Haydée Guillot Jiménez

Applying Process Mining to the Academic Administration Domain

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Antonio Luz Furtado

Rio de Janeiro
September 2017

Haydée Guillot Jiménez

Applying Process Mining to the Academic Administration Domain

Dissertation presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the undersigned Examination Committee.

Rio de Janeiro, September 21th, 2017

Haydée Guillot Jiménez

Graduated in Computer Science from the University of Havana (UH), Havana - Cuba in 2012. Joined the Master Program in Informatics at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) in 2015.

Bibliographic data

Jiménez, Haydée Guillot

Applying Process Mining to the Academic Administration Domain / Haydée Guillot Jiménez; advisor: Antonio Luz Furtado. – Rio de Janeiro: PUC-Rio, Departamento de Informática, 2017.

v., 67 f. : il. ; 29,7 cm

1. Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informatica.

Inclui bibliografia.

1. Informática – Teses. 2. Busca por palavras-chave. 3. Análise acadêmica. 4. frequência de conjuntos. 5. mineração de processos. 6. organização curricular. I. Furtado, Antonio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

First of all, I would like to thank my parents for all the love and strength that I need at this point of my life. My husband Daniel for embarking me on this trip that ended up being a wonderful experience. To my brother for always being a guide and example to follow, to my family and true friends in general for helping me to achieve this goal.

Thank you so much to Professor Antonio Luz Furtado and Professor Marco Antonio Casanova, for all the guidance provided throughout this time of learning.

Thanks to my guardian angel Carol for taking this journey with me and give me the necessary strength to success.

To PUC-Rio and CAPES for funding my research.

To all my classmates, professors and staff from the Department of Informatics.

Thank you so much to all of you!

Abstract

Jimenéz, Haydée Guillot; Furtado, Antonio Luz (Advisor). **Applying Process Mining to the Academic Administration Domain**. Rio de Janeiro, 2017. 67p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Higher Education Institutions keep a sizable amount of data, including student records and the structure of degree curricula. This work, adopting a process mining approach, focuses on the problem of identifying how closely students follow the recommended order of the courses in a degree curriculum, and to what extent their performance is affected by the order they actually adopt. It addresses this problem by applying to student records two already existing techniques: process discovery and conformance checking, and frequent itemsets. Finally, the dissertation covers experiments performed by applying these techniques to a case study involving over 60,000 student records from PUC-Rio. The experiments show that the frequent itemsets technique performs better than the process discovery and conformance checking techniques. They equally confirm the relevance of analyses based on the process mining approach to help academic coordinators in their quest for better degree curricula.

Keywords

Academic analytics; frequent itemsets; process mining; degree curriculum structure.

Resumo

Jimenéz, Haydée Guillot; Furtado, Antonio Luz (Orientador). **Aplicação de Mineração de Processos ao Domínio Acadêmico Administrativo**. Rio de Janeiro, 2017. 67p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

As instituições de ensino superior mantêm uma quantidade considerável de dados que incluem tanto os registros dos alunos como a estrutura dos currículos dos cursos de graduação. Este trabalho, adotando uma abordagem de mineração de processos, centra-se no problema de identificar quão próximo os alunos seguem a ordem recomendada das disciplinas em um currículo de graduação, e até que ponto o desempenho de cada aluno é afetado pela ordem que eles realmente adotam. O problema é abordado aplicando-se duas técnicas já existentes aos registros dos alunos: descoberta de processos e verificação de conformidade; e frequência de conjuntos de itens. Finalmente, a dissertação cobre experimentos realizados aplicando-se essas técnicas a um estudo de caso com mais de 60.000 registros de alunos da PUC-Rio. Os experimentos indicam que a técnica de frequência de conjuntos de itens produz melhores resultados do que as técnicas de descoberta de processos e verificação de conformidade. E confirmam igualmente a relevância de análises baseadas na abordagem de mineração de processos para ajudar coordenadores acadêmicos na busca de melhores currículos universitários.

Palavras-chave

Análise acadêmica; frequência de conjuntos; mineração de processos; organização curricular.

Table of Contents

List of figures

List of tables

1
Introduction

1.1 Motivation

Information systems are becoming more and more intertwined with the operational processes they support, which typically involve many types of events. Nevertheless, organizations have problems extracting valuable information from these data. The goal of process mining is to use available event data to extract process related information and automatically build process models by observing the events recorded by some information system (VAN DER AALST, 2011).

Process mining is a relatively young research discipline that lies between machine learning and data mining, on one side, and process modeling and analysis, on the other. Process mining can be used to discover, monitor and improve real processes by extracting knowledge from available event logs in today's information technology systems (MUNOZ-GAMA, 2014).

Most research in process mining has focused on process discovery techniques, neglecting the importance of conformance analysis. However, conformance techniques have become an essential part in the process management life cycle. These days, finding relations between variables in databases is one of the most important sources of knowledge, because this information can be used to support decision-making, which is particularly critical to activities such as marketing.

Finding frequent itemsets is a process mining technique for characterizing data by detecting regularities in large-scale data sets. Indeed, as we shall point out when discussing the current literature, several techniques have been developed to discover frequent itemsets.

The volume of information generated through Higher Education Institutions (HEIs) information systems is constantly growing. Recently, HEIs are increasingly investing in the treatment of all such information. Duly analyzed, the stored data can be taken as an added value, which can improve HEIs decision-making processes and strategies (OBLINDER; CAMPBELL, 2007). As a matter of fact, there is a large number of studies in the area of education, organized under the topics of *Academic Analytics* or *Educational Data Mining, Learning Analytics,* and *Predictive Analytics* (BAEPLER & MURDOCH, 2010; BOGARD et al., 2011; CALVERT, 2014; GOLDSTEIN & KATZ, 2005; LAURÍA et al., 2012; NORRIS & LEONARD, 2008; PARACK et al., 2012; PHILLIPS et al., 2012; TAIR & EL-HALEES, 2012; YUKSELTURK et al., 2014)

For example, uncovering hidden knowledge from student data may help achieve high rates of student *retention* (i.e. low rates of student dropout), as well as better understand how to attract new students. As a second example, tracking the set of courses followed by students can be quite useful to check degree curricula, and may eventually justify inviting the academic coordinators to joint discussions on the subject, with the participation of the research group in charge.

Such research can profitably apply process mining techniques, which, based on event logs, reproduce the footprint of the whole process. For the purpose of the academic research methodology that we adopt in this dissertation, a degree program is viewed as a *process*, where students are the *cases* and course status (enrollment or result) are the *activities*.

1.2 Goal and Structure of the Dissertation

The overall goal of this dissertation is to assess existing statistical and process mining techniques, in order to find which ones are more appropriate to answer certain meaningful questions about student behavior, with respect to the courses in which they enroll as part of their chosen curricula degree.

The text is structured as follows. Chapter 2 summarizes related work. Chapter 3 exposes the techniques used in the dissertation. Chapter 4 introduces the questions and the results obtained from the application of the techniques to real and anonymized student data. Finally, Chapter 5 presents the conclusions and our objectives for future work.

2
Related Work

In this chapter, we briefly discuss Academic Analytics, and summarize techniques and algorithms for Process Mining.

2.1 Academic Analytics

Academic Analytics, also known as Educational Data Mining, is a business intelligence process in the area of education, focused on supporting strategic decisions based on data. Academic Analytics can help academic coordinators improve students' success and reduce the dropout rates. Increased competition, accreditation, evaluation, and regulation are the main factors that encourage the adoption of analytics in HEIs. Goldstein and Katz (2005) argue that, although university centers have access to the required data, they still do not adequately analyze it. To avoid this gap, Norris and Leonard (2008) sought to explain what could be achieved by adequately analyzing the available data, taking into account the peculiarities of each university, course, teacher and student.

Decision-making, at its most basic level, draws conclusions and decisions based on experience, without extensive data analysis. However, decisions taken in HEIs are too vital to be based on assumptions or intuitions, and as a rule, must be supported by data and facts. Once one has the data and the appropriate analysts to answer the questions, it is important to realize that academic analysis varies according to the area to which it is applied. Goldstein and Katz (2005) show how the same data may respond to seven different functional areas. Moreover, accurate analysis requires the work of both system administrators and analysts, as otherwise the data will remain a total waste (BAEPLER & MURDOCH, 2010).

In the literature, there are several definitions for the term Learning Analytics, which in a broad sense is the process of collecting, structuring and analyzing the data obtained in academic environments. Some of these definitions are presented in what follows.

According to Siemens (2010) “Learning Analytics is the use of intelligent data, student-produced data, and analysis models to uncover information and social connections, originating in a digital environment to predict and advise the learning of people”. Papamitsiou and Economides (2014) stated that Learning Analytics aims at “modelling student behavior, predict performance, increase reflection and awareness, predict dropout and persistence, improve evaluation and feedback, makes it possible to recommend resources”.

The area of Predictive Analytics focuses on the students already enrolled in the courses, aiming at predicting the behavior of a new student and taking preventive actions, if necessary. The best-known techniques to extract this knowledge in the field of data mining are Classification, Clustering, Regression, Neural Networks, Association Rules, Decision Trees, Genetic Algorithm, among others.

The prediction of the behavior of the students has been studied for some time. Baradwaj and Pal (2012) presented the results of 9 studies carried out between the years 2005 and 2011 in various educational centers in Asia. He uses the Decision Trees technique, since it gives better results than the other techniques. Ahmed and Elaraby (2014) made a similar study, but focused more on generating classification rules. Ramaswami and Rathinasabapathy (2012) presented as main objective the construction of a Bayesian Network model to analyze the relationship between socioeconomic factors and academic factors in the student’s performance. In the research carried out by Saa (2016), personal and social data were also taken into account, but the techniques applied were Decision Trees and the Naïve Bayes algorithm. Deshpande et al. (2016) suggested that higher performance can be obtained by combining the Decision Trees technique with the k-Means clustering technique.

2.2 Process Mining

Process mining, in the business sense, was first introduced in 1998 by (AGRAWAL et al., 1998) under the name of *workflow mining*. Since then, many research groups have focused on the mining of process models; particularly relevant are the works of (VAN DER AALST et al, 2002; VAN DER AALST et al., 2003; VAN DER AALST et al., 2004; VAN DER AALST, 2011).

Figure 1 shows Van der Aalst's process mining view. Briefly, we have the real world, where things are happening, and we have a software system, which records, under the form of events, all things of interest that take place in the world. The arrows indicate that we can automatically learn a process model from such event data, and that conformance checking can be done by comparing event logs with the process model. Finally, we can enrich the process model with information about performance and possible deviations.

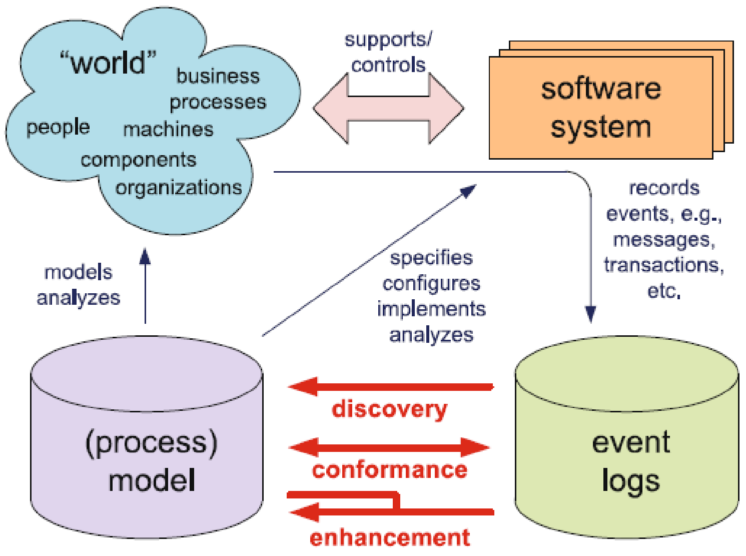


Figure 1. Process Mining workflow (VAN DER AALST, 2011, pp 9).

There are five quality criteria that allow us to evaluate how well a process model describes the observed data: *soundness*, *replay fitness*, *precision*, *generalization* and *simplicity* (VAN DER AALST, 2011). On the basis of these properties, we can evaluate process discovery algorithms, such as: ALPHA MINER, HEURISTIC MINER, INDUCTIVE MINER and FUZZY MINER. Cook et al. introduced the concept of process validation, setting up the beginning of conformance checking (COOK & WOLF, 1999; COOK et al., 2001). More comprehensive references are (VAN DER AALST, 2011; VAN DER AALST, 2012; ROZINAT & VAN DER AALST, 2008).

Another technique from the field of data mining is referred to as *frequent itemset* discovery, a theme addressed in (RAJARAMAN & ULLMAN, 2011; AGRAWAL et al., 1993; CHEUNG & FU, 2004; BAYARDO, 1998; VO et al., 2016).

2.2.1. Workflow Mining

Agrawal et al. (1998) applied process discovery in the context of IBM workflow products. They modelled processes as graphs, with individual tasks as nodes, and represented a possible flow from one activity to another as an edge. Their approach requires that the model have a single start task and a single end task.

Van der Aalst and coworkers developed techniques for (re)discovering workflow models (VAN DER AALST et al, 2002; VAN DER AALST et al., 2003; VAN DER AALST et al., 2004). Creating a workflow is a complicated task; usually there are discrepancies between the actual workflow processes and how these processes as perceived by management. They presented an algorithm to extract a process model from a log and represent it in terms of a Petri net, a technique known as a *Play-in process*. Unfortunately, it is not possible to (re)discover all workflow processes using this approach.

The ALPHA MINER algorithm was actually the first algorithm that bridged the gap between event logs or observed data in general, and the discovery of a process model (VAN DER AALST et al., 2002; VAN DER AALST et al., 2004). The ALPHA MINER algorithm uses as input an event log, but only cares about the ordering, ignoring the resources, other data elements, and the timestamps at which the events take place. Also, they do not use the case ID or any properties of the case. They only look at the order of activities within a particular case. Moreover, this algorithm does not guarantee soundness, does not filter out noise, has issues dealing with short-loops, and cannot discover long-term dependencies.

To overcome these difficulties, many extensions of the ALPHA MINER were proposed, such as: ALPHA+, which can deal with short-loops and self-loops (MEDEIROS et al., 2004); ALPHA++, which is able to find additional non-free choice constructs (WEN et al., 2007); and ALPHA#, which can also discover several invisible tasks, that is, tasks that exist in a process model, but not in the event log (WEN et al., 2010).

In addition to these extensions of ALPHA MINER, other algorithms have been implemented, such as the HEURISTIC MINER (WEIJTERS et al., 2006; WEIJTERS & RIBEIRO, 2010), and INDUCTIVE MINER (LEEMANS et al., 2013; LEEMANS et al., 2014).

2.2.2. Conformance Checking

Conformance checking is a process mining technique that compares an existing process model with an event log of the same process, and therefore belongs to the family of replay process techniques. It can be used to check whether reality, as recorded in the log, conforms to the model, and vice versa (WIKIPEDIA, 2017b). The goal is to find commonalities and discrepancies between the modeled behavior and the observed behavior.

Conformance checking can be performed for various reasons. First, it may be used to audit processes, that is, to check whether reality conforms to some normative or descriptive model (VAN DER AALST, 2011). Deviations may point to frauds, inefficiencies, and poorly designed or outdated procedures. Second, conformance checking can be used to evaluate process discovery results (VAN DER AALST & VERBEEK, 2014).

There are three types of conformance checking techniques: token-based replay, footprints comparison, and alignment-based. We refer the reader to (VAN DER AALST, 2011, pp 191-194) for a discussion about these approaches.

2.2.3. Frequent itemsets

In 1993, based on concepts about association rules, (AGRAWAL; IMIELIŃSKI; SWAMI) published a paper describing the relationships between stored data, focused mainly on analyzing the sales data of a supermarket. Subsequently, the problem was generalized to analyze different scenarios. The basic objectives are: to determine how often two or more objects co-occur and, once the frequency of such sets is found, to extract association rules between the objects.

