FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Exams Scheduling Exploratory Data Analysis

University of Coimbra
PDCTI - Research Methods

Francisco Baeta, Marcio Inácio, Marco Silva
2015235815, 2021183381, 2015243074

November 9, 2021

---

1. Introduction

When solving a computational problem, it is essential to evaluate the performance of the proposed algorithms (measured mainly as the execution time) to define the problems for which they are suitable. In some cases, a few instances of the given problem may be too big to be solved in a reasonable time. To analyze how an algorithm behaves in different scenarios, it is crucial to carry out experiments using multiple settings to see the influence of different variables on execution time.

This project compares two algorithms (`code1.c` and `code2.c`) that implement two randomized backtracking algorithms to solve the exam scheduling problem. Both algorithms require three arguments: a) random seed; b) maximum CPU time in seconds to run the program; c) name of the file that contains the exam's input data. This way, to create the input data file, we use a data generator in Python that randomly generates input data that can be read by the two implementations mentioned above. Thus, we need to define the number of exams, the probability that each pair of exams will have a student in common, and the random seed for the input data generator.

Therefore, we provide an Exploratory Data Analysis (EDA), as well as a Linear Regression model, to draw preliminary conclusions about how specific factors affect the algorithms and what is the most suitable one to use in certain conditions.

2. Problem Definition

The task to be solved by the algorithms is an instance of the graph coloring problem. Thus, given input data defined by the number of exams to be scheduled and the probability of each pair having a student in common, the algorithms must find the minimum number of time slots needed to accommodate all exams to ensure that each student can take their assigned exams without overlaps. Furthermore, it is essential to measure the execution time of each algorithm to characterize its performance according to several scenarios. Thus, the problem statement for this work is:

How can the input data (number of exams, student overlapping probability, and random seed) affect the algorithms' execution time to obtain a schedule with the minimum number of time slots?

The following section provides a clear definition of the variables used to solve this problem.

2.1. Variables

The effect we want to measure is the execution time of each algorithm (`code1.c` and `code2.c`). During the EDA described in Section 3, we assume that there is a set of conditions that remain stable such as the maximum number of time-slots available for the exams scheduling (300), the CPU cut-off time for the algorithms execution (100 seconds), and the machine used to perform the EDA. Regarding the variables, we can classify them as following:

- **Dependent variables:** Execution time for each algorithm, defined as a continuous variable.
- **Independent variables:** Number of exams and random seed for the input data generator, both defined as discrete variables, and the probability for each pair of exams will have a student in common, defined

as a continuous variable.

More details about the experimentation process will be given in the following section.

2.2. Experimental Setup

As we have control over three independent variables to generate different test scenarios, it is essential to define which values will be applied during the experiments. It is also necessary to set a cut-off time limit to avoid dealing with complex cases which cannot be solved in a reasonable time. Given this, first, we define some broad variables to get an overview of system performance. These results are shown in Section 3.2. After this, we redefine the variable given the conclusions obtained, to get detailed results, shown in Section 3.3.

Thus, we defined the values of the variables as follow:

1. **Seeds:** In Section 3.2, 30 different seeds were used in order to obtain a better overview of how randomness influence the algorithms' performance. Given the results, we decide to use only 5 seeds in Section 3.3 since this input does not significantly impact the results.
2. **Probability:**In Section 3.2, the probabilities were defined in an exponential fashion as being $\{\frac{2^n}{100} : n \leq 6 \in \mathbb{N}^*\} = \{2\%, 4\%, 8\%, 16\%, 32\%, 64\%\}$. We fixed 6 probability values to reduce the number of experiments needed since it would take too much time to deal with large runs. We also expect higher probabilities to produce more complex problem instances, causing timeout executions (when the algorithm reaches the cut-off time), which do not necessarily contribute to algorithms comparisons. Thus, given the results, in Section 3.3 we double the values, using a factor of $\sqrt{2}$ instead of 2 for the geometric progression.
3. **Number of exams:** Although the algorithms were developed to accept instances containing up to 300 exams, some toy experiments showed that these inputs were too big, and both methods always resulted in a timeout. Therefore, in Section 3.2, we used the following exams number: {2, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100}. Moreover, to achieve better resolution, in Section 3.3, we increase the exams number up to 100 with a step of 4. We do not test the scenarios in which there is 0 or 1 exam, as they are trivial and no calculations would be necessary regardless of the method used.

