

Preliminary draft 07:28 19 August 2022

19 August 2022
paulo.areyuna@sansano.usm.cl

Studying the Complexity of User Analysis tasks

Paulo Areyuna C.
CERN, CH-1211 Geneva, Switzerland

Keywords: PanDA, Data Analytics, Code Complexity

Summary

In this report we present our results for the measurement of complexity involving analysis tasks. We combine different metrics of analysis tasks such as the number of cores used, the execution time and the size of the datasets involved. With this we postulate a metric of task complexity. The resulting metric can be used to group different tasks due to an accidental stratification produced in the data.

Contents

1 Introduction **2**

2 Realistic measure of complexity **2**

3 Data-driven metric of complexity **2**
3.1 Variable definition 2
3.2 Studied samples 3
3.3 Looking for correlations 3
3.4 An empirical complexity 3
3.5 Complexity benchmarks for different input formats 5

4 Conclusions **6**

5 Acknowledgments **8**

6 Further information **8**

1 Introduction

Due to the large volume of data produced in the Large Hadron Collider, the analysis of these datasets is performed in a distributed system. Users submit their tasks that are executed in a distributed computing environment. The framework for management of the workload in the distributed system is called PanDA[1].
In order to optimize the task distribution over the different servers, it would be very useful to develop a metric to know in advance how long is a task going to take, so we tried to study the complexity of these tasks.

2 Realistic measure of complexity

In a first attempt, we used the package RADON [2] to calculate some complexity metrics using as an input the source code of the task. This package computes, among other things, standarad metrics of complexity, as the number of code lines, or the number of loops inside the code. However, the users upload compiled binaries of their tasks, so we can not have access to the source code using PanDA. We can't calculate the complexity of a task in advance without the analysis of the source code. So, we decided to calculate a posteriori task complexity for tasks that has already been completed

3 Data-driven metric of complexity

3.1 Variable definition

Due to the complications for knowing on advance the complexity of a task, we changed the objective of the project and tried another approach. We considered several studied different parameters from already done tasks. The relevant parameters for complexity studies are the following:

- corecount: The number of cores used to complete the task.
- input_dataset_size: The total size in bytes of all the input datasets.
- output_dataset_size:The total size in bytes of the output datasets generated by the task.
- libs_size: The total size in bytes of the libraries used to complete the task.
- duration_secs: Execution time of the task.

The way to obtain these metrics can be seen in Table 1. It's worth to mention that we also calculated some other metrics such as the number of finished files processes (input_dataset_nfilesfinished) and the format of the dataset (input_dataset_format). We didn't used these metrics to calculate complexity but they can be obtained with the scripts made during this work. These scripts can be found in the GitHub repository of the project (See Section 6).

[1]https://panda-wms.readthedocs.io/en/latest/index.html
[2]https://radon.readthedocs.io/en/latest/

Metric	ES index	procedure for extraction
corecount	atlas_tasks-*	Directly from the ES index
input_dataset_size	atlas_datasets_info-*	Sum over all input datasets
output_dataset_size	atlas_datasets_info-*	Sum over all output datasets
libs_size	atlas_datasets_info-*	Sum over all library datasets
duration_secs	atlas_tasks-* / atlas_jobs_archived-*	endtime-starttime.

Table 1: Description of the metrics extracted to compute the complexity of a task. The execution time uses 2 indexes because of missing timestamps in the task index.

3.2 Studied samples

We obtained the previously defined variables for all **analysis** tasks that got the status of **done** During the **first week of July**. Additionally, we added two extra filter: the first one was on the tasks submitted by the top 10 users with highest input dataset sizes for completing their tasks The second one was the removal of all tasks with input or output dataset size of zero. After that, we performed an outlier removal using the interquartile range criterium.
To gain some intuition, we obtained some histograms for the distribution of these variables (see Figure 1). We can see that in general, most of the tasks have small values for these parameters.

3.3 Looking for correlations

At first we wanted to find a way to relate the other parameters with the execution time of the task, so we searched for correlations between these parameters. The results are available in Figures 2 and 3. As can be seen, there are no clear correlations between the parameters and the execution time. Therefore, we can not define a complexity metric that can predict the execution time.