There are several algorithms that search for frequent itemsets in databases (RAJARAMAN & ULLMAN, 2011; CHEUNG & FU, 2004; VO et al., 2016; ZAKI & JR, 2014). The most known and used algorithm, called APRIORI (RAJARAMAN & ULLMAN, 2011), is based on counting the frequencies of occurrences of subsets in the database transactions. The algorithm first generates all subsets of size one, and counts the number of occurrences of the elements; then, it forms the subsets of size two, calculates their frequency, and so on. To avoid overwork, a *confidence threshold* is defined, and only those subsets that have a frequency greater than the confidence threshold are returned. So, if during the execution of the algorithm, the frequency of a subset is less than the threshold, any set that contains this subset is discarded.

Variations of the APRIORI algorithm include those that do not take into account the confidence threshold (CHEUNG & FU, 2004), and those that attempt to identify long patterns (BAYARDO, 1998) and use the N-list and subsume concepts (VO et al., 2016).

3
Process Mining Techniques

This chapter introduces the process mining techniques used in the dissertation. Section 3.1 presents discovery and conformance techniques, while Section 3.2 covers frequent itemsets techniques.

3.1 Discovery and Conformance Techniques

3.1.1. Alpha Miner

The ALPHA MINER algorithm (VAN DER AALST, 2011), given as input an event log *L*, consisting of a multiset of traces, where a trace is a sequence of activities, proceeds to discover a model *α(L)*, structured as a *Petri net*. Before describing the algorithm, we need to define a set of relationships that may exist between the activities.

Definition (Log-based ordering relations). Let *L* be an event log over a set of activities *A*. Let $a, b \in A$:

- *Direct succession*: $a >_L b$ if and only if there is a trace $\sigma = <t_1, t_2, t_3, ..., t_n>$ and $i \in \{1, . . ., n - 1\}$ such that $\sigma \in L$ and $t_i = a$ and $t_{i+1} = b$
- *Causality*: $a \rightarrow_L b$ if and only if $a >_L b$ and $b \not>_L a$
- *Choice*: $a \#_L b$ if and only if $a \not>_L b$ and $b \not>_L a$
- *Parallel*: $a \|_L b$ if and only if $a >_L b$ and $b >_L a$

For example, consider the following event log:

$$L_1 = [<a, b, c, d>^3, <a, c, b, d>^2, <a, e, d>]$$

For this event log, the following log-based ordering relations can be found

$>_{L1} = \{(a, b), (a, c), (a, e), (b, c), (c, b), (b, d), (c, d), (e, d)\}$

$\rightarrow_{L1} = \{(a, b), (a, c), (a, e), (b, d), (c, d), (e, d)\}$

$\#_{L1} = \{(a, a), (a, d), (b, b), (b, e), (c, c), (c, e), (d, a), (d, d), (e, b), (e, c), (e, e)\}$

$\|_{L1}$ = {(b, c), (c, b)}

The ALPHA MINER algorithm steps follow below, letting *L* be an event log and *α(L)* the resulting Petri net:

1. Determine the transition set T_{L}, in correspondence with the entire set of activities in *L*.
2. Fix the set of start activities (the first activity of each trace).
3. Fix the set of end activities (elements that appear last in a trace).
4. Create the set X_L with the computed pairs *(A, B)* of sets of activities such that:
   a. Every element $a \in A$ and every element $b \in B$ are causally related, that is, ($a \rightarrow_L b$).
   b. All elements in *A* are independent, that is, ($a_1 \#_L a_2$).
   c. All elements in *B* are independent, that is, ($b_1 \#_L b_2$).
5. Delete from X_L all pairs *(A, B)* that are not maximal among themselves, and place these maximal sets into a new set Y_L.
6. Determine the place set P_L, where each element *(A, B)* of Y_L is considered a place. To ensure the workflow structure, add a source place i_L and a sink place o_L.
7. Determine the flow relation set F_L:
   a. Connect each place $p_{(A,B)}$ with each element *a* of its set *A* of source transitions and with each element of its set *B* of target transitions.
   b. In addition, draw an arc from the source place i_L to each start transition $t \in T_I$ and an arc from each end transition $t \in T_O$ to the sink place o_L.
8. Return the Petri net $\alpha(L) = (P_L, T_L, F_L)$.

For example, consider the following event log:

L_2 = [<a, b, c, d, e, f, b, d, c, e, g>, <a, b, c, d, e, f, b, c, d, e, f, b, d, c, e, g>, <a, b, d, c, e, g>^{2}]

First we find the ordering relations

$>_{L2}$ = {(a, b), (b, c), (c, d), (d, e), (e, f), (f, b), (b, d), (d, c), (c, e), (e, g)}

$\rightarrow_{L2}$ = {(a, b), (b, c), (b, d), (c, e), (d, e), (e, f), (e, g), (f, b)}

$\#_{L2}$ = {(a, a), (a, c), (a, d), (a, e), (a, f), (a, g), (b, b), (b, e), (b, g), (c, a),

(c, c), (c, f), (c, g), (d, a), (d, d), (d, f), (d, g), (e, a), (e, b), (e, e),
(f, a), (f, c), (f, d), (f, f), (f, g), (g, a), (g, b), (g, c), (g, d), (g, f), (g, g)}

$||_{L2}$ = {(d, c), (c, d)}

Then, we apply the algorithm, and obtain:

1. T_L = {a, b, c, d, e, f, g}.
2. T_I = {a}.
3. T_O = {g}.
4. X_L = {({a}, {b}), ({b}, {c}), ({b}, {d}), ({c}, {e}), ({d}, {e}), ({e}, {f}), ({e}, {g}), ({f}, {b}), ({a, f}, {b})}.
5. Y_L = {({b}, {c}), ({b}, {d}), ({c}, {e}), ({d}, {e}), ({e}, {f}), ({e}, {g}), ({a, f}, {b})}.
6. P_L = {$p_{(\{b\}, \{c\})}$, $p_{(\{b\}, \{d\})}$, $p_{(\{c\}, \{e\})}$, $p_{(\{d\}, \{e\})}$, $p_{(\{e\}, \{f\})}$, $p_{(\{e\}, \{g\})}$, $p_{(\{a, f\}, \{b\})}$, i_L, o_L}.
7. F_L = {(b, $p_{(\{b\}, \{c\})}$), ($p_{(\{b\}, \{c\})}$, c), (b, $p_{(\{b\}, \{d\})}$), ($p_{(\{b\}, \{d\})}$, d), (c, $p_{(\{c\}, \{e\})}$), ($p_{(\{c\}, \{e\})}$, e), (d, $p_{(\{d\}, \{e\})}$), ($p_{(\{d\}, \{e\})}$, e), (e, $p_{(\{e\}, \{f\})}$), ($p_{(\{e\}, \{f\})}$, f), (e, $p_{(\{e\}, \{g\})}$), ($p_{(\{e\}, \{g\})}$, g), (a, $p_{(\{a, f\}, \{b\})}$), (f, $p_{(\{a, f\}, \{b\})}$) , ($p_{(\{a, f\}, \{b\})}$, b),
(i_L, a), (g, o_L)}.
8. α(L_2) = (P_L, T_L, F_L).

Figure 2 shows the model derived from event log L_2.

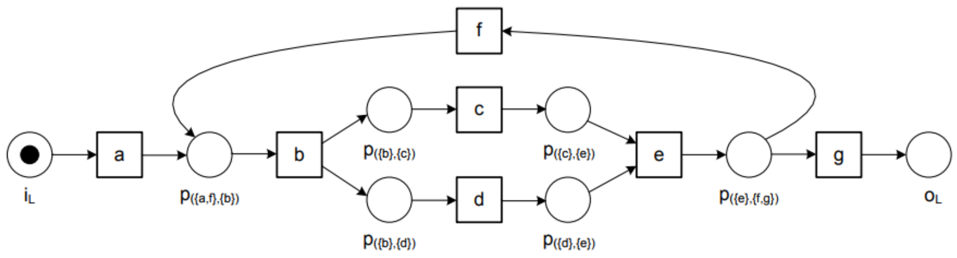


Figure 2. Petri net model obtained with ALPHA MINER.

The ALPHA algorithm has extensions, namely ALPHA+, ALPHA++, and ALPHA#, which basically follow the same 8 steps of the ALPHA algorithm, albeit with some variations, depending on the limitation they try to solve.

3.1.2. Heuristic Miner

The HEURISTIC MINER improves the ALPHA MINER in three ways: it takes frequencies into account, detects short-loops and detects skipping activities (WEIJTERS & MEDEIROS, 2006; WEIJTERS & RIBEIRO, 2010). It is based on the frequency in which one activity appears after another, and leaves only those flow paths where the frequencies are significant.

The first step is to build a directly-follows frequency matrix, constructed by counting how many times one activity appears directly after another:

$$|a >_L b| = \sum_{\sigma \in L} L(\sigma) \times \left|\left\{1 \leq i < |\sigma| \mid \sigma(i) = a \wedge \sigma(i+1) = b\right\}\right|$$

Once one has the frequency matrix, one can build the dependency matrix, where the dependency ratio between two activities is defined as:

$$|a \Rightarrow_L b| = \begin{cases} \frac{|a>_L b| - |b>_L a|}{|a>_L b| + |b>_L a| + 1} & \text{if } a \neq b \\ \frac{|a>_L a|}{|a>_L a| + 1} & \text{if } a = b \end{cases}$$

With the two matrices already built, we proceed, through a series of steps (described in detail in (VAN DER AALST, 2016)), to construct a Petri Net, where only those dependencies greater than a given threshold are represented.

As an example, we apply this algorithm to the event log:

L_3 = [<a, e>^{5}, <a, b, c, e>^{10}, <a, c, b, e>^{10}, <a, b, e>, <a, c, e>, <a, d, e>^{10}, <a, d, d, e>^{2}, <a, d, d, d, e>]

and obtain the frequency matrix:

$\lvert >_{L3} \rvert$	a	b	c	d	e
a		11	11	13	5
b			10		11
c		10			11
d				4	13
e					

and the dependency matrix:

|=>|	a	b	c	d	e
a		0.92	0.92	0.93	0.83
b	-0.92		0		0.92
c	-0.92	0			0.92
d	-0.93			0.80	0.93
e	-0.83	-0.92	-0.92	-0.93	

Figure 3 shows the Petri Net obtained from these two matrices.

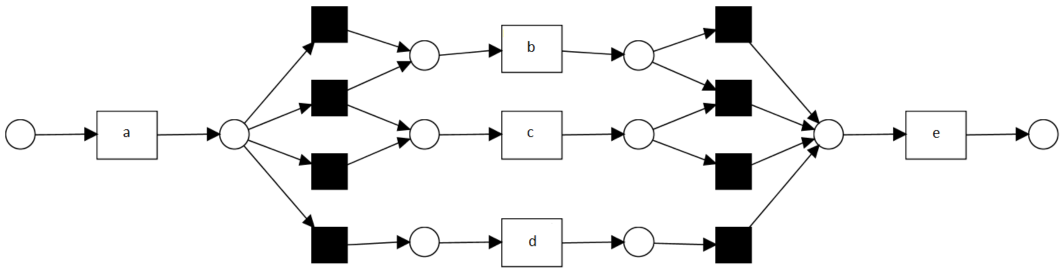


Figure 3. Petri net model obtained with HEURISTIC MINER.

Consider now the problem of constructing a Petri net using the dependency matrix. Assume a dependency threshold of 0.9. Then, the relationship between activities *a* and *e* is not represented because the value of the dependence between *a* and *e* is 0.83, below the threshold. Likewise, the relationship between activity *d* and itself is not represented. Furthermore, during the construction of the Petri net, one sometimes wants to express that particular transitions are not observable. Such transitions are called *silent* or *invisible* and are represented by black rectangles (VAN DER AALST, 2011).

3.1.3. Inductive Miner

The INDUCTIVE MINER guarantees sound process models. Instead of producing a Petri net, it builds a process tree (LEEMANS et al., 2013; LEEMANS et al., 2014). By repeatedly finding the most prominent split in the event log, the *detect* operator is executed, and continues on both sub logs until all logs are represented in the process tree.

For example, consider the event log:

L_4 = [<a, b, c, d, e, g>, <a, b, c, d, f, g>, <a, c, d, b, f, g>, <a, b, d, c, e, g>, <a, d, c, b, f, g>]

The first point that stands out is that all cases begin with activity *a*, so we can divide all cases into activity *a* and the rest, and the tree is represented with that activity in the first branch.

By analyzing the remaining subset, we can also see that all cases end with activity *g*. Accordingly, activity *g* is represented in the last branch of the tree.

The cases of the remaining subset are composed of 4 activities, that can be subdivided into two subsets: the first with 3 activities, and the second, with 1.

By analyzing the subset with a single activity, we can see that it is either *e* or *f*. So, when constructing the tree, we place the 'OR' function as the parent of these two branches of activities.

The subset with 3 activities cannot be further subdivided, but we see that all 3 activities can be executed in any order. So, we indicate that they can be executed in parallel, by placing them under an 'AND' function in the tree.

Finally, it is possible to construct a Petri net from the process tree, as shown in Figure 4.

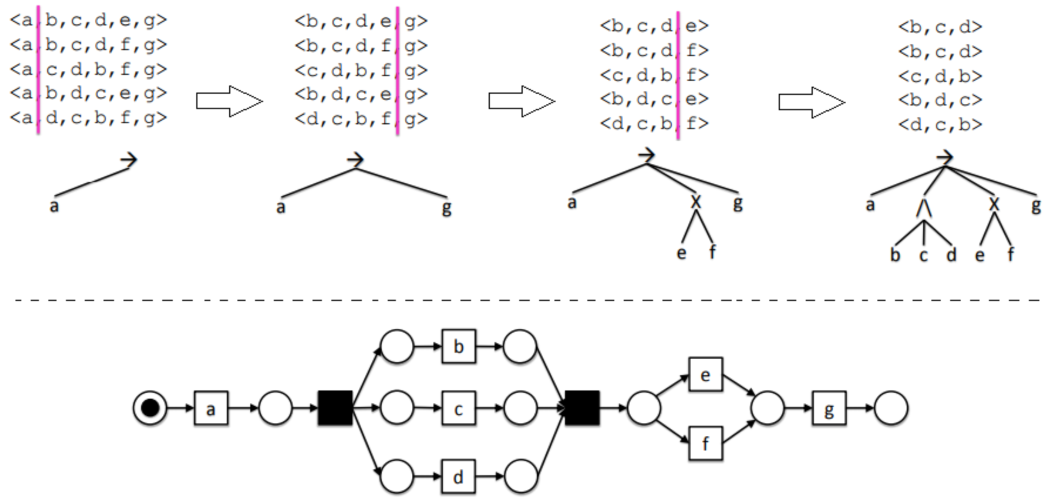


Figure 4. Process Tree and Petri net obtained with INDUCTIVE MINER.

3.1.4. Conformance Checking based on Alignment

The goal of CONFORMANCE CHECKING BASED ON ALIGNMENT is to make sure that the model and the logs are in agreement (VAN DER AALST & VERBEEK, 2014). By analyzing the real processes and diagnosing discrepancies, new insights can be obtained, which may show how to improve the information systems support.

The algorithm is quite simple. When it reaches a point of discrepancy, a deviation is annotated as a “move”, either in the model or in the log. At the end, it counts the number of deviations that were detected, and this reflects the cost of the alignment.

The best alignment between the model and the log depends on the cost function. The cost of doing a move in the model and in the log is usually 1, but in some tools this cost is defined by the user, who can assign penalties to activities that are being executed too late or by the wrong person.

Figure 5 illustrates an example where one tries to verify if the log <a, b, c, d, e, f, g> is represented by the Petri net shown. The first five activities are executed in both the log and the model, that is, the alignment between the log and the model is perfect. However, the Petri net does not allow executing activity *f* of the log. Indeed, activity *f* is not active, in the sense that it does not have a token in the previous place. For this reason, we mark a move in the model, and increase the cost of executing the log. Then, the final activity *g* is executed without problem. Hence, the cost of aligning the log with the model is 1.