In the next section, we present the Exploratory Data Analysis performed to test how can the independent values influence the performance of the algorithms.

3. Exploratory Data Analysis

In order to analyze how both algorithms behave under different conditions, we can take advantage of graphical visualization methods such as scatter plots, histograms, and others.

3.1. Software Development

To provide a high-level overview of the scheduling algorithms, a system was developed to, in run-time, generate the input data given the parameters discussed in Section 2.2. Therefore, the system runs each method storing their final workload (the amount of resources used during a program execution [1]). Hence, the developed system can operate autonomously, using the input data generator, run each algorithm with the generated input, and plot the results. This system was developed to run on a PC under either Windows or Linux environments.[1] The machine used for the experimentation has an AMD Ryzen 5 5600X CPU (12 core @ 3.700GHz) along with 32 GB of RAM.

3.2. Preliminary Results

This subsection represents the preliminary results obtained, the variables values utilized, and the correspondent results. These results were used to have a high-level overview of the algorithm's performance and to understand the suitable values to use in the experiments. Therefore, considering these results and analyses, we perform a detailed EDA , with more accuracy and details in Section 3.3.

3.2.1 Cut-off Time

A first observation that can be made upon the results concerns evaluating the number of executions that reached the cut-off time limit (timeout). These statistics can be seen in Figure 1. From the figure, it is possible to notice that a considerable fraction of the runs reached the cut-off time (approximately 44%). Therefore, comparing

[1]The code for all experiments and also the graphs is available at: `https://github.com/Superar/assignment-RM`.

both algorithms, we can see that there this no substantial difference between them, which means that both algorithms have the same sensitivity to cut-off time and the same complexity.

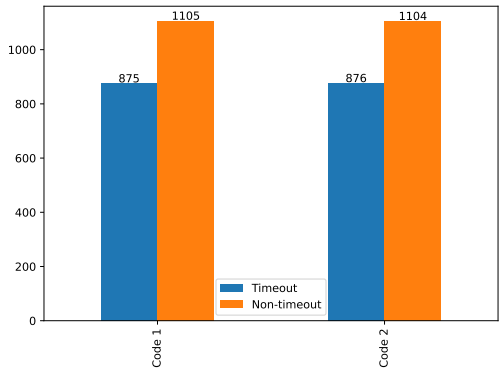


Figure 1: Cut-off time comparison

3.2.2 Algorithms Run Time

In order to analyze how the different variables discussed in Section 2.2 influence the total run time of each algorithm, we plotted the graph presented in Figure 2, which relates the average run time considering the seeds used, the number of exams, and the overlapping probability for each pair of exams. From the figure, we can observe that the run time does not rise much regardless of the probability for a small number of exams (less than 30). However, as the number of exams increases, the minimum probability value to achieve higher run times decreases, *e.g.*, for 100 exams, even a small probability of 4% already reaches the cut-off time limit.

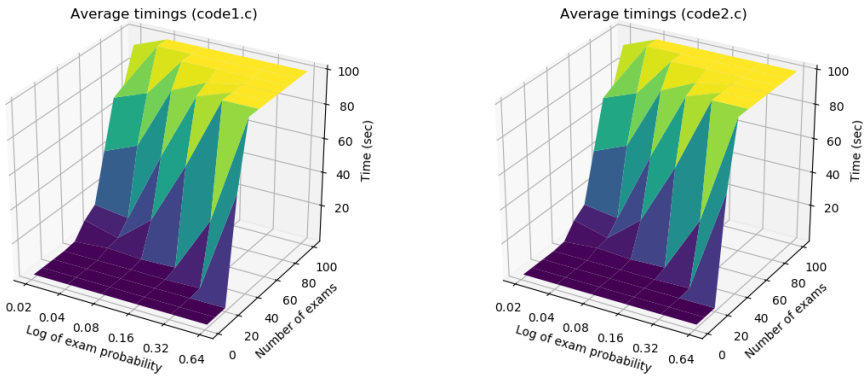


Figure 2: Algorithms run time comparison

As the graphs are based on the average run times overall seeds, we decided to create a grid plot using each experiment separately so that we can observe how the random aspect of each algorithm influences the distribution of run times. The results for the `code1.c` algorithm are presented in Figure 3.