3.4 An empirical complexity

Since we were not able to find a metric that can predict the execution time, we defined an empirical metric that takes into account also the execution time. To do so, we did a standardized scaling of our parameters considering the following definition for a variable x_i:

$$x_i^{std} = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{1}$$

Doing this, every parameter gets a value between 0 and 1, and the total complexity can be defined as the sum of these quantities:

$$\begin{aligned} \text{complexity} =& \text{corecount}^{std} + \text{input_dataset_size}^{std} + \text{output_dataset_size}^{std} + \\ &+ \text{libs_size}^{std} + \text{duration_secs}^{std} \end{aligned} \tag{2}$$

With this definition, the complexity distribution and it's relation with the execution time can be seen in Figure 4. From the definition of complexity in equation (2) it can be noted that

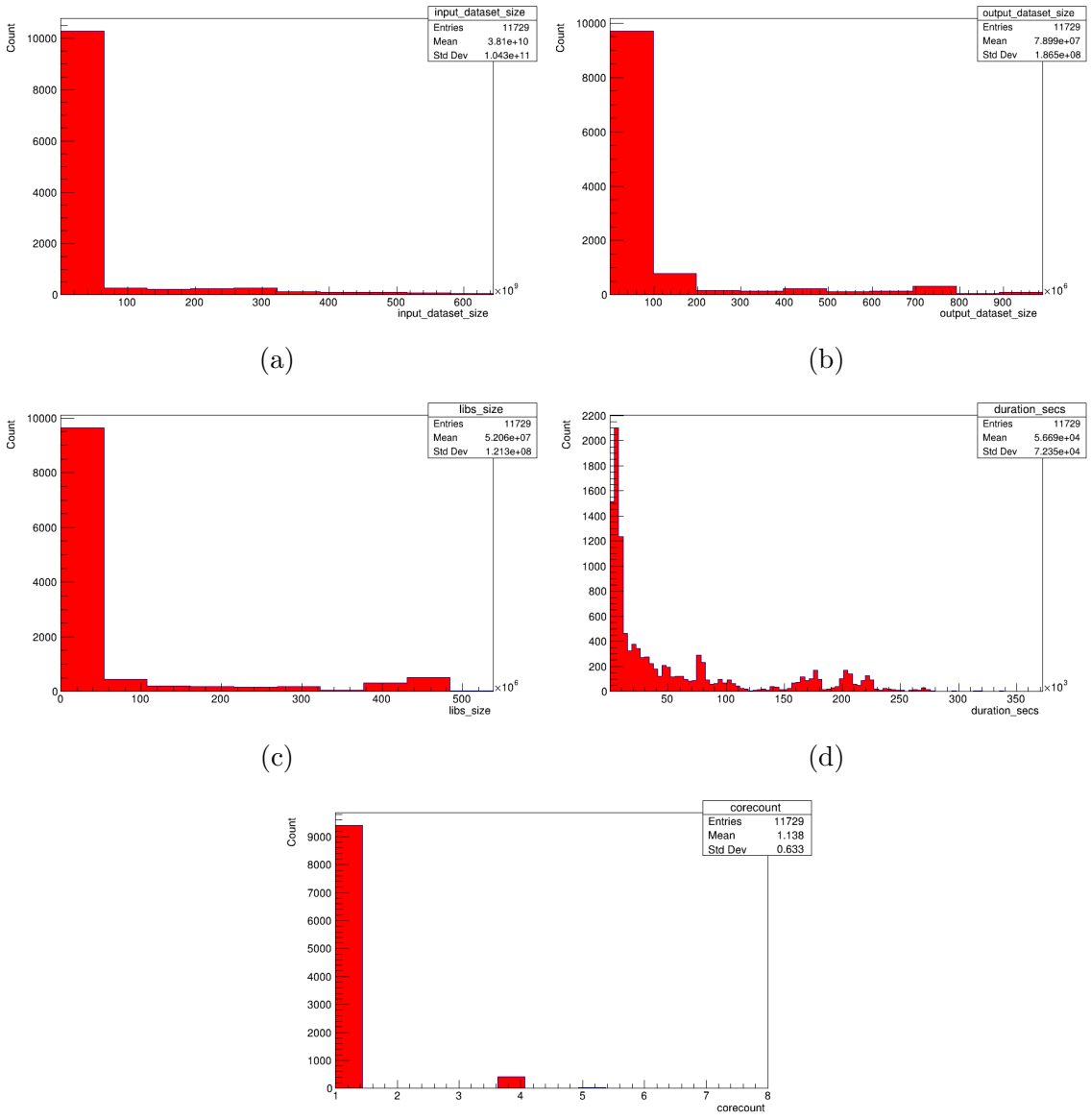


Figure 1: Distribution for the relevant variables

