300_20_100_3_07_01_dataset_3	16/04/2021 10:14	Hoja de cálculo d...	11 KB
300_20_100_3_07_01_dataset_4	16/04/2021 10:17	Hoja de cálculo d...	12 KB
300_20_100_3_07_01_dataset_5	16/04/2021 10:21	Hoja de cálculo d...	12 KB
300_20_100_3_07_01_default	16/04/2021 18:50	Hoja de cálculo d...	24 KB
300_50_300_5_09_02_dataset_1	16/04/2021 10:29	Hoja de cálculo d...	12 KB
300_50_300_5_09_02_dataset_2	16/04/2021 10:32	Hoja de cálculo d...	12 KB
300_50_300_5_09_02_dataset_3	16/04/2021 10:33	Hoja de cálculo d...	11 KB
300_50_300_5_09_02_dataset_4	16/04/2021 10:36	Hoja de cálculo d...	11 KB
300_50_300_5_09_02_dataset_5	16/04/2021 10:39	Hoja de cálculo d...	11 KB
300_50_300_5_09_02_default	16/04/2021 18:51	Hoja de cálculo d...	25 KB

Figure C.3: Integer encoding experiments.

Bibliography

[1] Moodle Universidad de Córdoba - Assignment 2 statement.

[2] Wikipedia - Genetic algorithm.

[3] Wikipedia - Knapsack problem.

[4] GeeksforGeeks - Crossover in Genetic Algorithms.

[5] GeeksforGeeks - Mutation in Genetic Algorithms.