Examining the figure, we can note that most variation due to the random seed can be perceived in the range of approximately 40 to 70 exams. When there are fewer exams to be scheduled, the algorithm is fast regardless of the seed. For larger instances of the problem, the timeout limit is reached consistently, unless dealing with a small overlapping probability as 2%. These same observations can be made for `code2.c`, as its graphs present the same behaviour. However, due to space limitations, we do not present this plotting here.

3.3. Detailed Analysis

To better understand the relationship between dependent and independent variables, and considering the results obtained in Section 3.2, we increase the granularity of our problem to understand in detail the behavior of the algorithm. Thus, we were able to double the resolution of the exam probability (using a factor of $\sqrt{2}$ instead of 2 for the geometric progression) while also testing the number of exams up to 100 with a step of 4. Nevertheless,

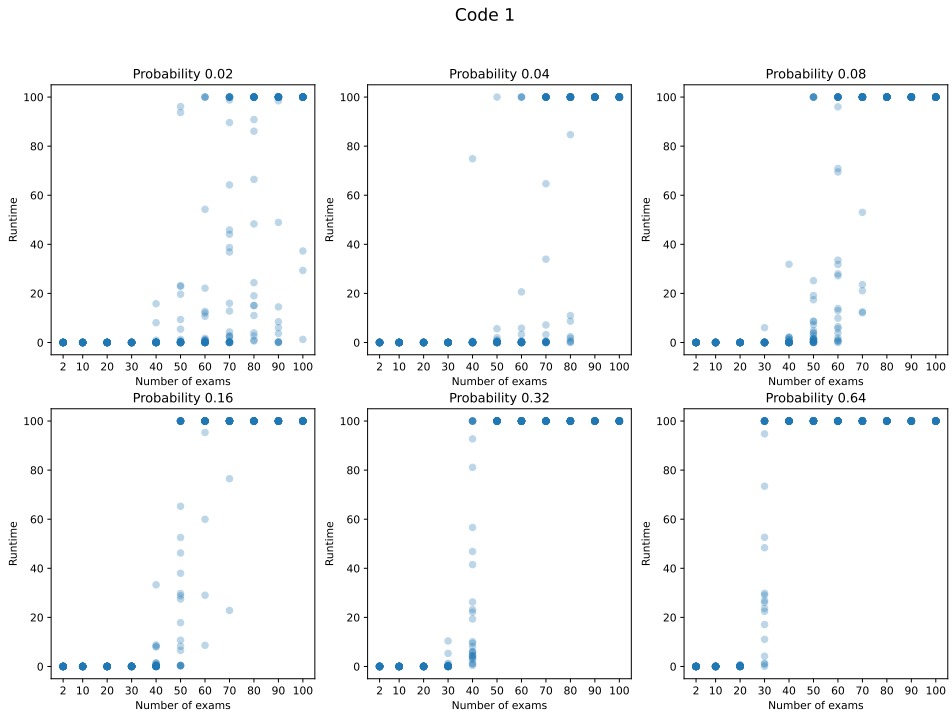
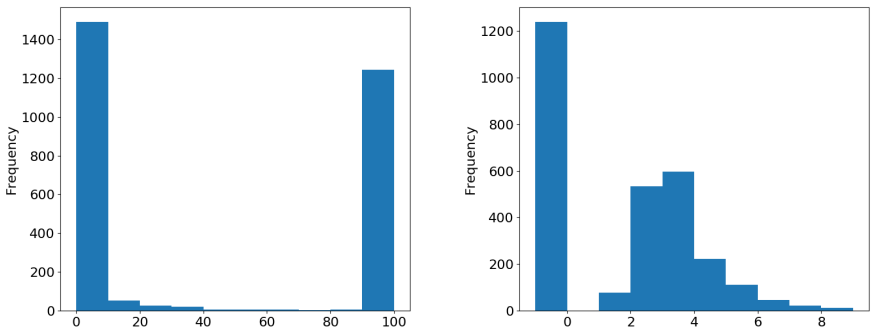


