Universidade de Coimbra
Faculdade de Ciências e Tecnologias

Mestrado em Engenharia Informática

Metodologias Experimentais em Informática
2021-2022

Exam Schedules - First Milestone

Gui Costa, 2021186342, guibscosta@hotmail.com
Tomás Ventura, 2018279147, tventura@dei.uc.pt
Gustavo Gama, 2020180035, gustavo.p.gama@gmail.com

Index

1. Problem Statement

In this paper we're going to study the performance of two backtracking algorithms that solve the exam scheduling problem. For that purpose, we will consider different scenarios and perform "EDA" to explore the data in order to draw conclusions for the algorithms (code1.c and code2.c).
So the objective of this paper is to answer the following question:

> *How does the number of exams and the probability of overlapping exams affect the performance of the exam scheduling algorithms?*

2. Identified Variables

The variables that we will have to take into account can be easily identified at first glance. Those being:

- the number of exams (variable n in gen.py);
- the number of time slots available;
- probability that each pair of exams will have a student in common (variable p in gen.py)
- CPU execution time
- the different Codes

However, simply identifying them as variables for the experiment is not accurate enough. Therefore, we will have to separate them into more distinct concepts. With that in mind, the probability that each pair of exams will have a student in common, the codes and the number of exams are classified as Independent Variables, while the number of slots available and CPU execution time are classified as a Dependent Variable.

3. Experimental Setup

3.1.Files used during the experiment

We compiled Code1.c and Code2.c using 'gcc -O3 codeX.c -o codeX.exe'.
Gen.py was modified into a class that could be called by dataManage.py, which is itself a class responsible for creating and running tests as well as storing their outputs. It also has the capability of compiling the C files. The data was stored in .dat files in a folder named Dados, the files were named using the structure p_XX_n_YY_r_ZZ_s_ww.dat where XX stands for overlap probability, YY stands for the number of exams and ZZ for the number of the run (which ranges from 0 to 30).
As for the results they are stored in a Results folder, and assigned their respective code's folder; inside each code's folder there are 3 folders, absolutes, averages and values which store times for each run, average time for each of the 30 runs and output values respectively. Additionally we have a text file called used_seeds.txt that stores all the used seeds, in ascending order of P,N and R respectively, as such, the first seed corresponds to

test p_0.0_n_5_r_0 and the second to p_0.0_n_5_r_1 and so on. Having these seeds separately helps with processing them if needed.

2.2 Setup

Given the fact that Experiments are the crucial part of this assignment, we heavily tested our program in order to achieve results as accurately as possible, in this case we increased the amount of tests used by 3.5x, thus, we decided to test the program by altering the following variables: Probability of overlap (from 0 to 1 with increments of 0.05, like so: 0.00, 0.05,0.1,etc); The number of exams remains the same (5, 10, 15, 20, 25, 30, 35, 40); We changed the seeds used so that they were different for every single test we ran.
We defined the max CPU time same as before, at 60 seconds.(of 60 seconds)
Each individual test (consisting of a Probability of overlap, a
Number of exams and a fixed max cpu time) was run 30 times, with different seeds each time, and then averaged out.

System specifications break down as follows:

- Processor: Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz
- RAM: 8,00 GB (7,89 GB Usable)
- Operating System: Windows 10 Home version 21H1, 64 bits

4. Data/measurements from experiments

4.1

In this section we present the results of the initial tests we ran, in order for us to get a first taste of what kind of performances we should expect from our future more complex tests.

P	5	10	15	20	25
0.0	0.0	0.0	3.3333333333333335e-05	3.3333333333333335e-05	0.0
0.05	0.0	0.0	0.0	6.666666666666667e-05	0.0004
0.1	0.0	0.0	0.0	0.0	0.0024000000000000002
0.15	0.0	0.0	3.3333333333333335e-05	6.666666666666667e-05	0.0021666666666666666
0.2	0.0	0.0	3.3333333333333335e-05	0.0002	0.0005666666666666668
0.25	0.0	3.3333333333333335e-05	0.0001	0.0003333333333333333	0.002800000000000001
0.3	0.0	0.0	0.0	0.0004333333333333337	0.005900000000000001
0.35	0.0	0.0	6.666666666666667e-05	0.0009666666666666667	0.012433333333333334
0.4	0.0	0.0	0.0001	0.0066	0.02026666666666667
0.45	0.0	0.0	6.666666666666667e-05	0.004533333333333334	0.10056666666666664
0.5	0.0	0.0	0.0001	0.0017333333333333335	0.05413333333333332

Table.1 Example of Code 1's average times for some Ps and Ns.