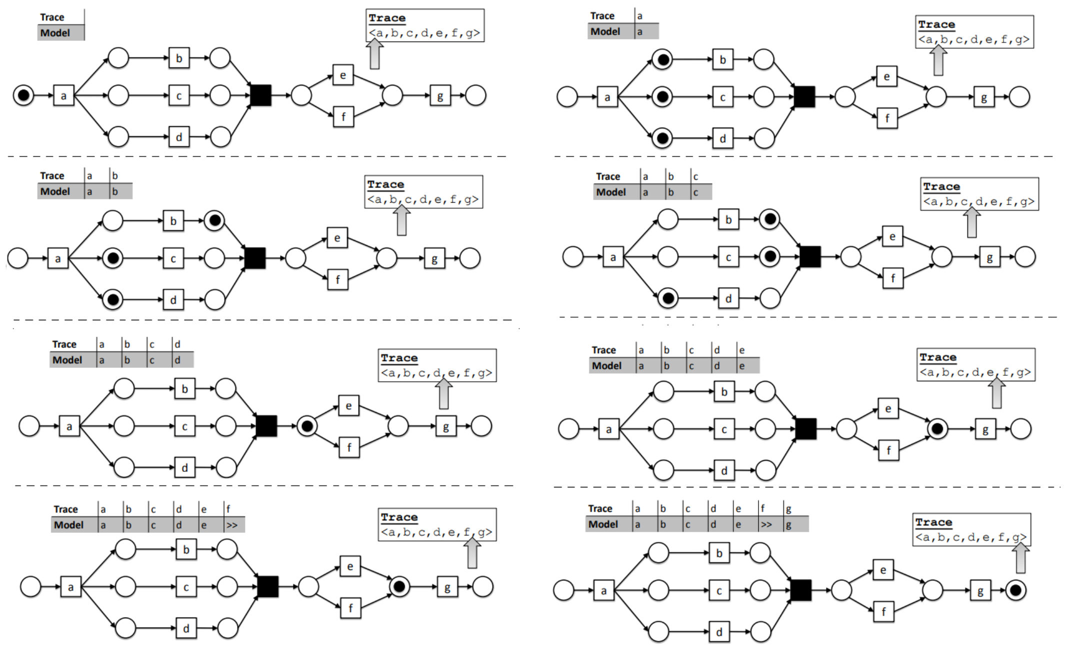


Figure 5. CONFORMANCE CHECKING BASED ON ALIGNMENT.

All in all, in the problems under investigation in this dissertation, as we have already argued in a paper related to our project (GOTTIN et al., 2017), process mining algorithms − such as alpha, heuristic, fuzzy and inductive, developed in software tools, such as ProM − proved not to be fully adequate, since the order in which students enroll in courses has to respect their pre-requisites, which are known in advance.

3.2 Frequent Itemsets Techniques

The frequent itemsets technique is based on finding, for all sets of a dataset, the frequencies with which each subset appears in the data set. (RAJARAMAN & ULLMAN, 2011; CHEUNG & FU, 2004; VO et al., 2016; ZAKI & JR, 2014). For example, if we have the set

S = [{1, 2, 3, 4}, {1, 2, 4}, {1, 2}, {2, 3, 4}, {2, 3}, {3, 4}, {2, 4}]

the frequencies for all possible subsets are:

{1} = 3, {2} = 6, {3} = 4, {4} = 5,

{1, 2} = 3, {1, 3} = 1, {1, 4} = 2, {2, 3} = 3, {2, 4} = 4, {3, 4} = 3,

{1, 2, 3} = 1, {1, 2, 4} = 2, {2, 3, 4} = 2,

{1, 2, 3, 4} = 1

A brute-force algorithm would create all possible combinations, and calculate the frequencies of each. For a small dataset, the cost of this operation might be low, but with a very large dataset, the creation of all possible combinations of itemsets will compromise the performance of the algorithm.

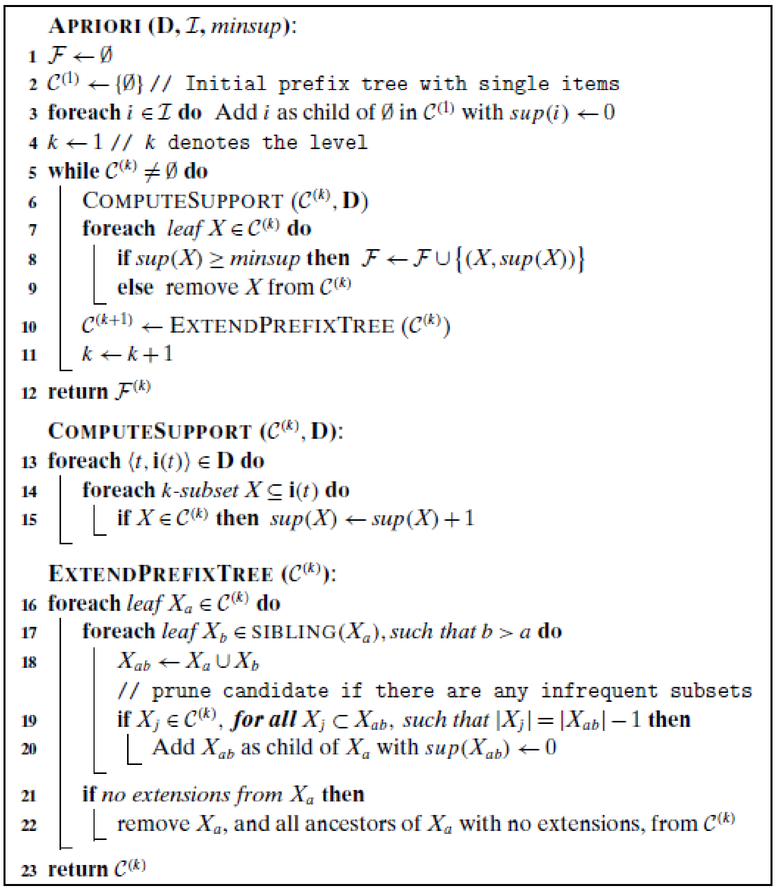
To avoid working with low frequency subsets that do not represent relevant information, a threshold can be defined, so that only those subsets that have a frequency greater than the threshold are considered.

Among the algorithms that improve the brute force algorithm, we find the APRIORI algorithm (RAJARAMAN & ULLMAN, 2011), which takes into account two properties to obtain this improvement (where X and Y any two itemsets):

1. If the frequency of X is greater than the threshold, then any frequency of a subset of X is also greater than the threshold.
2. If the frequency of X is smaller than the threshold, then any set containing X will have a lower frequency than the threshold.

Figure 6 presents the pseudo code of the APRIORI algorithm. Lines 5-11 constitute the main loop that computes the frequency of a subset $C^{(k)}$, and maintains $C^{(k)}$ only if its frequency is above a given threshold; if $C^{(k)}$ is discarded, all sets containing $C^{(k)}$ will also be discarded.

We note that there is a library in Python, called “pymining”, that includes a function called "itemmining", which is based on the APRIORI algorithm.



```
APRIORI ($\mathbf{D}, \mathcal{I}, minsup$):
$\mathcal{F} \leftarrow \emptyset$
$\mathcal{C}^{(1)} \leftarrow \{\emptyset\}$ // Initial prefix tree with single items
foreach $i \in \mathcal{I}$ do  Add $i$ as child of $\emptyset$ in $\mathcal{C}^{(1)}$ with $sup(i) \leftarrow 0$
$k \leftarrow 1$ // $k$ denotes the level
while $\mathcal{C}^{(k)} \neq \emptyset$ do
    COMPUTESUPPORT ($\mathcal{C}^{(k)}, \mathbf{D}$)
    foreach leaf $X \in \mathcal{C}^{(k)}$ do
        if $sup(X) \geq minsup$ then $\mathcal{F} \leftarrow \mathcal{F} \cup \{(X, sup(X))\}$
        else remove $X$ from $\mathcal{C}^{(k)}$
    $\mathcal{C}^{(k+1)} \leftarrow$ EXTENDPREFIXTREE ($\mathcal{C}^{(k)}$)
    $k \leftarrow k + 1$
return $\mathcal{F}^{(k)}$

COMPUTESUPPORT ($\mathcal{C}^{(k)}, \mathbf{D}$):
foreach $\langle t, \mathbf{i}(t) \rangle \in \mathbf{D}$ do
    foreach $k$-subset $X \subseteq \mathbf{i}(t)$ do
        if $X \in \mathcal{C}^{(k)}$ then $sup(X) \leftarrow sup(X) + 1$

EXTENDPREFIXTREE ($\mathcal{C}^{(k)}$):
foreach leaf $X_a \in \mathcal{C}^{(k)}$ do
    foreach leaf $X_b \in$ SIBLING$(X_a)$, such that $b > a$ do
        $X_{ab} \leftarrow X_a \cup X_b$
        // prune candidate if there are any infrequent subsets
        if $X_j \in \mathcal{C}^{(k)}$, for all $X_j \subset X_{ab}$, such that $|X_j| = |X_{ab}| - 1$ then
            Add $X_{ab}$ as child of $X_a$ with $sup(X_{ab}) \leftarrow 0$
    if no extensions from $X_a$ then
        remove $X_a$, and all ancestors of $X_a$ with no extensions, from $\mathcal{C}^{(k)}$
return $\mathcal{C}^{(k)}$
```

Figure 6. Pseudo code of APRIORI Algorithm (RAJARAMAN & ULLMAN, 2011).

If we apply this algorithm to the same set *S* that we have previously used as example:

S = [{1, 2, 3, 4}, {1, 2, 4}, {1, 2}, {2, 3, 4}, {2, 3}, {3, 4}, {2, 4}]

and establish a threshold value of 3, the resulting set of frequencies would be:

{1} = 3, {2} = 6, {3} = 4, {4} = 5,

{1, 2} = 3, {2, 3} = 3, {2, 4} = 4, {3, 4} = 3

Observing the results obtained by applying the APRIORI algorithm to the set *S*, we can see that subsets {1} and {1,2} have the same frequency, so that subset {1} can be eliminated from the answer, due to the presence of its superset {1,2} of equal frequency. We can then work only with *maximal* subsets (CHENG et al., 2008), defined as:

Let F be a set of subsets together with their frequencies. The set F is maximal iff, for any X in F, there is no Y in F such that $freq(Y) = freq(X)$ and $X \subset Y$.

Then, by applying the APRIORI algorithm with the maximal set rule, the final result for set S would be:

{2} = 6, {4} = 5, {3} = 4, {2, 4} = 4, {1, 2} = 3, {2, 3} = 3, {3, 4} = 3

Among the techniques for frequent itemsets analysis, one must cite hierarchical agglomerative clustering (RAJARAMAN & ULLMAN, 2011), applying, in special, the still well-accepted Ward's linkage method and using a tree diagram, known as *dendrogram*, to conveniently display the arrangement of the clusters (WARD, 1963; FERREIRA & HITCHCOCK, 2009). As has been argued in a recent paper of our project (GOTTIN et al., 2017), and will be treated in section 4.4.2. of the present dissertation, this technique proved to be particularly adequate to handle our example academic administration example.

4
Methodology and a Case Study on Academic Analytics

In this chapter, we will apply all algorithms defined in the previous chapter to a (real) set of student records. Section 4.1 introduces some basic questions about academic data. Section 4.2 presents the structure of the data, and how they will be stored. The final sections present the results obtained.

4.1 Methodology

4.1.1. Basic Statistical Analysis

When starting to analyze educational data, based on ideas given by specialists in the area (for example, the degree coordinators), some basic questions arise, such as:

Question (a). How many students we have, grouped by degrees' curricula and student status?

Question (b). For each degree, in which semester students more often dropout?

Question (c). For each degree subject, how much the total number of dropout, enrolled and graduated students differ in terms of students' performance?

Question (d). For each course, what are the approved and failed rates, according to the semester in which the students enroll in the course?

Question (e). For each course, how long do the students advance or delay enrolling in the course, with respect to the recommended semester, vis-à-vis the approval rates?

Simple statistical methods can be used to answer such basic questions, since courses can be analyzed independently of each other. However, some questions require analyzing sets of courses, one such question being:

Question (f). For a given semester, how closely do the courses that students typically enroll in come to match the set of courses that the degree curriculum recommends for the semester?

4.1.2. Process Mining Analysis

For this type of analysis, we intend to stop regarding the sequences of student records merely as data, and will begin to visualize them as process logs. In our case, following the conventions of the process mining literature, we treat the degree as a *process*, where students are the *cases* and the courses are the *activities*. Using process mining techniques, we are able to address the following questions:

Question (g). Do the students follow the recommended order of courses in their degree curriculum?

Question (h). What are the sets of courses that students most often take in a semester?

For the first question, we will use discovery and conformance techniques, whereas, for the second question, we will adopt frequent itemset techniques.

4.2 Data Organization and Processing

4.2.1. Data Sources

The data was obtained from the academic systems of PUC-Rio, duly anonymized for the sake of confidentiality.

The data comes from two different sources:

- Spreadsheet reports, in *.xlsx* format, for each degree in the university. They comprise: status; degree (code and name); major (code and name); emphasis (code and name); curriculum (year and code); student (ID, zip code and entrance punctuation (ENEM)); course (code, number of credits and class); year of enrollment; semester of enrollment; the student’s status in a course (canceled, passed or failed); and student’s academic status (enrolled, dropout or graduated); name of the professor; the final grade obtained in the course.

- Spreadsheet reports, in *.xlsx* format, for each degree curriculum in the university. They comprise: the recommended semester of a course; group by course (if any).

Derived data was generating by joining the spreadsheets, consisting of:

- The year and the semester in which the student was admitted to the university; the effective semester in which the student enrolled in a course; the difference between the effective semester and the recommended semester of the course; the number of times the student took the course.

We then converted the data into a *dimensional model* (KIMBALL & ROSS, 2013), shown in Figure 7.

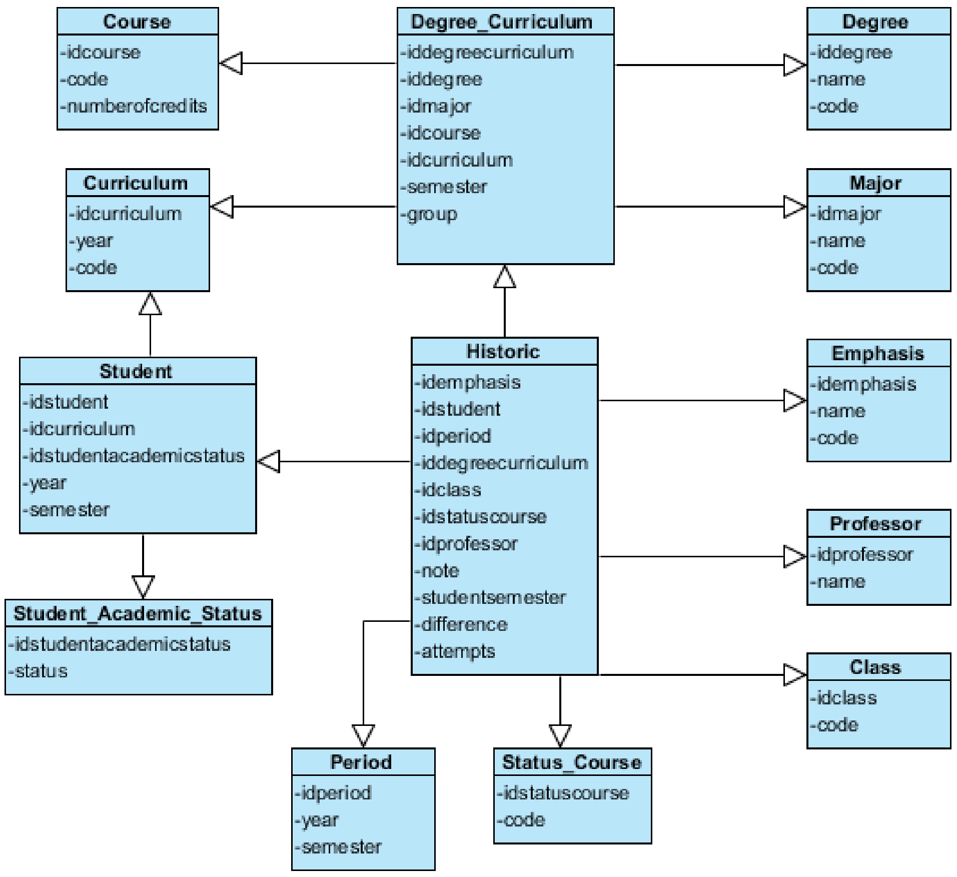


Figure 7. Dimensional model.