Figure 3: Seeds influence in `code1.c` run time

to counterbalance the higher computational cost, we decided to relax the need for conventional statistical significance and perform 5 seeds per run instead of the conventional 30 [3], since the results presented in Section 3.2 shows that the seeds have no significant impact on the analysis.

Figure 4 shows the histograms for the distribution run-time analysis, and the correspondent output value N. The output value N corresponds to the minimum time-slots necessary to accommodate all the exams defined in the input data, considering the probability of student overlap.



(a) Execution time (s) (b) Output values (N)

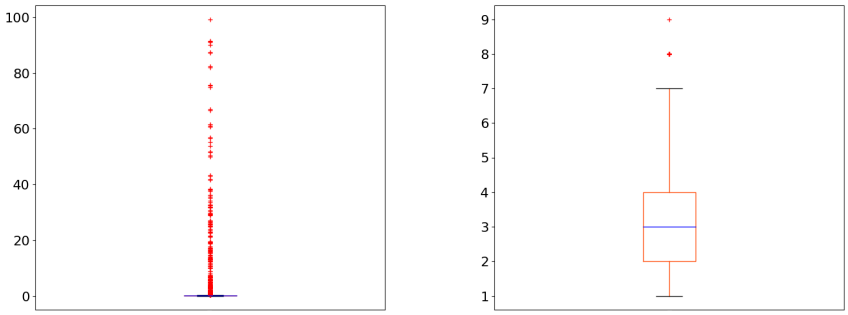
Figure 4: Histogram for time and output values distribution

From the Figure 4a, we verify that the computational times are mostly either close to 0 or reaching the cut-off threshold (100 seconds), with a few runs displaying times between 10 and 40 seconds. Overall, by removing the threshold on the cut-off time, the distribution displays exponential behavior. Generally speaking, the hypothesis that times increase exponentially with the increase of both probability and number of exams is consistent with the time complexity of the backtracking algorithms used in both `code1.c` and `code2.c`.

Regarding Figure 4b, the frequency for the least number of slots found (*i.e.* N) seems to resemble a skewed normal distribution. Again, we verify a high frequency of cut-offs that is evidenced by the values with $N = -1$.

With the high pervasiveness of cut-offs in mind, we follow this last analysis with the corresponding box-plot representations, shown in Figure 5. However, this time we have removed the samples that reached the cut-off

time from the data set. Overall, the box plot for the computational time (Figure 5a) proves rather uninsightful due to the exponential distribution of times (all quartile values are close to 0). On the other hand, the box-plot for the N values (Figure 5b) clearly displays a median of $N = 3$, with an interquartile range of $[2, 4]$. We also corroborate the aforementioned skew towards smaller values of N, with a minimum and maximum score of 1 and 7, respectively.



(a) Execution time (s) (b) Output values (N)

Figure 5: Boxplot representation for the distributions

To complement our analysis, Figure 6 shows the 3D surface plot corresponding to the time taken for each algorithm run as a function of both the output value N and the assignment probability. Similar to the graphs displayed for the preliminary experiment in Section 3.2, these surface plots encompass all samples, including cut-off times, and display the logarithm of exam probability to ease visualization. The higher granularity in the problem domains seems to make a case for the steep increase in computational time. As shown, for the relatively small probability of 0.02, we need a problem of almost 100 exams to reach the cut-off time. On the contrary, by raising the probability to 0.64, we verify that even problems with few exams (20) seem to struggle to stay below the time threshold. One glaring difference illustrated by this plot is that its surface is much more irregular, when comparing to Figure 2, leading to higher standard deviations amongst seeds (shown in Figure6b) when compared to the shown in the preliminary results. The lower number of seeds can partially explain this run for each problem configuration. Nonetheless, to determine if higher deviations and solely a product of lower statistical significance or a symptom of the problem's nature, we would need to consider more seeds, which surpasses our allocated time in the department's machine for the time being.