the relation between complexity and execution time should be linear. The scope of the linear function depends on the standardization of the variable, while the intercept depends on the combination of dataset sizes and the number of cores. It's worth to mention that the points in figure 4b can be fitted using different linear functions for different collections of points. The relevant feature of this figure is that the straight lines are completely distinguishable, so we can conclude that our complexity metric generates an accidental stratification on the dataset, as can be seen in Figure 4b.

For the sake of completeness, we included the distribution of the rest of the parameters in the scatter plot of the complexity against the execution time. We can see in Figure 5 that there is an interplay between the sum of the dataset sizes, since the complexity depends on

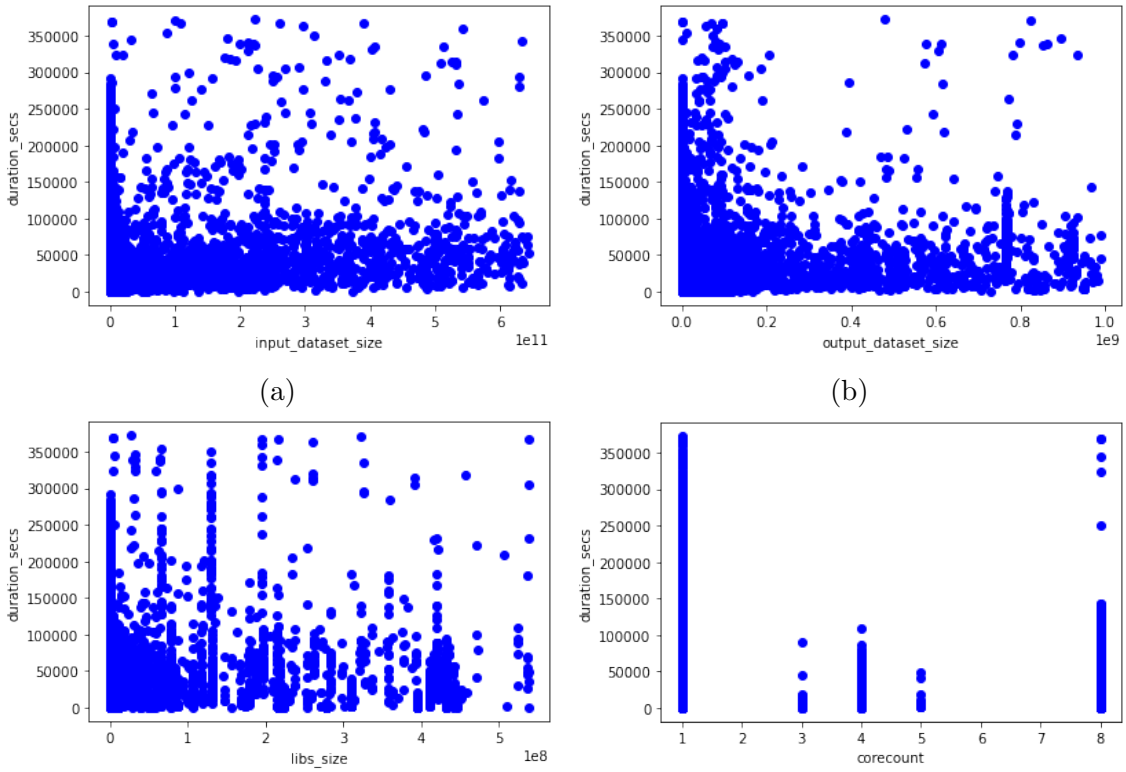


Figure 2: Correlations between the variables these plots were obtained to see the predictive power of the parameters on the execution time.

the sum of input, library and output size, the scatter plots for these variables don't present a clear separation as the one of the core count.