4.2.2. SGDA – Academic Data Management System

For the analysis of all such data, we developed an application, called Academic Data Management System – SGDA (in Portuguese: **S**istema de **G**erência de **Dados Acadêmicos**), described as follows. The dimensional model was implemented as a relational database, whose schema in shown in Figure 8, using the PostgreSQL database management system. The ETL (Extraction-Transform-Load) process to convert the original data to the relational database was implemented in Python.

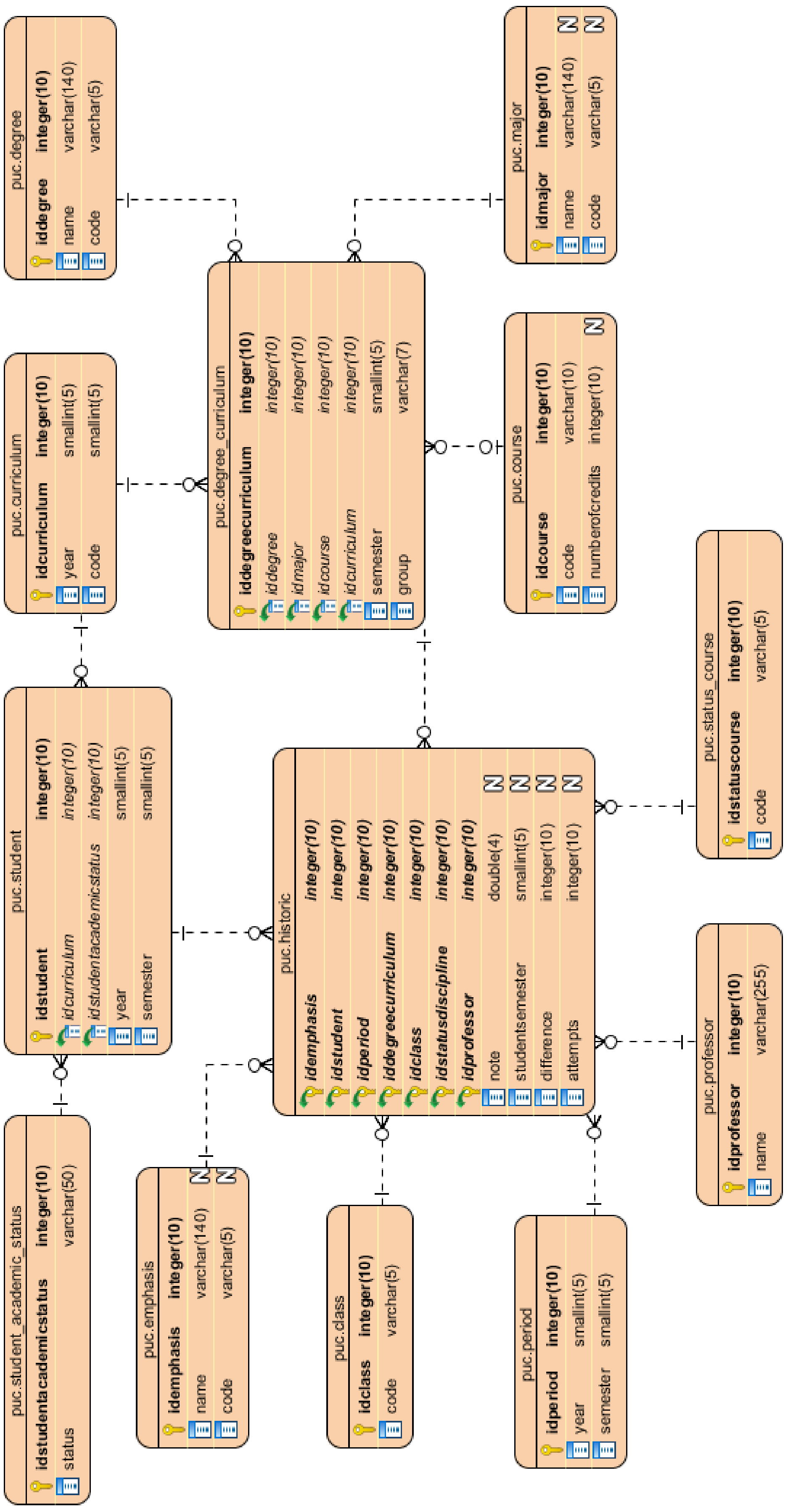


Figure 8. Architecture of the SGDA database

4.3 Basic Statistical Analysis - Results

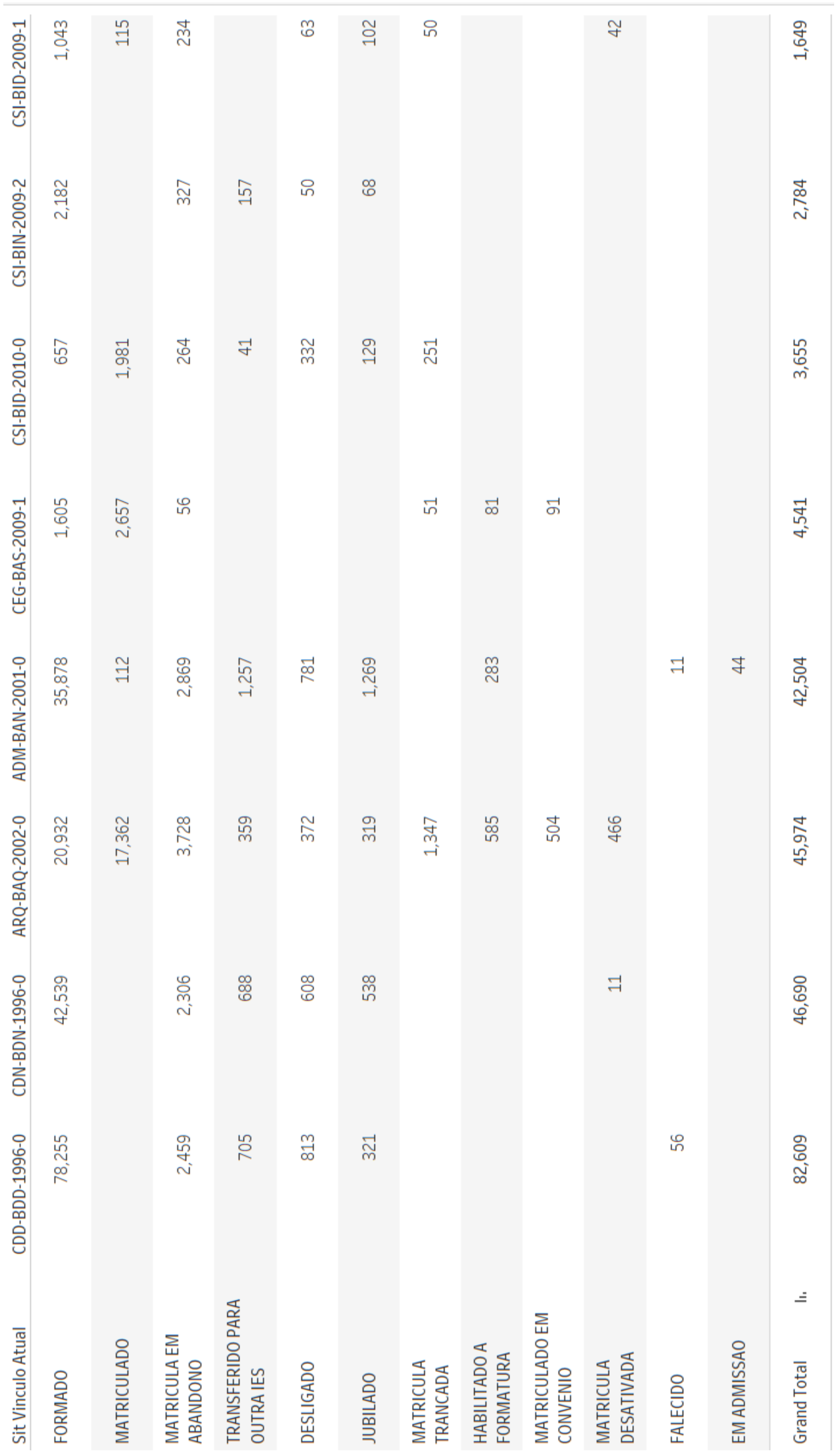
With the data already processed and stored, we answered the questions raised in section 4.1.

Consider the first question:

> **Question (a).** How many students we have, grouped by degrees' curricula and student status?

To answer this question, we constructed Table 1. By analyzing the data in Table 1, we observe that the degrees' curricula with more students are those of Law, Architecture and Administration, which have a large number of graduated students. Those of Engineering, on the contrary, tend to have a discrete number of students.

Table 1. Student distribution by degree and status.



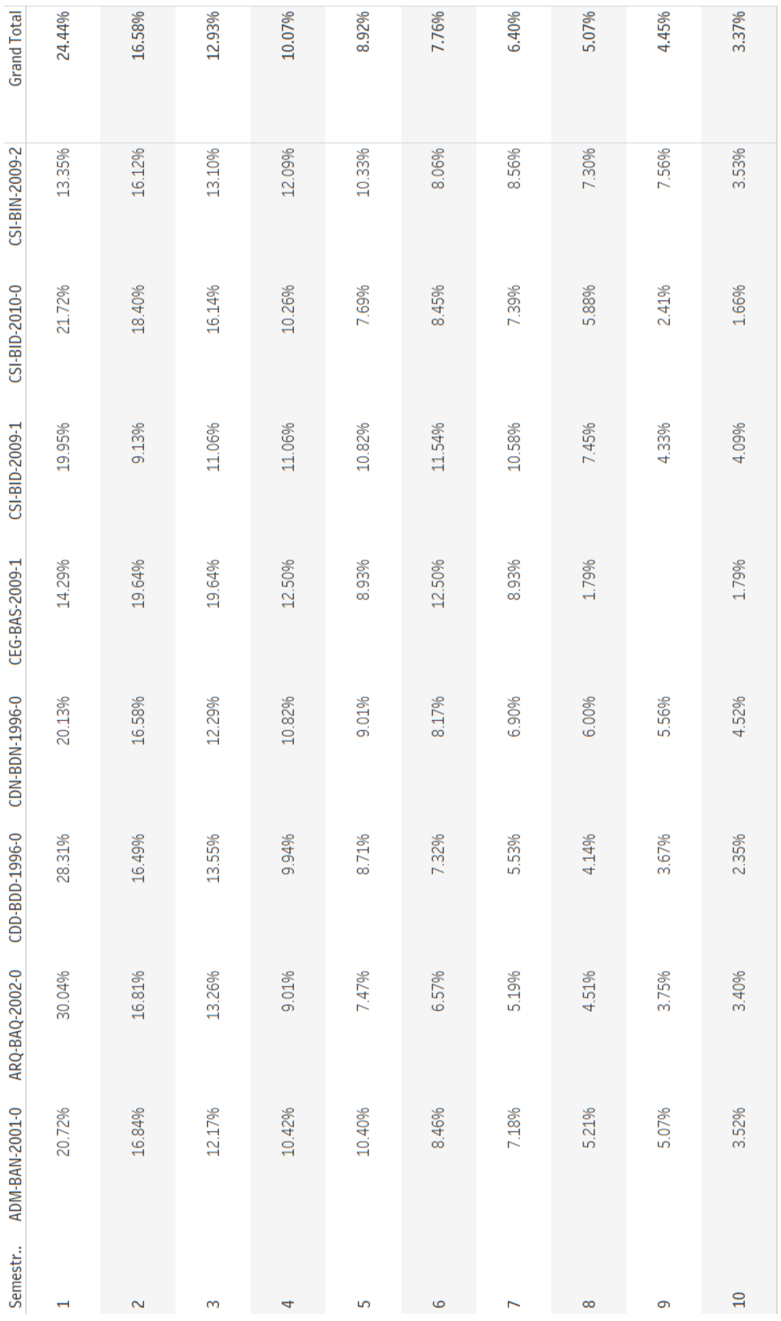
Sit Vinculo Atual	CDD-BDD-1996-0	CDN-BDN-1996-0	ARQ-BAQ-2002-0	ADM-BAN-2001-0	CEG-BAS-2009-1	CSI-BID-2010-0	CSI-BIN-2009-2	CSI-BID-2009-1
FORMADO	78,255	42,539	20,932	35,878	1,605	657	2,182	1,043
MATRICULADO			17,362	112	2,657	1,981		115
MATRICULA EM ABANDONO	2,459	2,306	3,728	2,869	56	264	327	234
TRANSFERIDO PARA OUTRA IES	705	688	359	1,257		41	157	
DESLIGADO	813	608	372	781		332	50	63
JUBILADO	321	538	319	1,269		129	68	102
MATRICULA TRANCADA			1,347		51	251		50
HABILITADO A FORMATURA			585	283	81			
MATRICULADO EM CONVENIO			504		91			
MATRICULA DESATIVADA		11	466					42
FALECIDO	56			11				
EM ADMISSAO				44				
Grand Total	82,609	46,690	45,974	42,504	4,541	3,655	2,784	1,649

Consider now the second question:

> **Question (b).** For each degree, in what semester students more often dropout?

To answer this question, we constructed Table 2. Observing Table 2, we find that the most critical semesters are the first two.

Table 2. Percentage of dropout by degree and semester.



Semestr..	ADM-BAN-2001-0	ARQ-BAQ-2002-0	CDD-BDD-1996-0	CDN-BDN-1996-0	CEG-BAS-2009-1	CSI-BID-2009-1	CSI-BID-2010-0	CSI-BIN-2009-2	Grand Total
1	20.72%	30.04%	28.31%	20.13%	14.29%	19.95%	21.72%	13.35%	24.44%
2	16.84%	16.81%	16.49%	16.58%	19.64%	9.13%	18.40%	16.12%	16.58%
3	12.17%	13.26%	13.55%	12.29%	19.64%	11.06%	16.14%	13.10%	12.93%
4	10.42%	9.01%	9.94%	10.82%	12.50%	11.06%	10.26%	12.09%	10.07%
5	10.40%	7.47%	8.71%	9.01%	8.93%	10.82%	7.69%	10.33%	8.92%
6	8.46%	6.57%	7.32%	8.17%	12.50%	11.54%	8.45%	8.06%	7.76%
7	7.18%	5.19%	5.53%	6.90%	8.93%	10.58%	7.39%	8.56%	6.40%
8	5.21%	4.51%	4.14%	6.00%	1.79%	7.45%	5.88%	7.30%	5.07%
9	5.07%	3.75%	3.67%	5.56%		4.33%	2.41%	7.56%	4.45%
10	3.52%	3.40%	2.35%	4.52%	1.79%	4.09%	1.66%	3.53%	3.37%

As for the third question:

Question (c). For each degree subject, how the total number of dropout, enrolled and graduated students differ in terms of students' performance?

In order to better understand student performance, we executed some exploratory tasks, which resulted in preliminary visualizations. This kind of visualization allows us to compare the general performance of the student categories – *enrolled*, *dropout* or *graduated*.

As an example, Figure 9 shows a normalized histogram of the average grades for the Architecture degree. Note that, for the 3 groups, grades below 5.0 are less frequent, with the exception of grade 0.0, which is typically given to students who do not cancel their enrollment, but fail the course, either by not obtaining a passing grade or by exceeding the allowed number of absences.