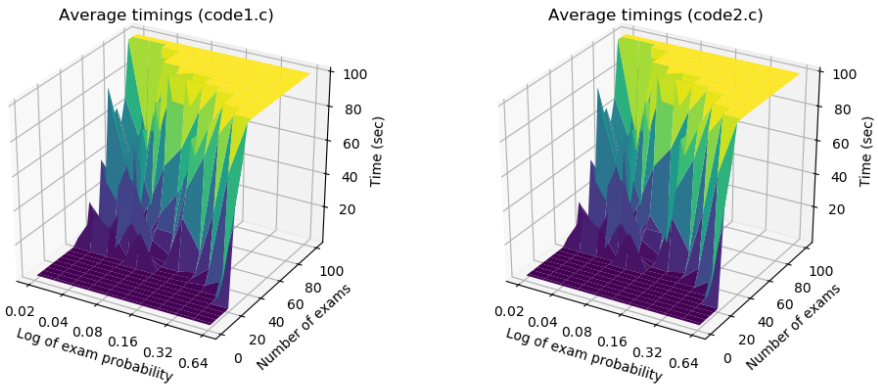
3.4. Correlation

Correlation Analysis is critical when attempting to understand the relationship between variables, and it is a required step before testing any hypothesis on your data. The correlation values are between -1 and 1. -1 denotes a completely negative correlation, which means that as one goes up, the other goes down to the same degree. Similarly, 1 represents a positive association.

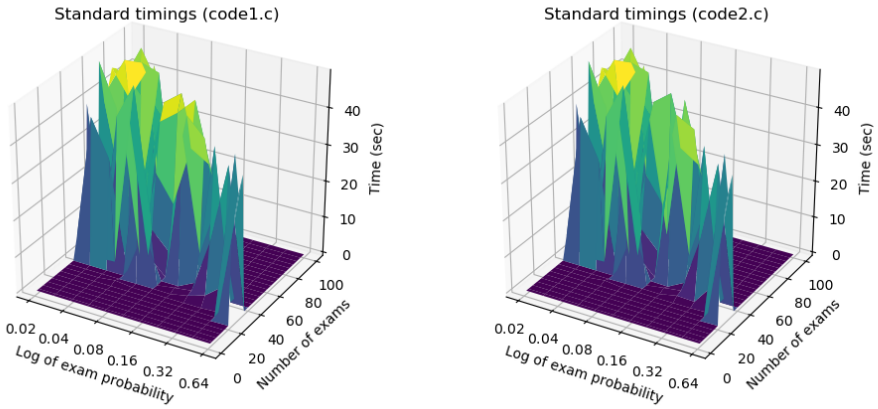
Figure 7 tries to find the correlation among all the variables together. The darker the pink color is, the higher the correlation is. From the figure, it is possible to notice that the main correlation happens between the output value (N) and the probability variable (0.6 in Figure 7a, and 0.7 in Figure 7a). This means that, for high probabilities of each pair of exams having a student in common, more time-slots will be needed to accommodate all the students (N). In the same way, the number of exams will also influence the output value, however, with less impact (0.2 for both cases).

Regarding the algorithm's run time, if we increase the number of exams, the algorithm's run-time will also increase, with a relation of 0.3 for both. The probability value does not have a significant impact in the algorithm's run-time. If we see the Figure 7a, we can notice that the is a correlation value of 0.03, and in the Figure 7b the correlation is -0.006. Although these values are close to 0, on the limit, we can state that there is a negative correlation on the second case, which means that an increase in the probability means a reduction in the `code2.c` run-time.

Therefore, we can conclude that the minimum number of time-slots necessary to accommodate all the exams without students' overlap is highly dependent on the probability value, having the increase in the number of exams less impact. Comparing both algorithms, it is possible to conclude that they are quite similar; however, the output value of `code2.c` is more influenced by the probability value than `code1.c`. Thus, for a real situation where the exams have a high probability of having a student in common, `code1.c` is better to solve the problem.