3.5 Complexity benchmarks for different input formats

We studied the behaviour of our new metric for the following input dataset formats: DAOD_PHYS, DAOD_JETM6, DAOD_TOPQ1 and DAOD_EXOT5. The complexity distributions for these dataset formats can be seen in Figure 6.

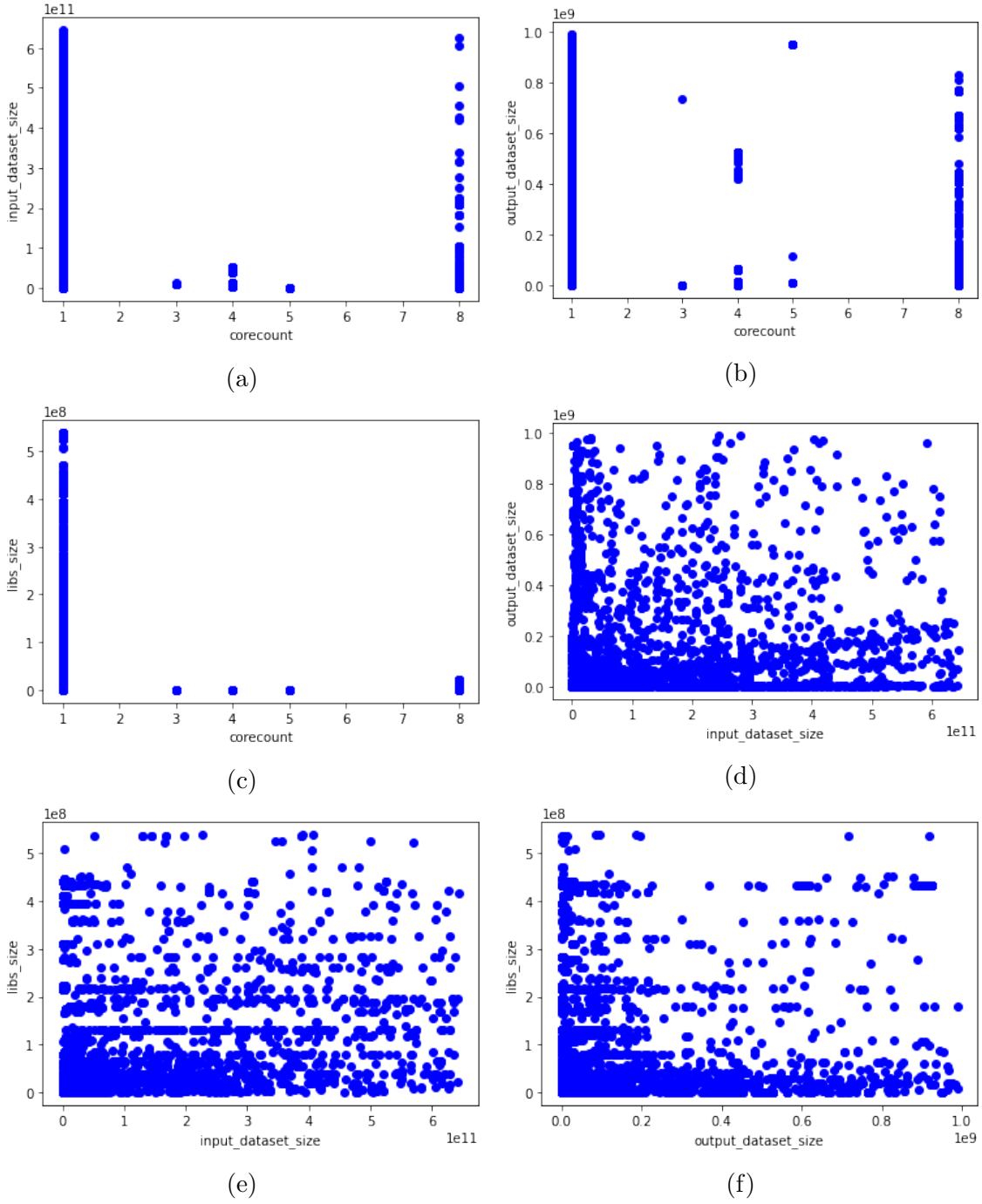
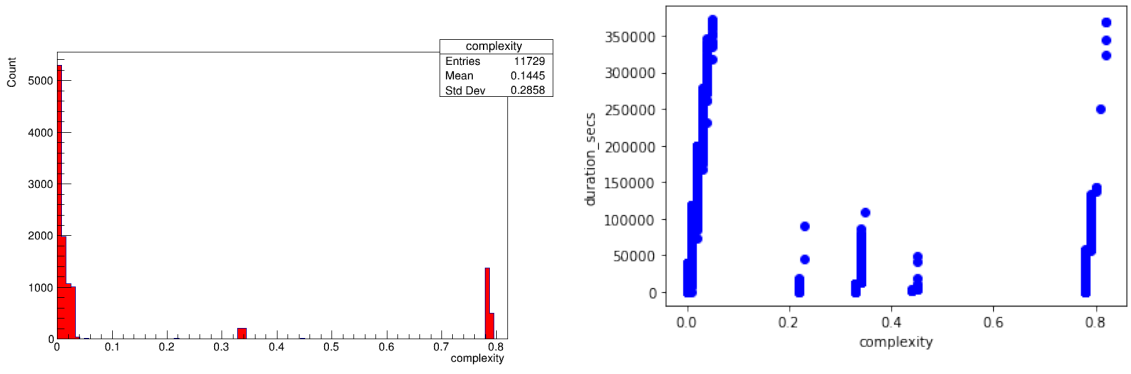