With a general idea of the profile of the students in the degree, we proceeded to compare the dropout and graduated students with the current enrolled students. We use two different metrics: The Euclidean distance (WIKIPEDIA, 2017c) and the Jensen-Shannon divergence (WIKIPEDIA, 2017a). For the Architecture degree, Figure 9 shows that the pattern displayed for the currently enrolled students is quite similar to that of the graduated students. Indeed, using Euclidian distance, we obtain that the distance between the students who are enrolled in the course with those who dropout is 0.1016, and the distance with those already graduated is 0.0290. As the first value is larger, we can confirm that the distribution of the enrolled students is closer to that of the graduated students than that of the dropout students. To further validate this statement, we also used the Jensen-Shannon metric, and obtained 0.0226 and 0.0027, respectively, which reinforces the conclusion.

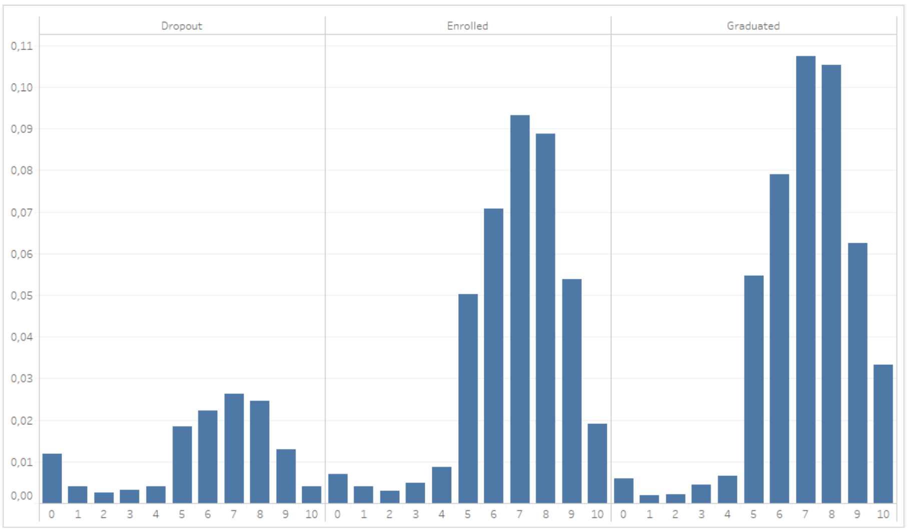


Figure 9. Normalized histogram of ARC students' grades.

Consider now the fourth question, which is of particular interest for the stakeholders and managers of the system:

> **Question (d).** For each course, what are the approved and failed rates, according to the semester in which the students enroll in the course?

We chose the following visualization scheme to display the answer:

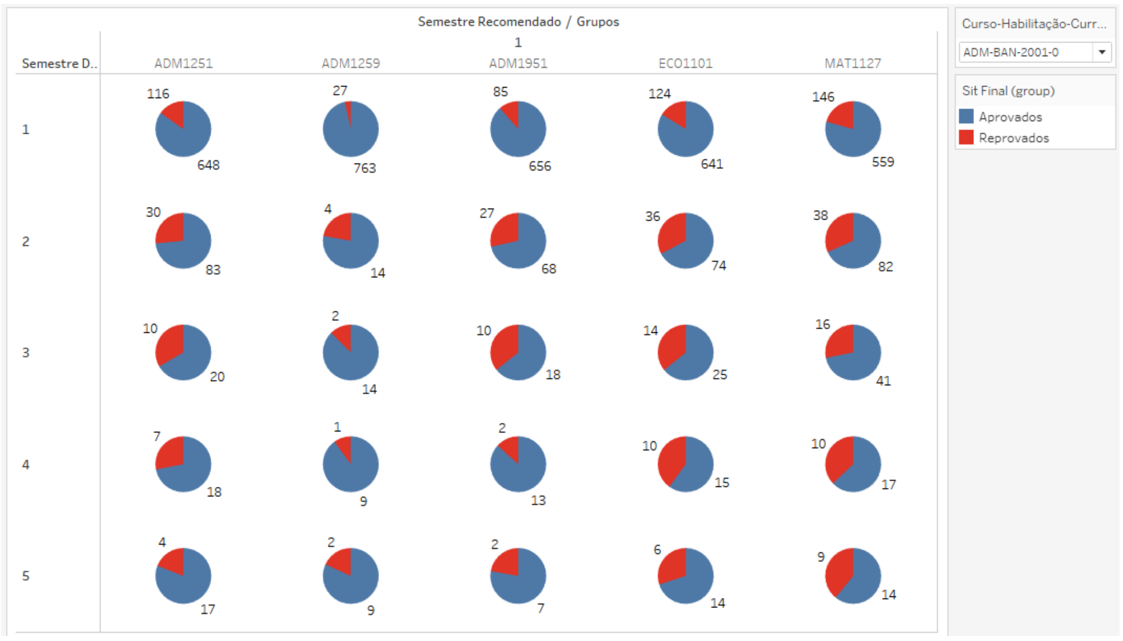


Figure 10. Percentage of approval and failure by course and semester.

Figure 10 depicts 5 courses recommended for the first semester of the Administration degree. The pie charts represent the ratio between approved (blue) and failed (red) students in each course, when taken in the effective semester registered in the respective row. The numbers state the number of observations of each status in the pie chart. Despite the low number of observations for longer delays, we can clearly see that the approval rates for these courses are higher when they are taken in the recommended first semester. This is an indication that the curriculum correctly prioritizes these courses. Besides supporting this kind of analysis, the diagram gives us a general view of students' performance throughout the degree.

Consider now the fifth question:

Question (e). For each course, how long do students advance or delay enrolling in the course, with respect to the recommended semester, vis-à-vis the approval rates?

To answer this question, we adopted the visualization strategy illustrated in Figures 11 and 12. It represents the proportion of enrollments in each course recommended for a given semester. The size of the square in each cell gives the percentage of students that advanced enrollment in that course (columns marked with a negative integer) or delayed enrollment (columns marked with a positive integer). The color also marks the amount of delay or advance, according to the legend.

For the Law degree, Figure 11 indicates that there are 3 courses recommended for the 7th and 8th semesters in which students very frequently advance their enrollment – a fact that should be further investigated, as it suggests that changing the order of the courses of the current Law degree curriculum might better suit the students' demands.

In contrast, the Information Systems degree visualization (Figure 12) shows that the students quite frequently tend to delay enrolling in the courses recommended for the first semesters.

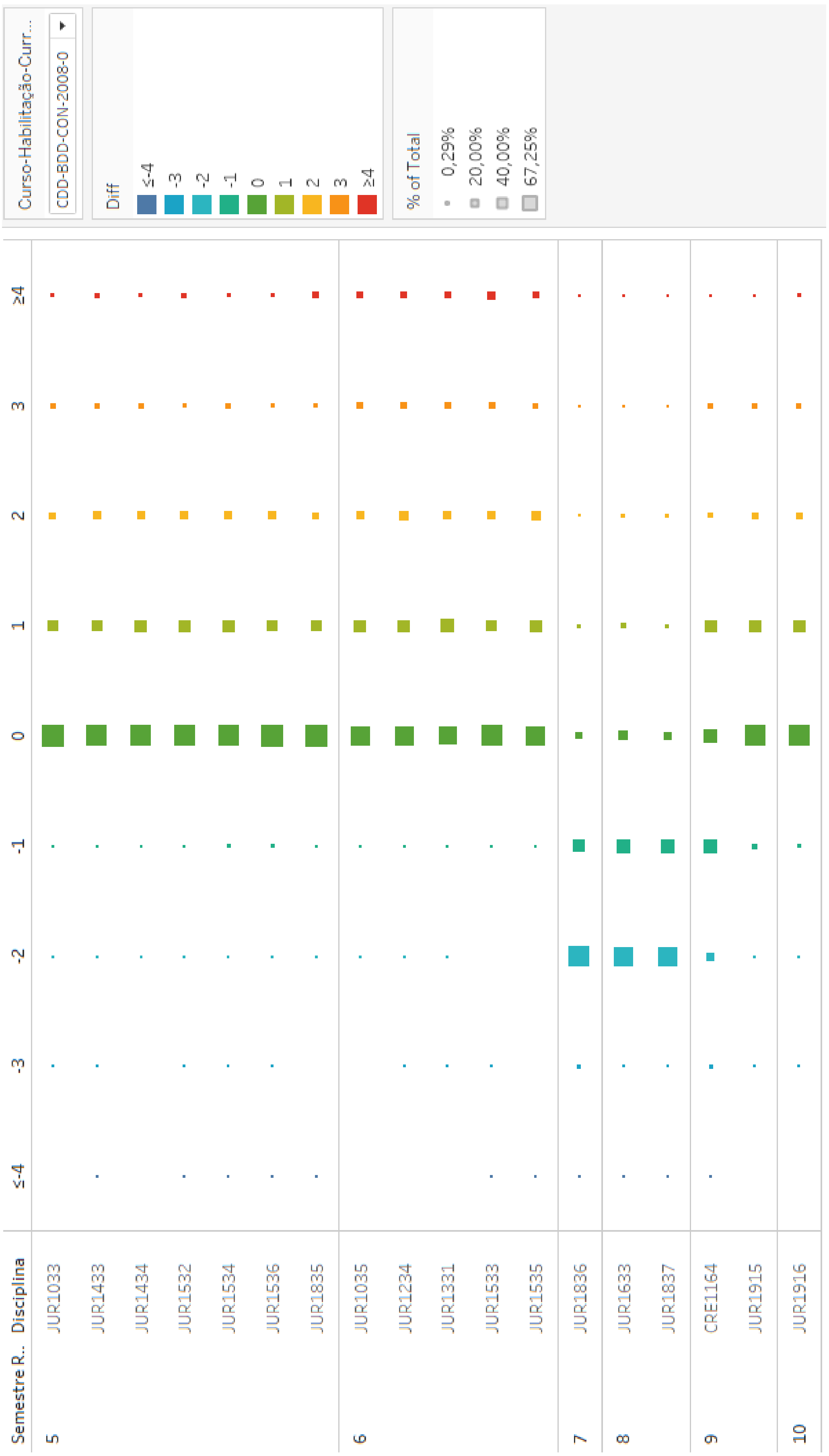


Figure 11. Analysis of student enrollment and performance in the Law degree.

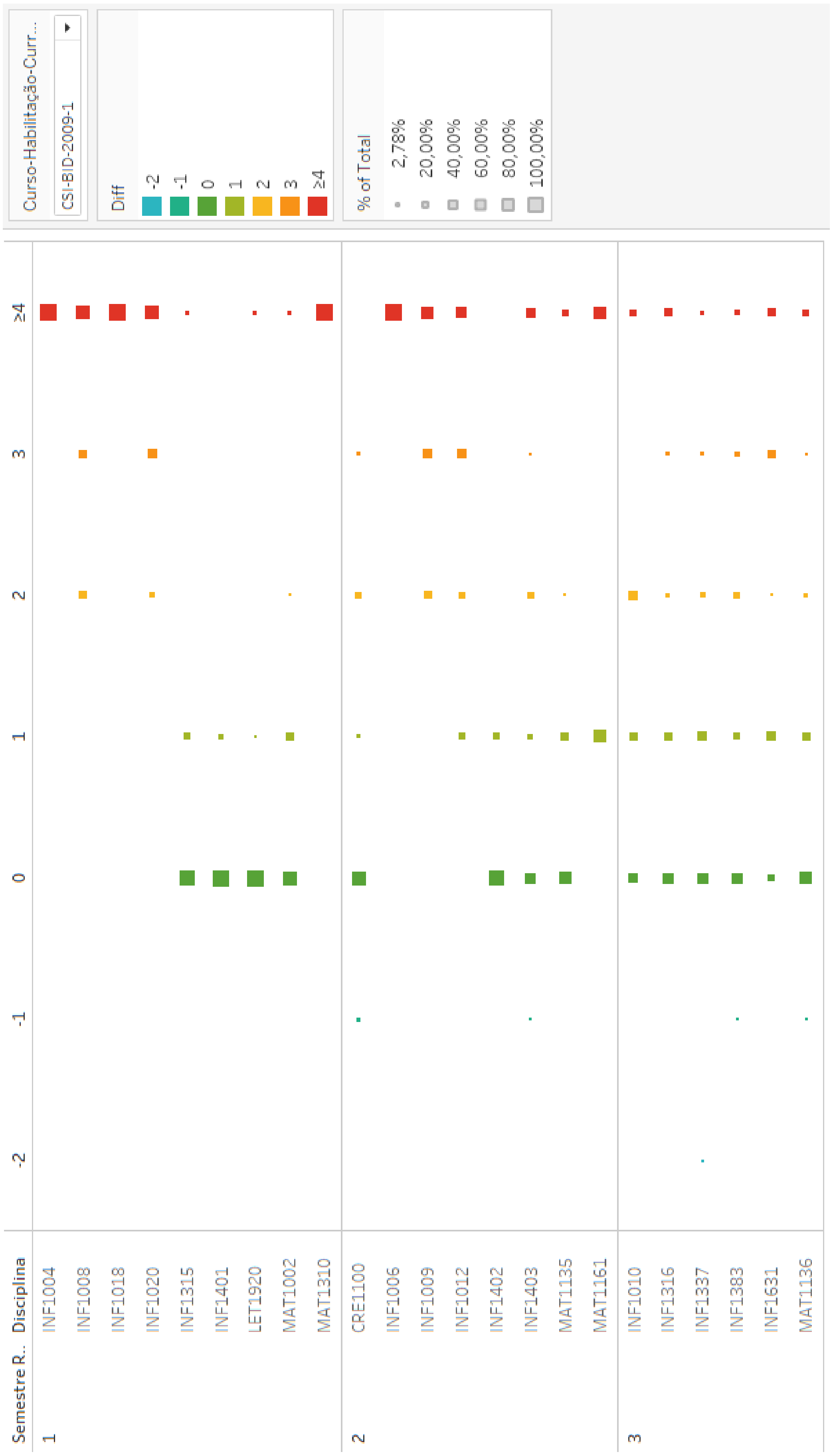


Figure 12. Analysis of student enrollment and performance in the Information Systems degree.

Consider now the sixth question:

Question (f). For a given semester, how closely do the courses that students typically enroll in come to match the set of courses that the degree curriculum recommends for the semester?

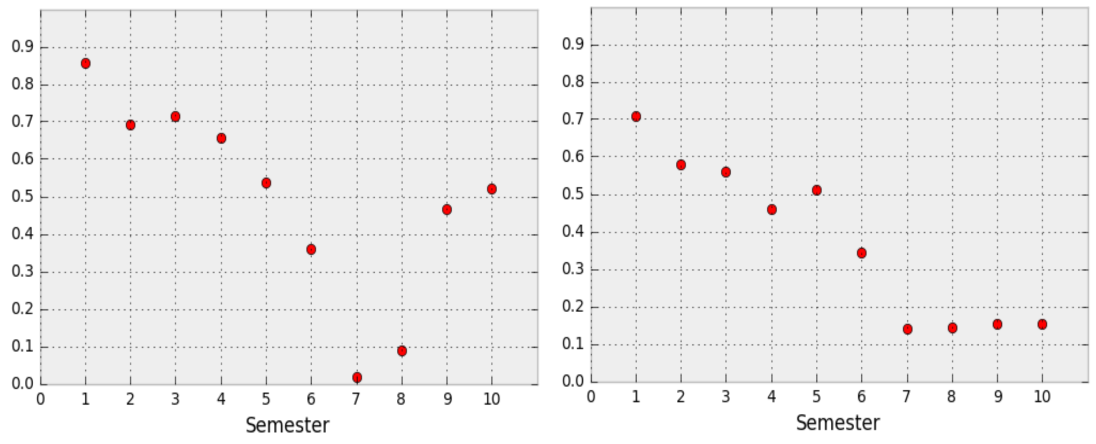
One way to answer this question is to calculate an index that, for each given semester, directly compares the set of courses in which the students enrolled (including those which they failed) with the set of courses recommended for the semester. More precisely, let *ST* be a given set of students enrolled in a degree *C* over a period of time *T*, expressed in semesters. The *degree-semester adherence index*, denoted $I_{C,S}$, evaluates how closely the students in *ST* followed the recommended set of courses *R* for degree *C* at a given semester *S* in *T*. It is defined as:

$$I_{C,S} = \frac{1}{n}\sum_{i=1}^{n}\frac{|E_i \cap R|}{|E_i \cup R|}$$

where: *n* is the number of students in the set of students *ST* enrolled in degree *C* at semester *S*; E_i is the set of courses in which student *i* in *ST* enrolled at semester *S*; and *R* is the set of courses recommended for degree *C* at semester *S*. Note that the fraction in the summation measures the *Jaccard similarity* (RAJARAMAN & ULLMAN, 2011) between E_i and *R*.