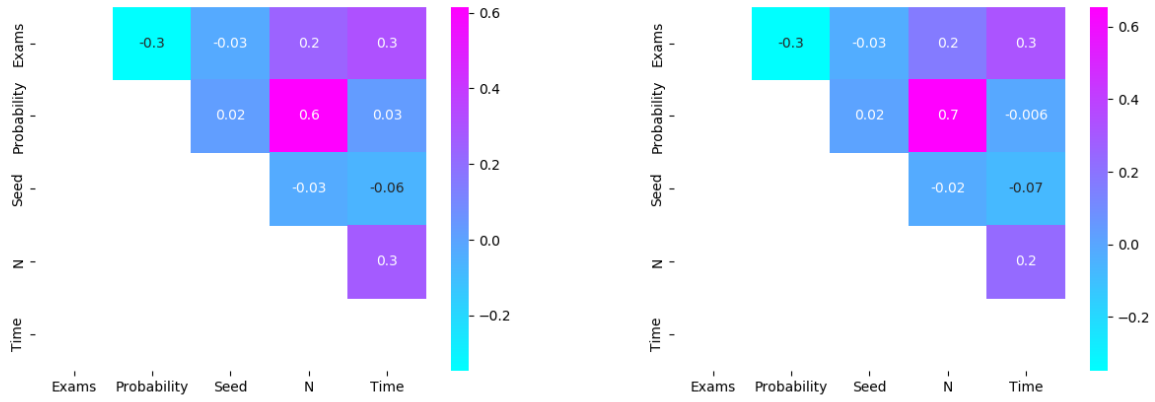


(a) Average timings.



(b) Standard deviation of timings.

Figure 6: Surface plots for the computational time



(a) `code1.c` (b) `code2.c`

Figure 7: Correlation

4. Linear Regression

Another way of characterizing the data obtained from the experiments is through linear regression, an approximate mathematical model created from available information which can be used to estimate the run time for

a given instance of the problem [2, 4].

From the analysis presented in Section 3, it is possible to notice that the run time increases in an exponential fashion according to both the probability and the number of exams with a forced plateau when time reaches the cut-off limit. Thus, we perform a log transformation in run time to obtain data that can be described by a linear model more efficiently.

Furthermore, similar to the previous evaluation, we fixed the value for one factor into different subgroups so that the influence of each variable can be measured for the different strata. It is also important to report that, to obtain a model that describes the algorithms' actual behavior, we do not take into account the results that reached the cut-off time limit since it is an artificial instrument, which could distort the model.

Firstly, the results of the linear regression by fixing the probability values is presented in Figure 8. From the figure it is possible no notice that the data can be indeed described by the linear model in this logarithmic space, however we can also see that, as the probability of overlapping exams increases, there is fewer data to be fitted into the model, since they tend to reach the time limit faster. Nonetheless, the model can still capture this tendency to some degree: e.g. for a probability of 45.25%, the run time predicted by the model can reach up 10^{13} seconds (more than 300,000 years).

In the figure we also report the values for the model's variance (R^2) measure, which represents the gain of introducing the linear model in comparison to the mean value [4, p. 383]. There are, however, examples of how the data distribution may influence this value. For example, for a probability of 22.63%, the R^2 is strongly negative for both approaches, possibly due to a higher concentration of values around the mean when there are 40 or 50 exams, but the graph shows that the linear model is still suitable for smaller quantities of exams.