P	5	10	15	20	25
0.0	1	1	1	1	1
0.05	1	2	2	2	2
0.1	1	2	2	3	3
0.15	2	2	3	3	3
0.2	2	2	4	3	4
0.25	2	3	3	4	4
0.3	2	4	4	4	4
0.35	2	3	4	4	5
0.4	2	4	4	5	6
0.45	4	3	5	6	6
0.5	4	4	5	5	7

Table.2 Example of Code 1's values for some Ps and Ns.

4.2

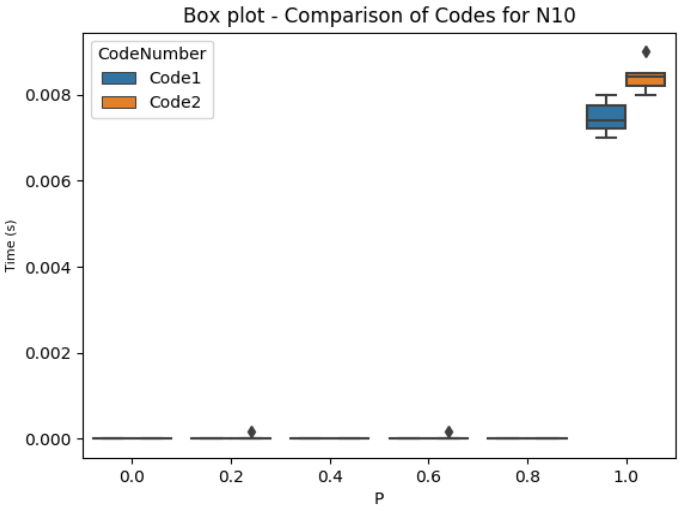
In this section we're going to perform data visualization for our results. Firstly, we're going to show the plots that describe the impact of the variable p (probability that each pair of exams will have a student in common) and then, the variable n (number of exams) on the CPU execution time for the different algorithms (Code 1-2).

In order to facilitate the comparison and the analysis of the results, we decided to choose some of the values for n and p based on what we think might be a more realistic scenario for this scheduling problem (this means a limited number of exams and overlap probability).
Scenarios:

- scenario 1: smaller number of exams(10-20).
- scenario 2: bigger number of exams(25-40).

In all of those scenarios we vary the probability p, we decided to keep the probability up to 100% to see the impact, but in some real scenarios, we can consider that probability to go up to around 40%.

a) Results scenario 1



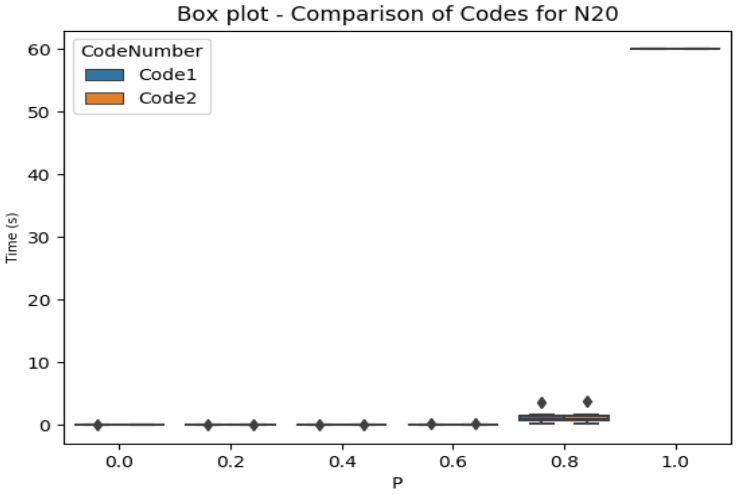
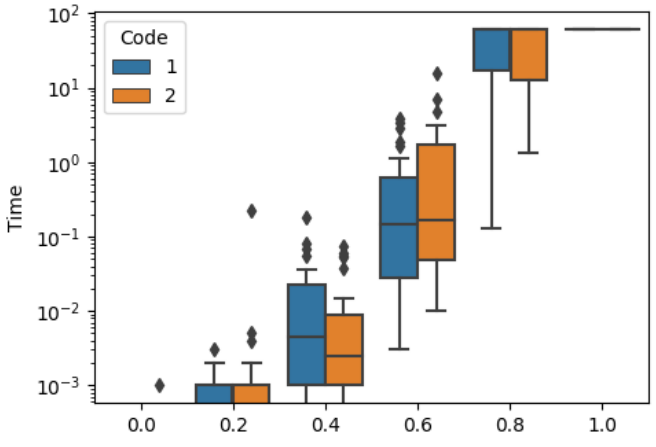