Also note that $I_{C,S} \in [0,1]$, where $I_{C,S} = 0$ iff no student enrolls in any of the recommended courses for degree *C* at semester *S*, and $I_{C,S} = 1$ iff all students enroll in exactly the recommended courses. The *degree adherence index* would then be defined as the average of the degree-semester adherence indices for the semesters of the degree.

Figure 13 illustrates the degree-semester adherence indices for the Law degree (a) and for the Architecture degree (b). For both degrees, a sharp drop is observed in the 7th semester, which means that in this semester students are not following the recommended courses. Figure 11, shown before when discussing question (e), corroborates this behavior for the Law degree, and the conclusion is that the low index value for this semester is due to the fact that students very often advance enrollment in the (single) recommended course, JUR1836. One difference between the 2 degrees is the behavior in the last 2 semesters: while students frequently follow the recommended courses for the Law degree, indicated by a high index for these semesters, this is not true for the Architecture degree. One possible explanation would be that Architecture students take more than the 10 recommended semesters to graduate.



(a) (b)

Figure 13. Degree-semester adherence index. (a) Law degree. (b) Architecture degree.

The overall degree adherence indices are 0.49 for Law and 0.37 for Architecture, both of which are fairly low, and therefore suggest that the academic coordinators should indeed rethink the current recommended curricula.

4.4 Process Mining Analysis - Results

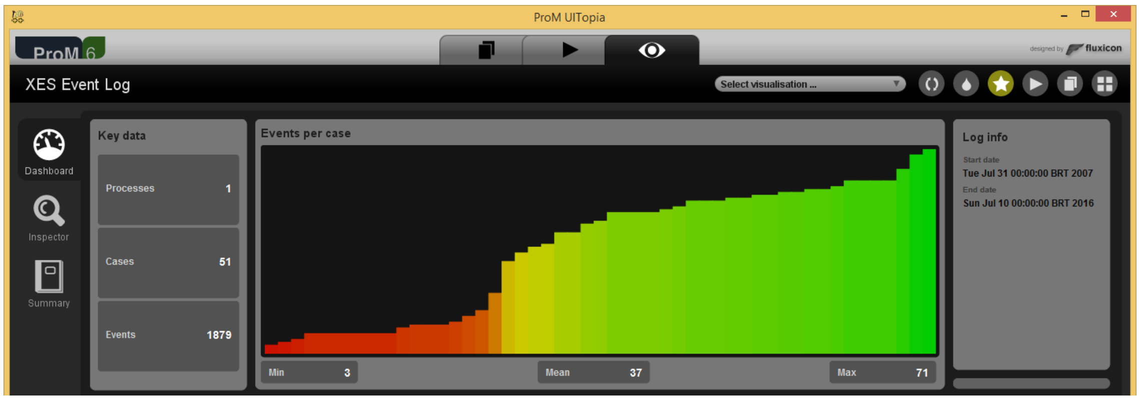
4.4.1. PROM

Question (g). Do the students follow the recommended order of courses in their degree curriculum?

To answer this question, we first utilized the free tool ProM (EINDHOVEN UNIVERSITY OF TECHNOLOGY), which is widely used for process mining, and implements most of the algorithms that we surveyed in Chapter 3: Alpha miner (with all its extensions), Heuristic miner, Inductive miner, and Conformance Checking based on Alignment. We ran the discovery algorithms and analyzed the results for the 2009-1 Information Systems grade curriculum.

The degree data used in our experiments has 1,777 records, conveying information about 51 students. As process mining algorithms need an initial and a final task, we added a registration task and a final status task for each student, thereby increasing the number of records to 1,879.

Figure 14 shows the data load image generated by the ProM tool.



Key data

Processes	1
Cases	51
Events	1879

Min 3 Mean 37 Max 71

Log info
Start date
Tue Jul 31 00:00:00 BRT 2007
End date
Sun Jul 10 00:00:00 BRT 2016

Figure 14. Summary of data load for ProM.

After loading the data, we executed the algorithms available in ProM. As Figure 15 shows, from the result of ALPHA MINER it is impossible to draw any conclusion. Even after applying some of the extensions, the results are not improved.

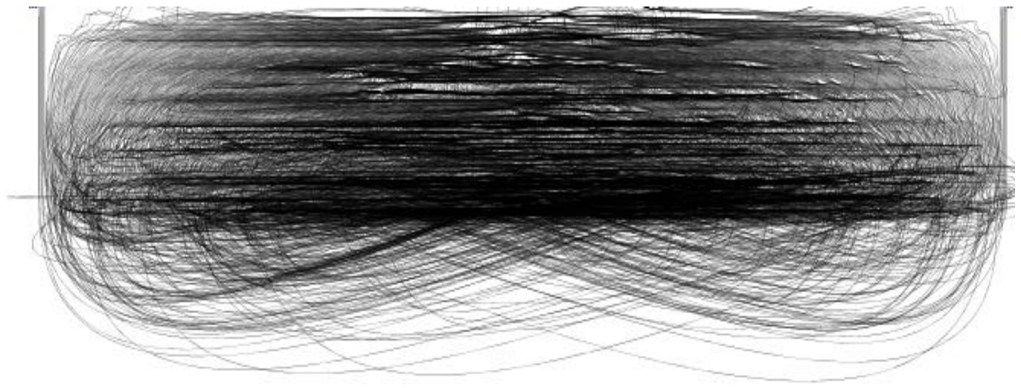


Figure 15. Results of ALPHA MINER.

HEURISTIC MINER produces better results (Figure 16), since it is possible to observe the flow paths that students take during the degree. However, by analyzing the results, we were able to verify that courses of the same semester are not being represented in parallel (Figure 16(a)) as they should, but sequentially, even though they have the same dates. For ProM to take into account the parallelism that exists between the courses that are taken in the same semester, it would be necessary to modify the logs by arranging all courses of the same semester in all possible ways, which is not a viable solution, given the amount of data.

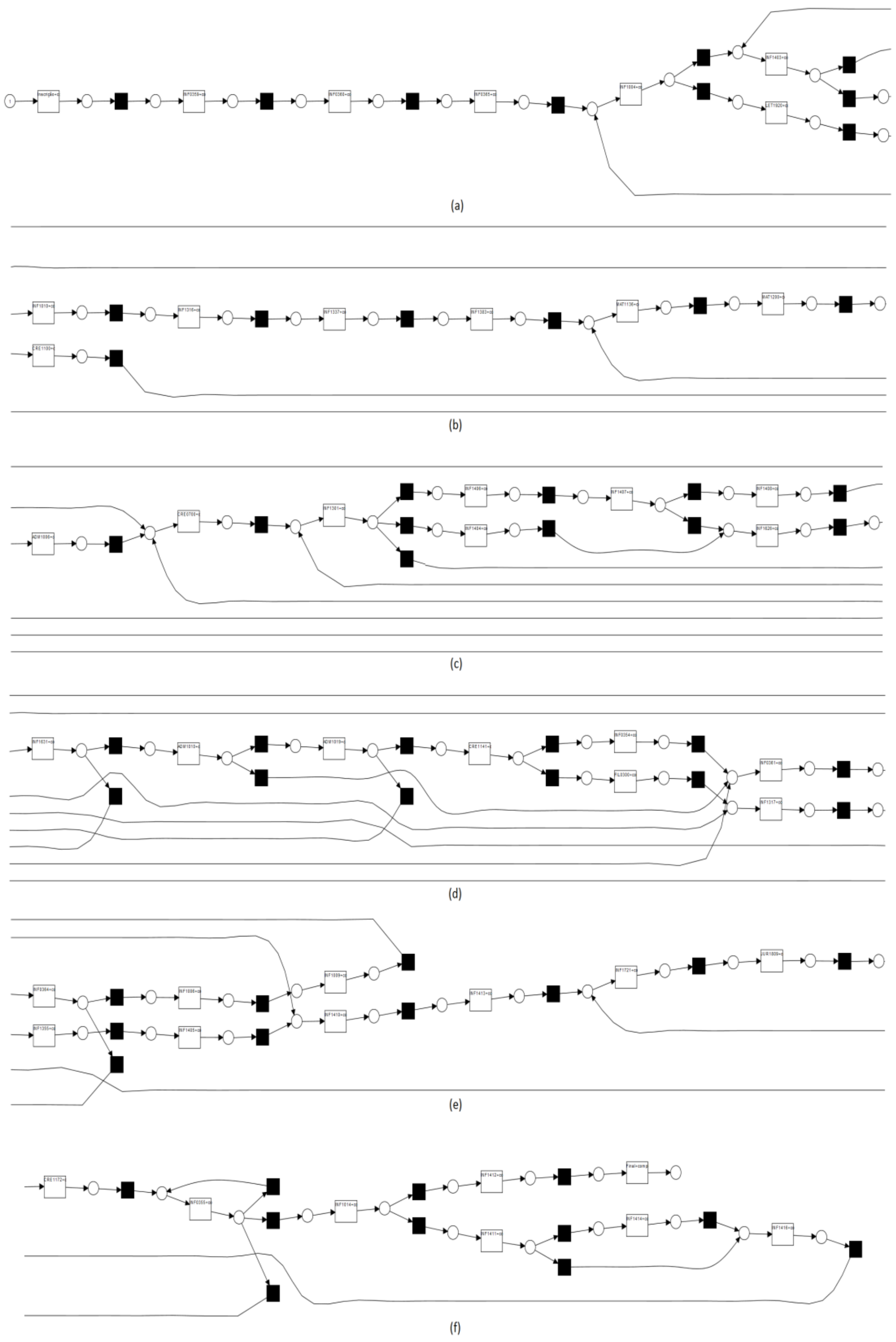


Figure 16. Results of HEURISTIC MINER.

The result of INDUCTIVE MINER (Figure 17) is a decision tree. Ideally, to answer our question, each child branch set should represent a semester. However, we see that most parent function nodes have the tag 'XOR', which means that only one of the disciplines of the child branches have been chosen, so we could not tell whether the students are following the recommended order.

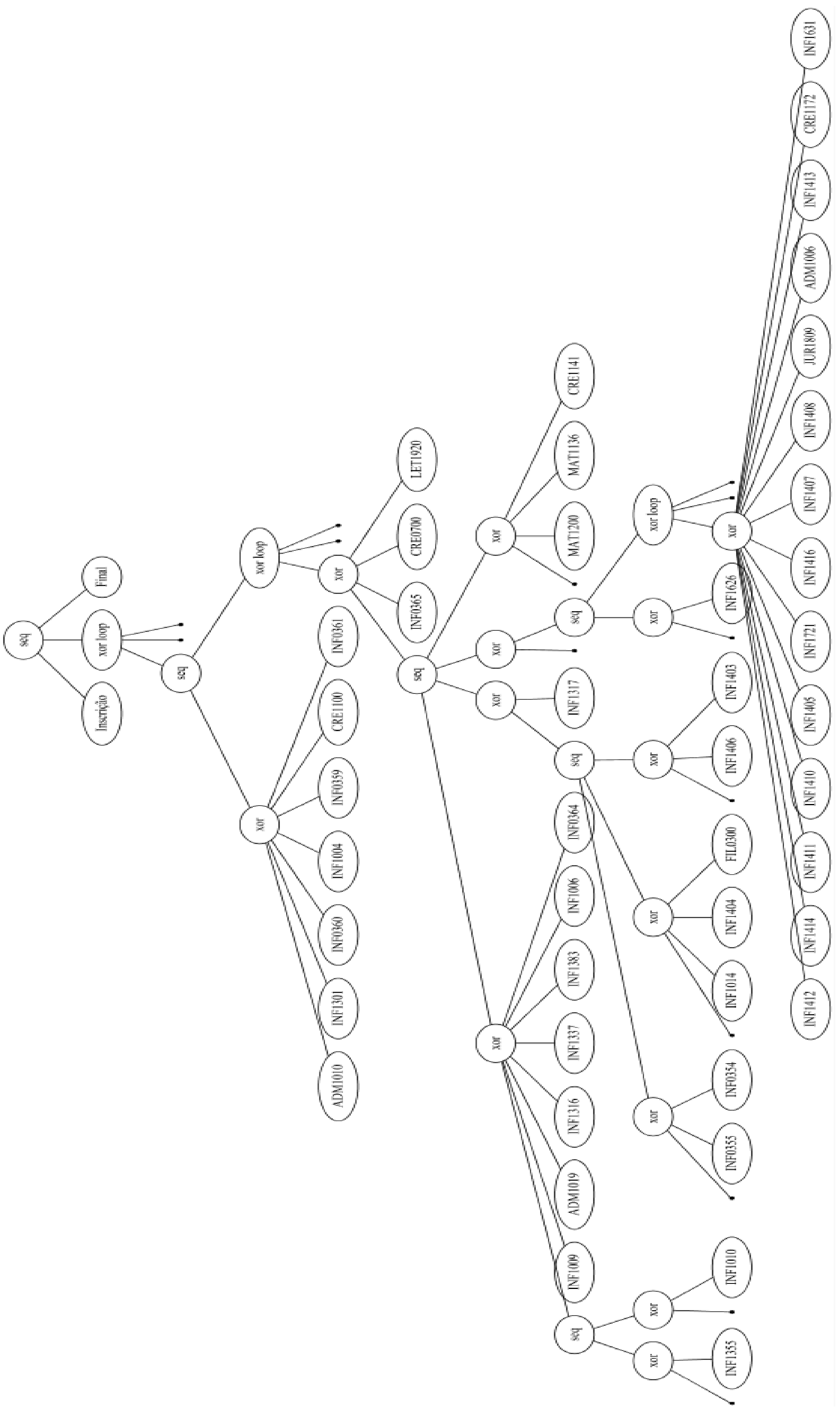


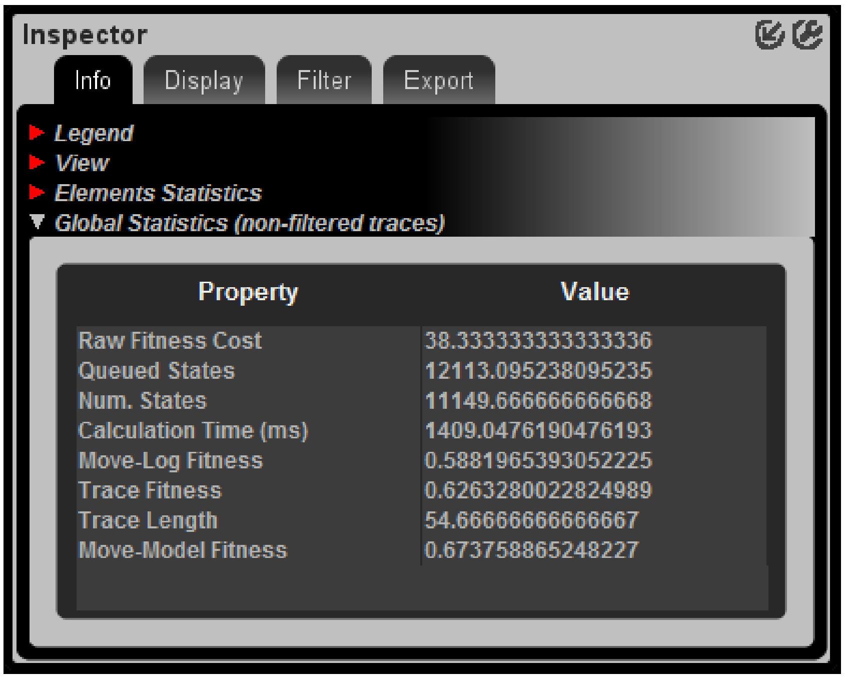
Figure 17. Results of INDUCTIVE MINER.