Figure 3: Scatter plots to look for the possibility of dimensional reduction

4 Conclusions

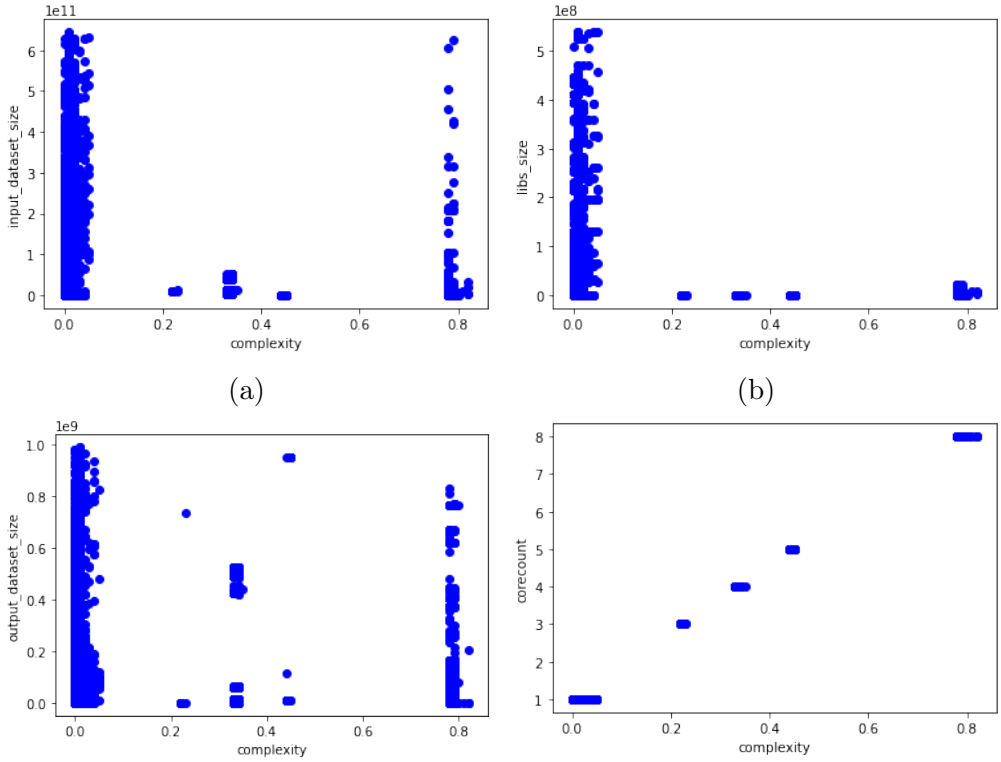
In this project we were able to define a metric that measures the complexity of a task. The definition of this metric needs info from different ES indexes and we developed a code that extracts all this info into a single parameter of complexity. This metric has no predictive power but it will be useful for monitoring the complexity of already done tasks. One thing