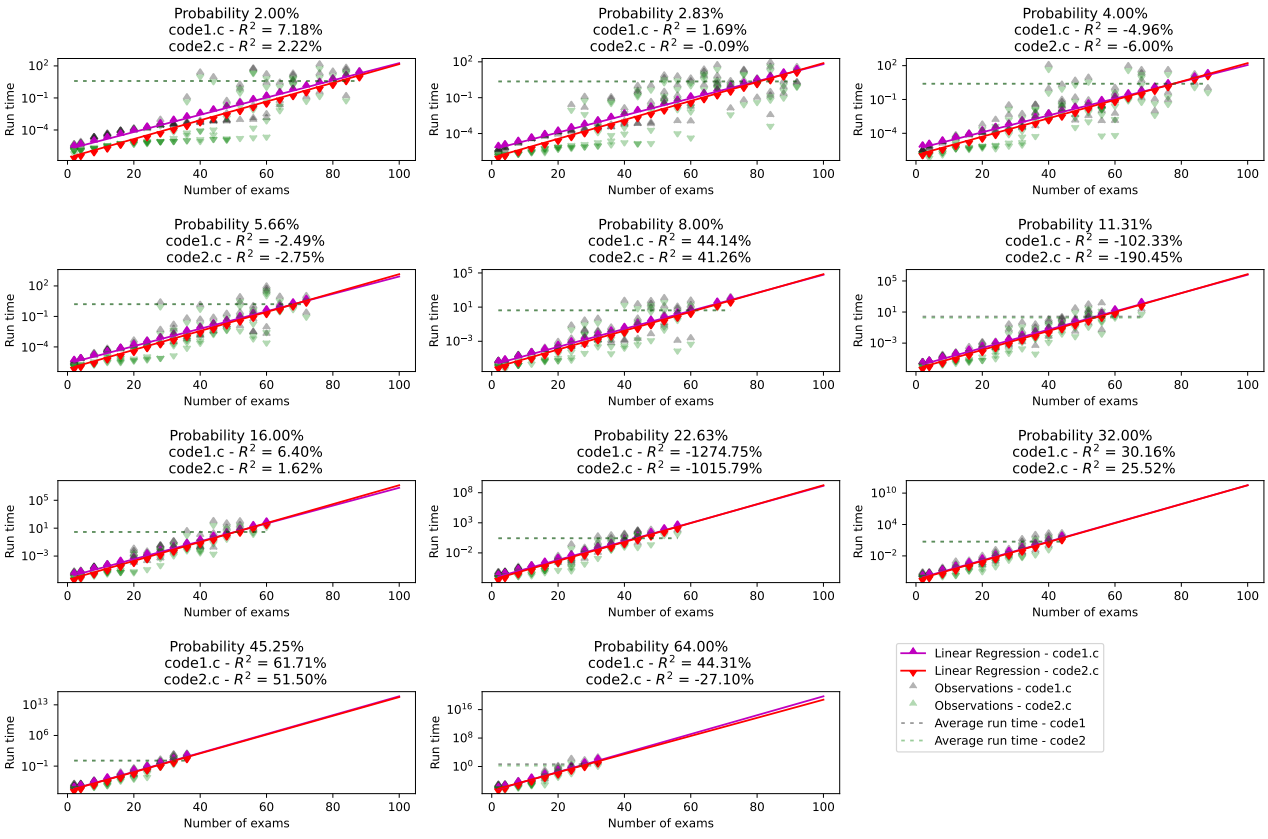


Figure 8: Linear regression model for fixed probability values

The same Linear Regression experiment was executed by fixing the number of exams and fitting the model to relate the overlapping probability with the run time. These results are shown in Figure 9. For completion, we decided to keep the experiments in which there is not sufficient data to obtain a full-fledged model, specially with more than 40 exams. As discussed earlier, when there is a high number of exams, the algorithms reach the time limit easily, resulting in cases such as with 96 and 100 exams, in which we are left with no data to fit a model, or other cases (76, 80, 84 and 88 exams) in which the resulting model does not reflect the expected behaviour (increasing run time) due to lack of data and random fluctuation.

Besides those extreme cases, the same observations from Figure 8 can be made in this case. The fitted model does seem to describe the tendency of the data well even in cases with a negative R^2 measure, specially when comparing how lower values are distributed with relation to the mean. It is also interesting to notice that

for a small number of exams (2, 4 and 8 exams), the `code1.c` approach tends to decrease its run time as the probability increases, however the magnitude of the results is still small (10^{-6}) and this difference may be due to randomness rather than performance.