We next decided to execute these algorithms with different subsets of data, for example, only students that were approved in a course. But still, even with this restricted choice of data, ProM would not parallelize courses in which a student enrolled in during the same semester. Such inadequacies finally led us to realize that using the process discovery algorithms available in ProM did not seem to be always a convenient option for our kind of application.

We also analyzed the results obtained by CONFORMANCE CHECKING BASED ON ALIGNMENT. In addition to the logs, we started with a Petri net model, created by representing courses and the semesters for which they are recommended. As the result of the algorithm, the number of times an activity was executed or not was indicated in each transition of the Petri net.

This algorithm was applied only to the students that graduated, and consequently would have taken all the courses defined in the curriculum. Indeed, taking into account students that had not completed the degree would lead to less satisfactory results in terms of fitness, since all courses that such students did not take would trivially fail to be in correspondence with the full model, and would unnecessarily increase the cost of the process. The data we used consisted of 21 students and 1,148 records.

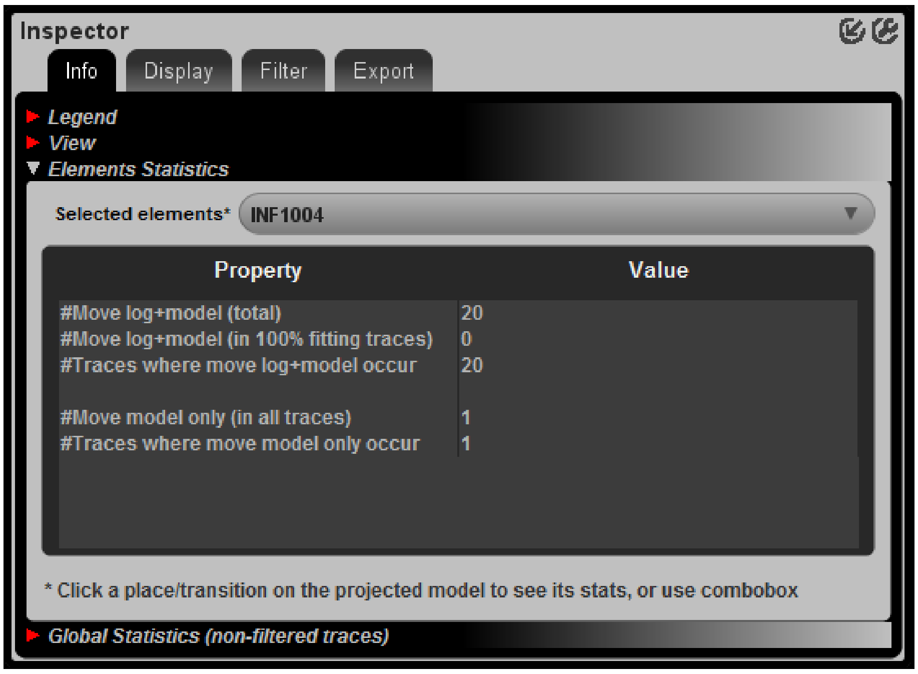
We obtained a Petri net with a fitness of 0.6263 (cf. the results as shown in tabular form in Figure 18). Ideally this value should be close to 1. A better score would possibly be obtained if we tried to analyze in depth each of the transitions and places.



Property	Value
Raw Fitness Cost	38.333333333333336
Queued States	12113.095238095235
Num. States	11149.666666666668
Calculation Time (ms)	1409.0476190476193
Move-Log Fitness	0.5881965393052225
Trace Fitness	0.6263280022824989
Trace Length	54.66666666666667
Move-Model Fitness	0.673758865248227

Figure 18. Results of CONFORMANCE CHECKING BASED ON ALIGNMENT.

Consider course INF1004, which belongs to the first semester. As Figure 19(a) shows, the course was taken by 20 students in the first semester, and only 1 student delayed enrolling. By analyzing the places (Figure 19(b)), we can see, for the same course INF1004, that in 6 occasions some student re-enrolled in this course (because he failed or cancelled the course in a previous semester), which is not allowed in the model.

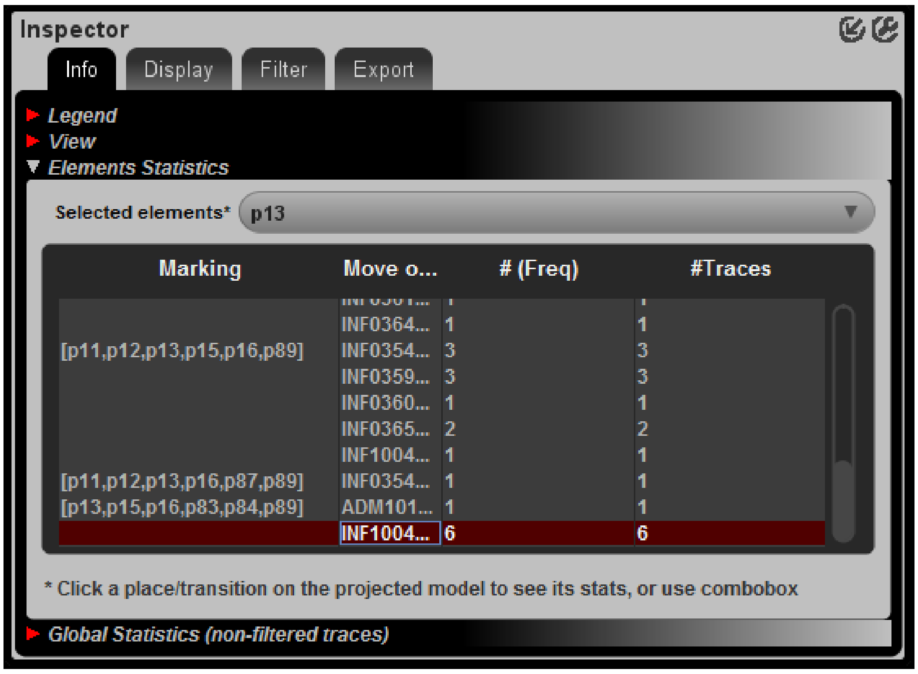


Selected elements* INF1004

Property	Value
#Move log+model (total)	20
#Move log+model (in 100% fitting traces)	0
#Traces where move log+model occur	20
#Move model only (in all traces)	1
#Traces where move model only occur	1

* Click a place/transition on the projected model to see its stats, or use combobox

(a)



Selected elements* p13

Marking	Move o...	# (Freq)	#Traces
	[illegible]	1	1
	INF0364...	1	1
[p11,p12,p13,p15,p16,p89]	INF0354...	3	3
	INF0359...	3	3
	INF0360...	1	1
	INF0365...	2	2
	INF1004...	1	1
[p11,p12,p13,p16,p87,p89]	INF0354...	1	1
[p13,p15,p16,p83,p84,p89]	ADM101...	1	1
	INF1004...	6	6

* Click a place/transition on the projected model to see its stats, or use combobox

(b)

Figure 19. Details of (a) Transformation INF1004, (b) Place p13.

To improve the fitness value, it would be necessary to change the model so that we would be able to analyze, for each course, if the students tend to delay the enrollment or re-enroll beyond the limit, which would further contribute to answering Question (e). In addition to that, we concluded that the discovery and conformance techniques are inadequate to address Question (f).

4.4.2. Frequent Itemsets Techniques

We applied the frequent itemsets techniques (RAJARAMAN & ULLMAN, 2011; CHEUNG & FU, 2004; VO et al., 2016; ZAKI & JR, 2014) to answer **Question (h)**: What are the sets of courses that students most often take in a semester?

Before using the library “pymining”, implemented in Python, to get all the subsets and their frequencies, we determined which specific data set would be in turn analyzed. Having selected a particular course, we then separately applied the **Apriori** algorithm for each of the 3 course statuses in a semester – *enrolled, approved or failed* – for each semester and degree.

We started by applying the algorithm to the Law Degree, Litigation emphasis, first semester, and enrolled students. Figure 20 shows the results obtained.

As a result, all frequencies were obtained for each of the possible subsets of the initial data set. Analyzing the data, the first thing we observe is that we have several sets that are subsets of other sets, all having the same frequency. In such cases, keeping the smaller sets does not add information, so we eliminated them without influencing the results. For example, we have {HIS0201, SOC0202} and {HIS0201, SOC0202, JUR1830}, both with frequency 305; so, we can eliminate {HIS0201, SOC0202}, considering that it is subsumed by the other set.

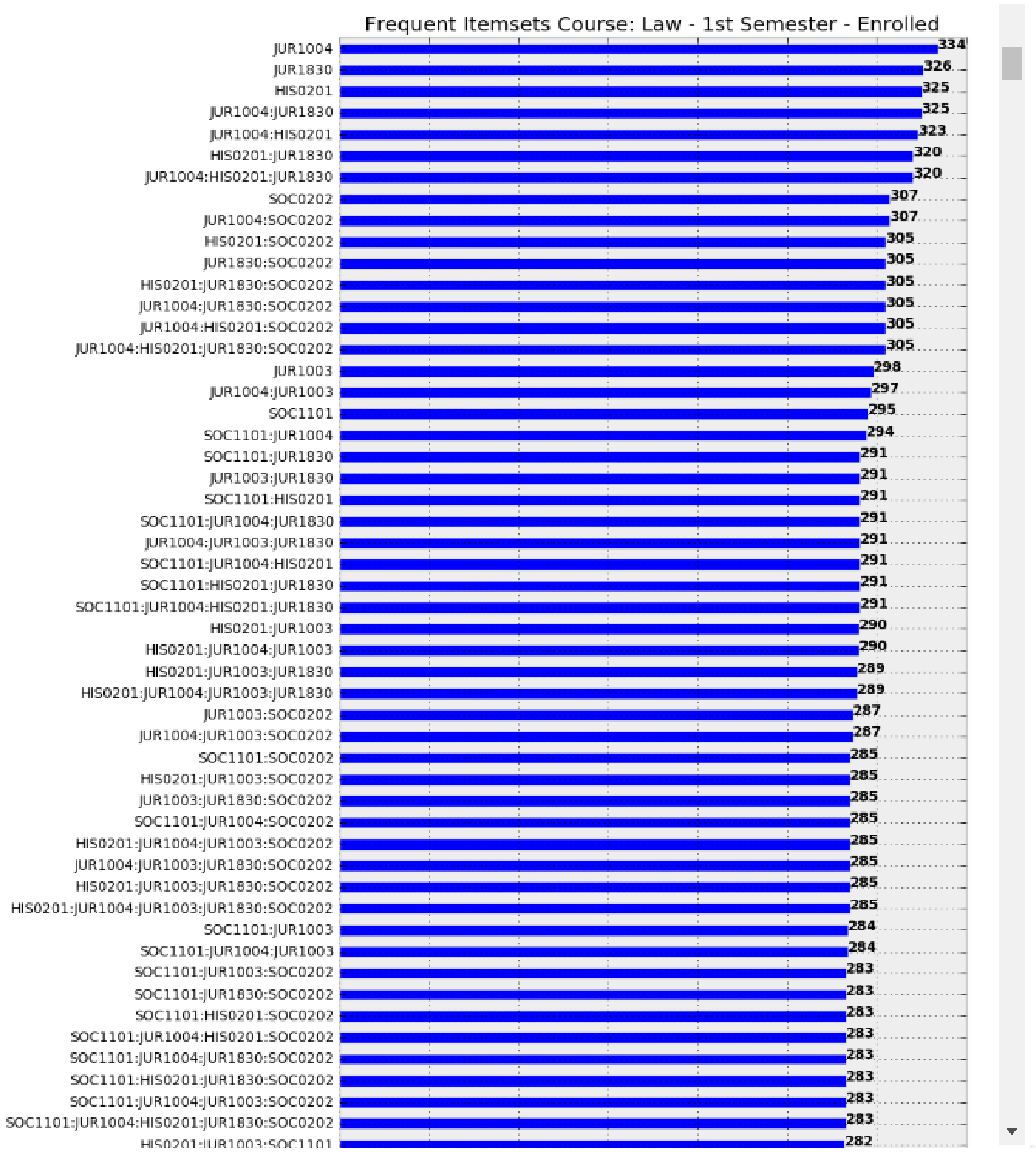


Figure 20. Results of APRIORI algorithm.

Figure 21 shows other interesting analyses. Observe the small difference in frequency values between some sets and their subsets. Indeed, a difference in frequency wherever a negligible number of students is involved does not modify the final analysis. We decided, therefore, to also discard these smaller subsets. For example, since we have {SOC1101}, with a frequency of 295, and {SOC1101, JUR1004}, with 294, the subset {SOC1101} can be safely eliminated.

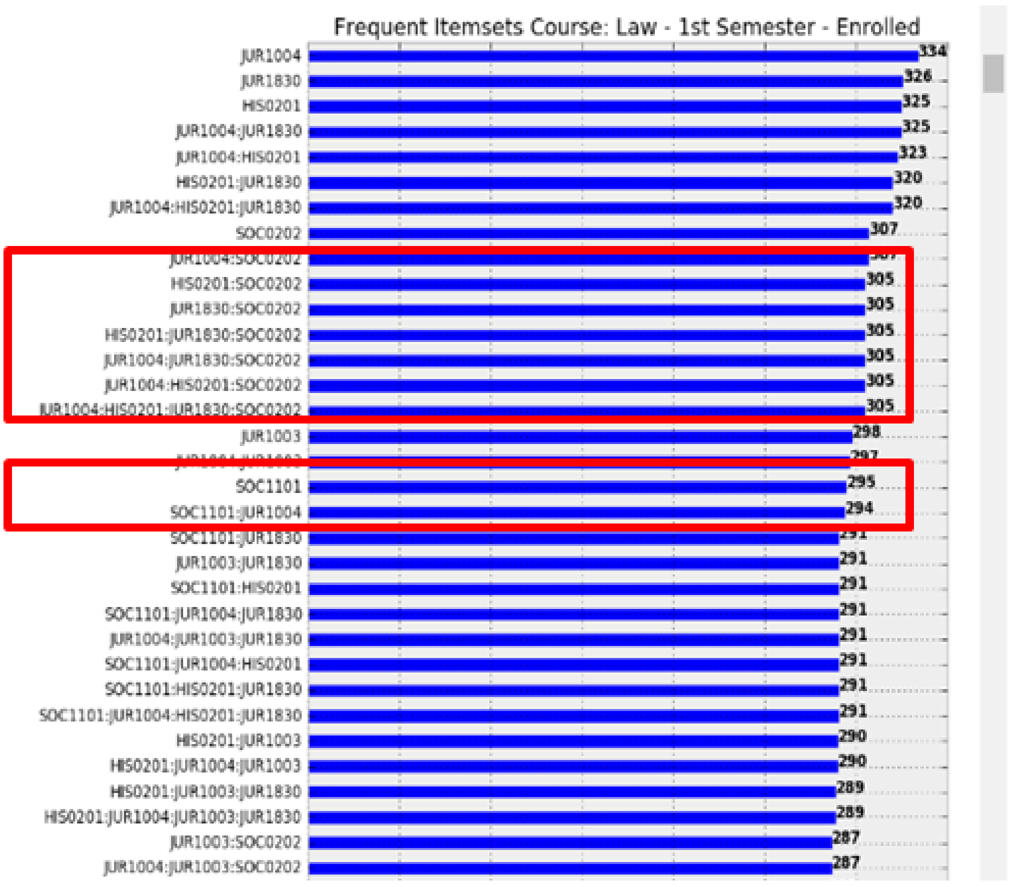


Figure 21. Analysis in detail of results the frequent itemsets.

To tackle this issue, we resorted to hierarchical agglomerative clustering (RAJARAMAN & ULLMAN, 2011), according to Ward's linkage method (WARD, 1963; FERREIRA & HITCHCOCK, 2009) for each status and each semester. In order to determine the number of clusters to be created, taking into account the difference of frequencies, and without losing important information, the first step was the creation of a dendrogram (FERREIRA & HITCHCOCK, 2009) for the dataset.

Figure 22 shows the dendrogram for Law Degree, Litigation emphasis, first semester. In this case, we experimented with different cluster sizes (two, three and four) and the decision was to create three clusters because it is more representative. As argued before, we proceeded to eliminate a set whenever it was a subset of another set present in the cluster. Figure 23 shows the final result of the frequent itemsets analysis, after this elimination process.