Fig.1 (left): Box plot for execution times for both codes with N=10.
Fig.2 (right): Box plot for execution times for both codes with N=20.

b) Results scenario 2



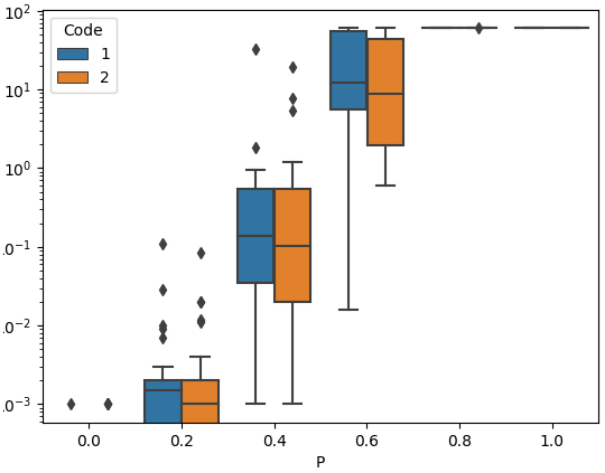
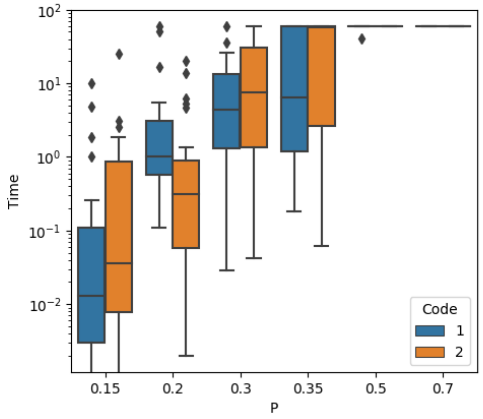


Fig.3 (left): LogScale: Box plot for execution times for both codes with N=25.
Fig.4 (right): LogScale: Box plot for execution times for both codes with N=30.



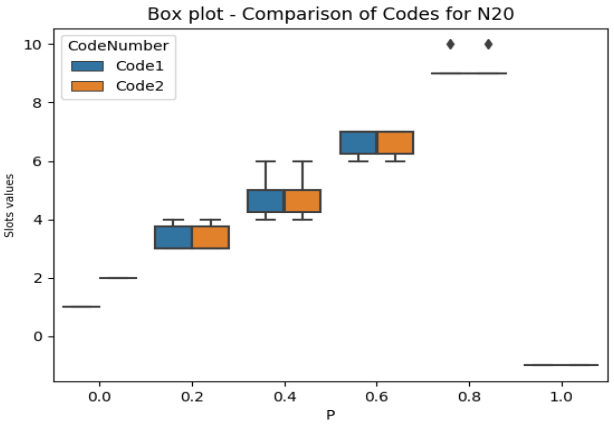


Fig.5(left): LogScale: Box plot for execution times for both codes with N=40.
Fig.6(right): Box plot for the number of Slots for both codes with N=20.