(a) Complexity distribution

(b) Execution time as a function of the complexity

Figure 4: Complexity metric and it's relation with the execution time.



(a)

(b)

(c)

(d)

Figure 5: Relations between complexity and the relevant parameters.

that could be done is to modify our definition of complexity, maybe define several complexities and compute some benchmarks to find an optimal definition of complexity. Another interesting possibility is to group the tasks considering the stratification that is produced naturally from the definition of the metric.

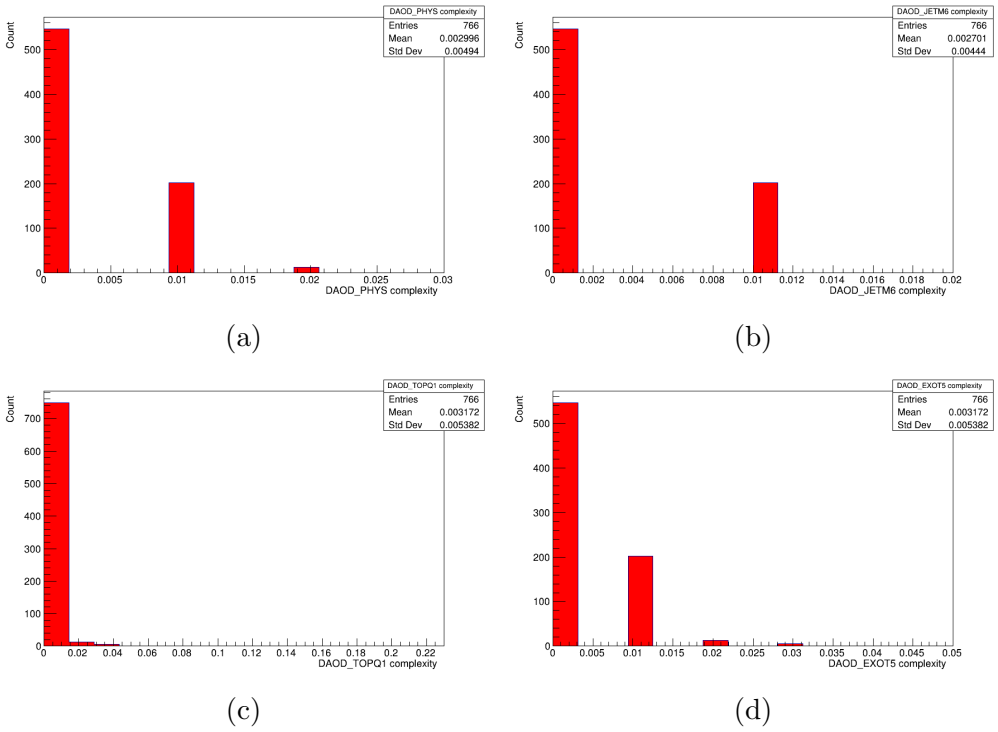


Figure 6: Relations between complexity and the different values of the relevant parameters.

5 Acknowledgments

First, I would like to thank my home institution, Instituto Milenio SAPHIR, for partial funding to attend the Summer Student Programme. In second place, I would like to thank my supervisor here at CERN, Alexei Klimentov. Also, additional thanks to Aleksandr Alekseev and Maria Grigorieva for their incredible patience with me. As I come from a Physics background, entering the world of Data Analytics was hard at the beginning and I wouldn't have make it without their help.

6 Further information

The GitHub repository of the project can be found in the following link:
https://github.com/pauloareyuna/complexity-metrics