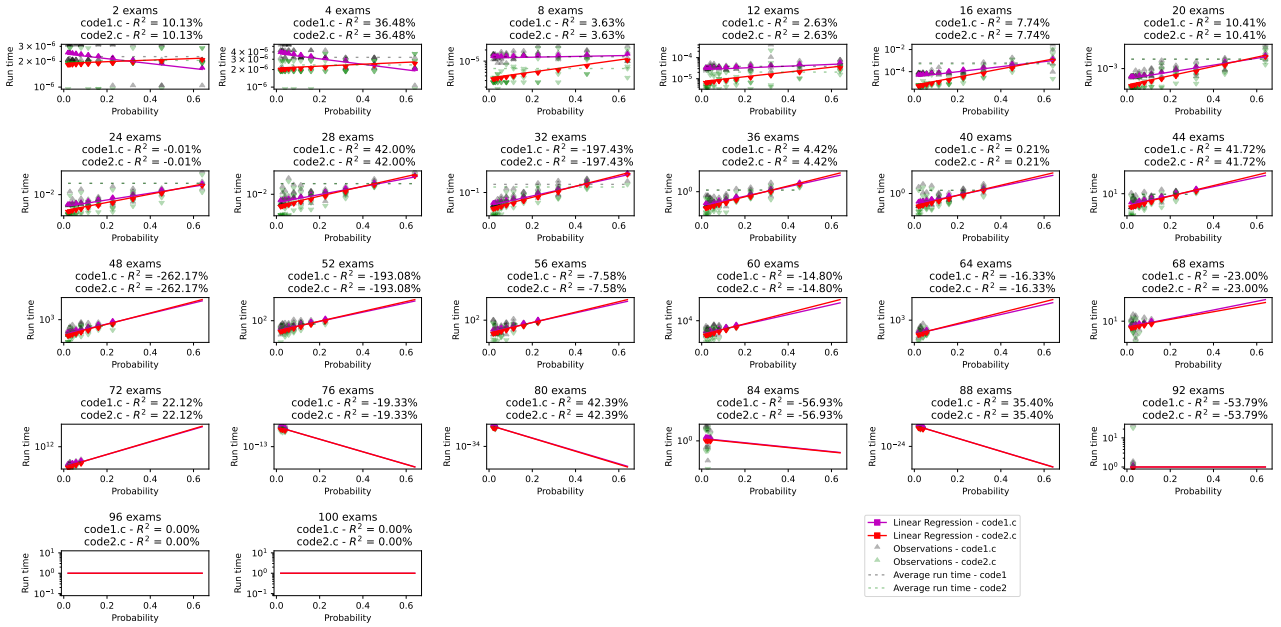


Figure 9: Linear regression model for fixed numbers of exams

5. Conclusion and Future Scope

After all the experiments carried out and a careful analysis of the data collected, it is possible to conclude that both algorithms have similar behavior. The main factors influencing its performance are the number of exams, and the probability, with a non-direct relationship between them. That is, for low numbers of exams, the probability value has little impact on the running time of the algorithms; however, for large exam values, a small increase in probability can exponentially increase the algorithms' execution time. A more detailed analysis, through the correlation graphs, reveals that the `code2.c` is more influenced by the probability value than `code1.c`.

The linear regression model also indicates that the algorithms have an exponential complexity regarding both variables. We also show that for large instances, the cut-off time limit is reached even for low probabilities, resulting in difficulties to approximate a model for them.

The following steps of this work involve the pre-registration of hypotheses, which is based on the outcomes of this work, to confirm the elements that may impact the performance of your algorithms and define hypotheses to test in the third comprehensive research. We also would like to evaluate how a combination of both variables may perform in a Linear Regression model.

References

[1] Bigelow, S. J. Definiton - workload. `https://searchdatacenter.techtarget.com/definition/workload`, 2020. Accessed: 2020-12-30.

[2] IBM. Linear regression. `https://www.ibm.com/topics/linear-regression`. Accessed: 2021/11/03.

[3] Mordkoff, J. T. The assumption (s) of normality. *Dostupno na: goo. gl/g7MCwK (Pristupljeno 27.05. 2017.)* (2016).

[4] Morettin, P. A., and Bussab, W. d. O. *Estatistica basica*. Saraiva, São Paulo, 2010.