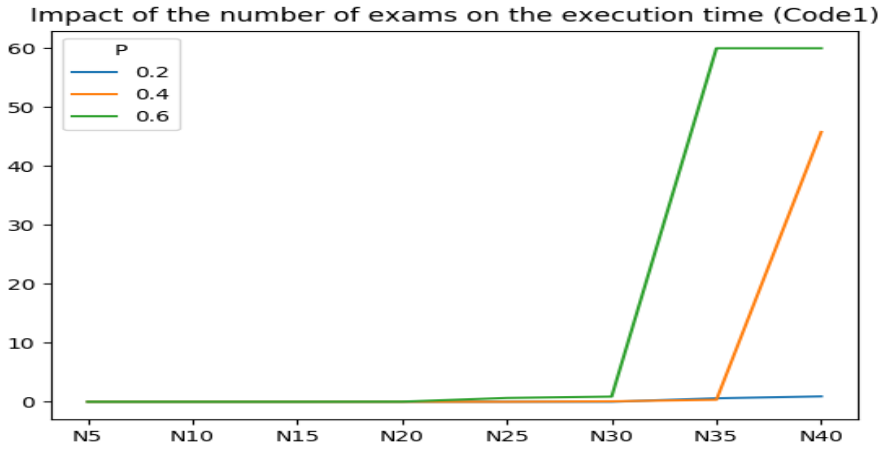


Fig.7: Plot for the impact of the number of exams on the execution time (code 1, with some fixed probabilities, we obtain a similar plot for code2…).

5. Discussion - data visualization

As we can see, the values we assign to p and n are critical when it comes to obtaining a solution. With a small number of exams, results are almost instant, both algorithms can find a solution, even though we vary the probability p. This corresponds to scenario 1 (fig.1 and fig.2).

However, the more exams there are and the higher the probability is, the more the program will struggle to find a solution within the time limit provided. Typically, when the number of exams is greater than 35 and the probability greater than 40%-60%, it becomes impossible to obtain a solution within the 60 seconds established. This corresponds to scenario 2, where we consider a greater number of exams (fig.3-5).
To confirm some of those previous results, we can decide to vary the number of exams for some fixed probabilities, we can see as we increase the number of the exams; depending on the probability value, it can affect the performance in time of the algorithm (fig.7).

Also, we can also compare the minimum number of time slots for each algorithm, this result can be seen on the figure 6. We obtain the same results for both codes, this can be explained by the fact that both algorithms implement the same strategy, they're both backtracking algorithms (one is top-down , the other is bottom-up) and end up with the same values for the slots, the only thing that changes is the execution time.

6. Regression models

In order to understand the behavior of both algorithms, we decided to apply the regression model on our data, this allows us to make predictions on data. We will evaluate the quality of the model with the coefficient of determination $R^{2.}$
This coefficient is bounded in the range [0;1]. So for a value of 1 we are in the best case scenario where the modeled values match the observed values.

Description of the parameters in the results:

- The number of exam values chosen were the following: 20 and 25.
- A variable probability like previously, going from 0 to 100%

So we're going to use the variable P in our model.

a) Linear std regression

First, we attempted to perform a standard linear model (fig.7). As expected this model doesn't match our data well, we can see it on table 1 that shows the R-squared values.

LR std. model	**Code 1 (n=20)**	**Code 2 (n=20)**	**Code 1 (n=25)**	**Code 2 (n=25)**
R-squared value	0.3288	0.3147	0.489	0.4652
A	-8.122	-7.820	-9.922	-10.639
B	28.299	27.200	41.005	43.411

Table 1: Results for the linear std model. R-squared values.

b) Exponential model

Since the previous model doesn't fit the data well (it was expected due to the nature of the data distribution). We decided to apply a transformation. Instead of the normal formula: $y = a + bx$; We applied the exponential one for the model: $ln(y) = a + bx$.

Exp. model	**Code 1 (n=20)**	**Code 2(n=20)**	**Code 1 (n=25)**	**Code 2 (n=25)**
R-squared value	0.7579	0.674	0.7177	0.9459
A	-16.936	-14.047	-12.562	-9.2797
B	23.172	18.331	21.417	15.7900

Table 2: Results for the exponential model. R-squared values.

We can see that with the transformation we obtain a better model, the coefficient of determination is closer to 1 (table 2). This suggests a strong relation with our variable p for the both scenarios (20 and 25 for the number of exams).

We can see the line that fits the model on the following figures(8-9):

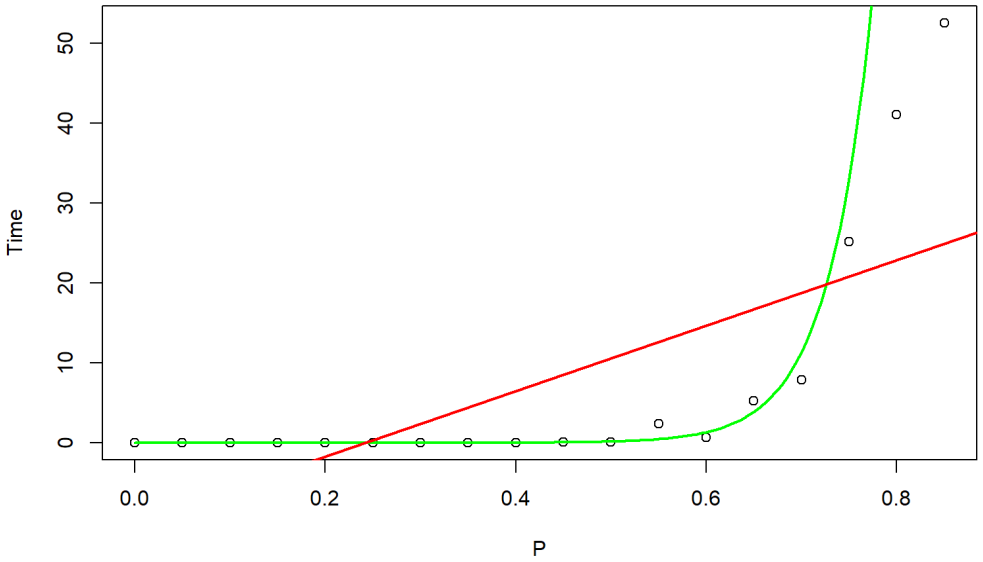


Fig.8: Comparison of the models: linear regression line (red) and transformed one (green). Time on y axis.

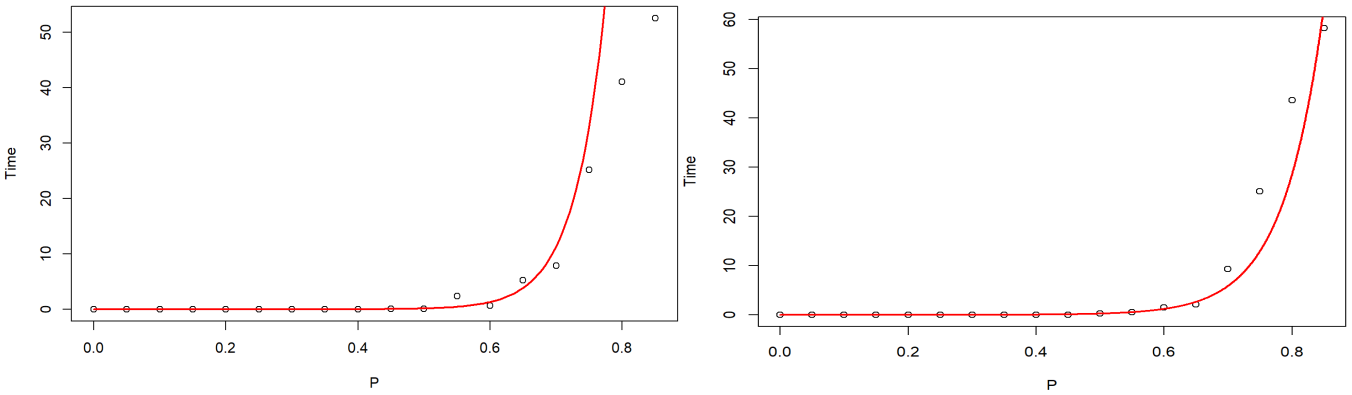


Fig.9: Code 1; example of fitting line for exp. model. (N25) ; Time on y axis.
Fig.10: Code 2; example of fitting line for exp. model. (N25) ; Time on y axis.

8) Conclusion

To conclude on this paper about EDA, we can say that the independent variables have a strong relation with the dependent variables, specially the execution time, for each code version. We used a few regression models to see this relation.
This can also be observed with the multiple plots (data visualization step), we could conclude that as the number of exams increase with the probabilities p, our execution times increase considerably, often reaching our limit of CPU time.
As for the slots values obtained, we could conclude with this experiment that both algorithms have the same performance in terms of time slots that are possible to use (they return the same value for non zero probabilities p).
So, to compare both algorithms, overall we can say that they perform the same, with a few differences concerning times, based on the data visualization, we could see that the code 2 is faster than code 1 for the second scenario (higher number of exams, specially for N30).

Annex A (Changes)

The changes made to this milestone were as follows:

1. New tests with different seeds each time (experimental setup);
2. Table with examples of the measured data, some data was omitted due to space constraints;
3. Updated tables 1 and 2 (with the new dataset)
4. Updated fig. 8 and 9 (with the new dataset and added more P values);
5. Update plots (logscales); to fix scale problems