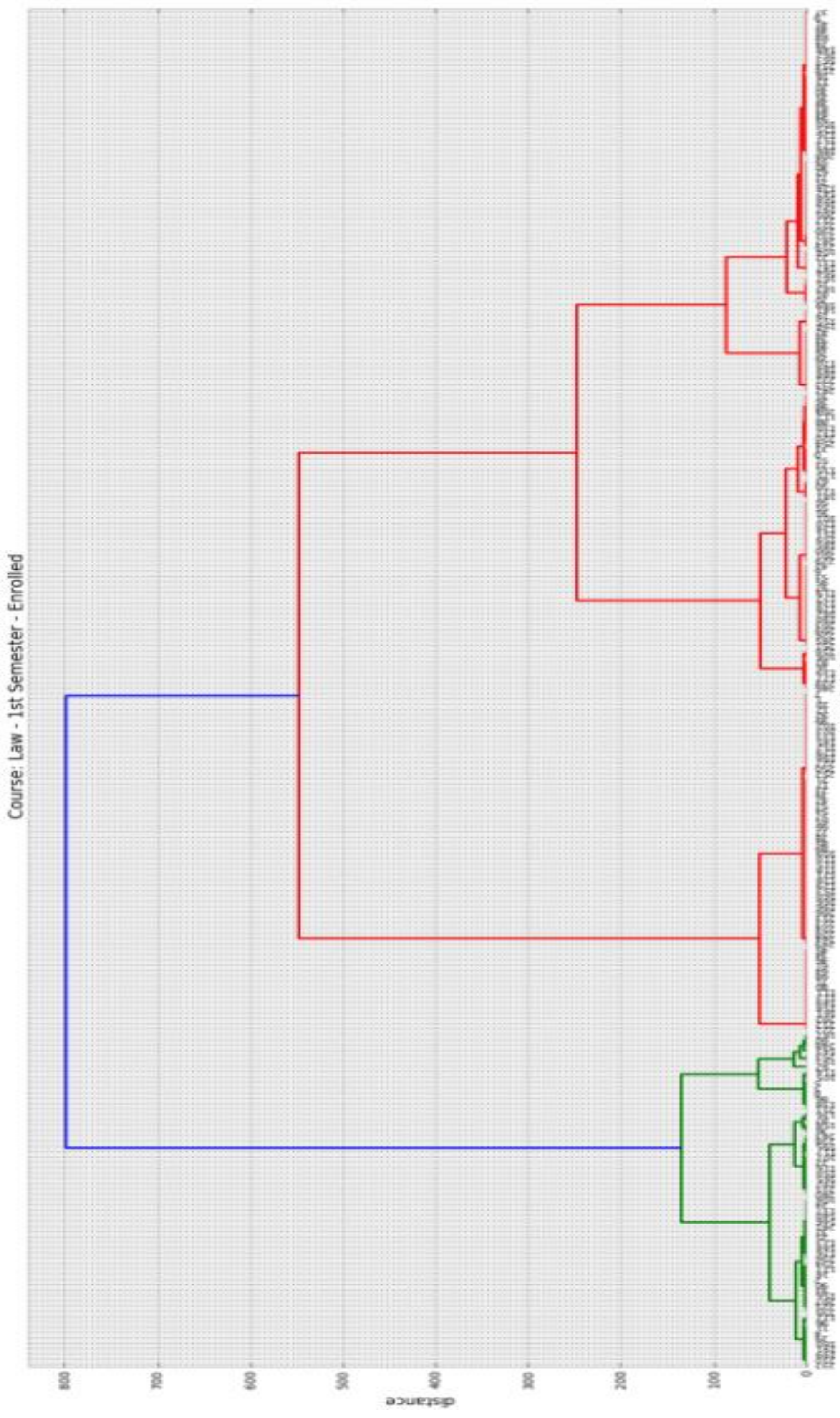


Figure 22. Dendrogram of the Law degree, litigation emphasis, 1st semester.

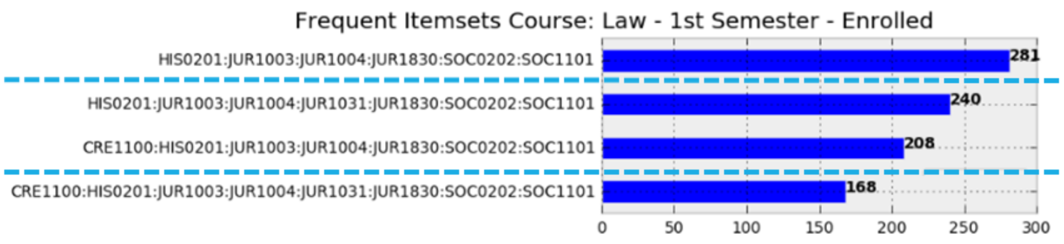


Figure 23. Frequent itemsets, enrolled students, 1st semester, LL course degree.

As presented in Figure 23, the resulting information looks better organized thanks to the removal of unnecessary subsets. It is clear that the set of courses proposed in the recommended curriculum – shown in the last line of Figure 23 – is less frequent than the sets of courses that students actually enrolled in, represented in the other lines of the figure. With this information, the coordinators of the degree should become aware that most of the students do not tend to enroll in courses CRE1100 and JUR1031 in the suggested semester.

In order to support the decisions taken from analyzing the frequent itemsets of the students enrolled, it is also necessary to reapply the algorithms to the same set of data, but this time only leaving those courses that the students either succeed to pass or decide to drop. Figure 24 shows the results for these two sets of data.

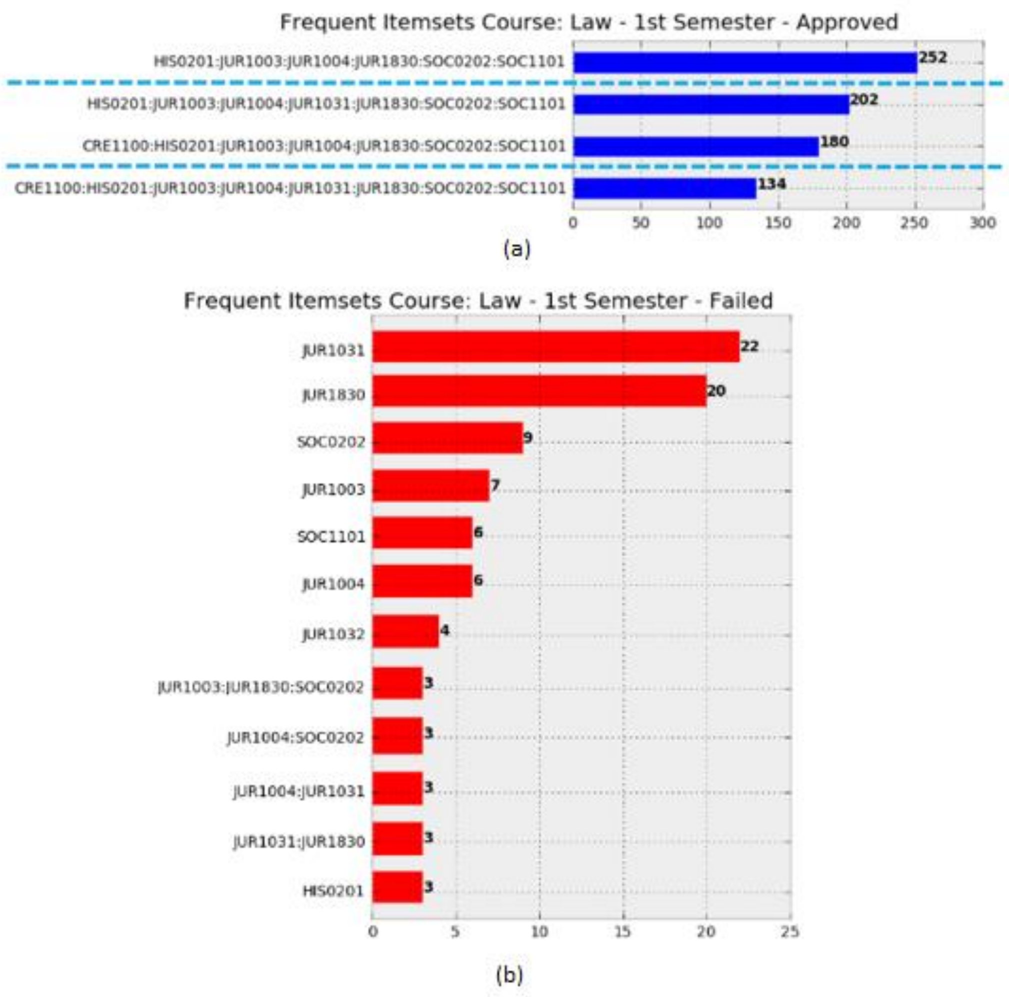


Figure 24. Frequent itemsets, 1st semester, LL course degree (a) Approved students. (b) Failing students.

In conclusion, when applying the frequent itemsets technique for the dataset containing only the students that passed, we obtained the same sets of courses as before, but with a lower frequency. For the dataset containing only the students that stopped taking the course, we uncovered which were the two most problematic disciplines of the semester, namely JUR1031 and JUR1830.

5
Conclusions and Future Work

In this work, we applied a process mining approach to analyze course degree curricula. We surveyed a number of currently used techniques, and, next, endeavored to test them over undergraduate student files of PUC-Rio, in an attempt to assess their efficacy to answer our questions. The dataset used in our experiments comprised over 60,000 records of nearly 1,700 students.

As we finish our study, we feel confident to conclude that:

- The analysis of academic domains as processes is a promising approach.
- Software tools, such as ProM, did not prove appropriate for answering our questions, for the reasons explained at the end of section 3.1.
- In contrast, techniques for mining frequent itemsets were fundamental to compare the recommended versus the actually followed order of courses, as well as to find in what courses the student enrollment was more frequently delayed or advanced.
- On the basis of our proposed sort of analysis, the academic coordinators should be in a better position to identify where the problems are, helping them to formulate more effective academic policies.

Part of the results obtained in the dissertation were published in the 17th IEEE International Conference on Advanced Learning Technologies (GOTTIN et al., 2017)

As a preliminary phase of future work, we plan to apply the methodology described in Chapter 4 to other universities, in order to start adapting it to the needs of the academic administration domain in general. We also plan to validate the results with course coordinators.

Our long-term goal is to incorporate a consistent combination of the more successful techniques in a tool that may help academic coordinators compose recommended curricula, in a way that effectively improves the performance of the students, meeting as much as possible their demands and minimizing both dropout and failure rates.

6
Bibliography

AGRAWAL, R.; GUNOPULOS, D.; LEYMANN, F. **Mining process models from workflow logs**. Advances in Database Technology — EDBT'98. Anais: Lecture Notes in Computer Science. In: International Conference on Extending Database Technology. Springer, Berlin, Heidelberg, 23 Mar. 1998. Available at: <https://link.springer.com/chapter/10.1007/BFb0101003>

AGRAWAL, R.; IMIELIŃSKI, T.; SWAMI, A. **Mining Association Rules Between Sets of Items in Large Databases**. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. Anais: SIGMOD'93. New York, NY, USA: ACM, 1993. Available at: <http://doi.acm.org/10.1145/170035.170072>.

AHMED, A. B. E. D.; ELARABY, I. S. **Data Mining: A prediction for Student's Performance Using Classification Method**. World Journal of Computer Application and Technology, v. 2, n. 2, p. 43–47, Feb. 2014.

BAEPLER, P.; MURDOCH, C. **Academic Analytics and Data Mining in Higher Education**. International Journal for the Scholarship of Teaching and Learning, v. 4, n. 2, 1 Jul. 2010.

BARADWAJ, B. K.; PAL, S. **Mining Educational Data to Analyze Students' Performance**. arXiv:1201.3417 [cs], 16 Jan. 2012.

BAYARDO, R. J., Jr. **Efficiently Mining Long Patterns from Databases**. Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data. Anais: SIGMOD '98. New York, NY, USA: ACM, 1998. Available at: <http://doi.acm.org/10.1145/276304.276313>.

BOGARD, M. et al. **A comparison of empirical models for predicting student retention.** White Paper. Office of Institutional Research, Western Kentucky University, 2011.

CALVERT, C. **Developing a model and applications for probabilities of student success: a case study of predictive analytics**. Open Learning: The Journal of Open, Distance and e-Learning, 2014.

CHENG, J.; KE, Y.; NG, W. **A Survey on Algorithms for Mining Frequent Itemsets over Data Streams**. Knowledge and Information Systems, v. 16, n. 1, p. 1–27, Jul. 2008.

CHEUNG, Y.-L.; FU, A. W.-C. **Mining frequent itemsets without support threshold: with and without item constraints**. IEEE Transactions on Knowledge and Data Engineering, v. 16, n. 9, p. 1052–1069, Sep. 2004.

COOK, J. E.; HE, C.; MA, C. **Measuring behavioral correspondence to a timed concurrent model**. Proceedings IEEE International Conference on Software Maintenance. ICSM 2001. Anais. In: Proceedings IEEE International Conference on Software Maintenance. ICSM 2001.

COOK, J. E.; WOLF, A. L. **Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model**. ACM Trans. Softw. Eng. Methodol., v. 8, n. 2, p. 147–176, Apr. 1999.

DESHPANDE, A. et al. **Student Performance Analysis, Visualization and Prediction Using Data Mining Techniques**. Imperial Journal of Interdisciplinary Research, v. 2, n. 5, 1 Apr. 2016.

EINDHOVEN UNIVERSITY OF TECHNOLOGY. **Process Mining**. Available at: <http://www.processmining.org/prom/start>. Access in: 15 Jun. 2016.

FERREIRA, L.; HITCHCOCK, D. B. **A Comparison of Hierarchical Methods for Clustering Functional Data.** Communications in Statistics - Simulation and Computation, v. 38, n. 9, p. 1925–1949, 1 Oct. 2009.

GOLDSTEIN, P.; KATZ, R. **Academic Analytics: The Uses of Management Information and Technology in Higher Education.** EDUCASE Publications, , 12 Dec. 2005. Available at: <https://library.educause.edu/resources/2005/12/academic-analytics-the-uses-of-management-information-and-technology-in-higher-education>

GOTTIN, V. et al. **An Analysis of Degree Curricula through Mining Student Records**. 2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT). Anais. In: 2017 IEEE 17Th International Conference on Advanced Learning Technologies (ICALT). Jul. 2017

KIMBALL, R.; ROSS, M. **The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling**. Edición: 3. Auflage. ed. Indianapolis, Ind: Wiley John + Sons, 2013.

KIMBALL, R.; ROSS, M. **The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling**. Edition: 3. Auflage. ed. Indianapolis, Ind: Wiley John + Sons, 2013.

LAURÍA, E. J. M. et al. **Mining Academic Data to Improve College Student Retention: An Open Source Perspective**. Proceedings of the 2Nd International Conference on Learning Analytics and Knowledge. Anais: LAK '12. New York, NY, USA: ACM, 2012. Available at: <http://doi.acm.org/10.1145/2330601.2330637>.

LEEMANS, S. J. J.; FAHLAND, D.; VAN DER AALST, W. M. P. **Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour**. Business Process Management Workshops. Anais: Lecture Notes in Business Information Processing. In: International Comferemce on Business Process Management. Springer, Cham, 26 Aug. 2013. Available at: <https://link.springer.com/chapter/10.1007/978-3-319-06257-0_6>

LEEMANS, S. J. J.; FAHLAND, D.; VAN DER AALST, W. M. P. **Process and deviation exploration with Inductive visual Miner.** CEUR-WS.org, 2014. Available at: <http://repository.tue.nl/783935>.

MEDEIROS, A. K. A. D. et al. **Process Mining for Ubiquitous Mobile Systems: An Overview and a Concrete Algorithm**. Ubiquitous Mobile Information and Collaboration Systems. **Anais**: Lecture Notes in Computer Science. In: International Workshop on Ubiquitous Mobile Information and Collaboration Systems. Springer, Berlin, Heidelberg, 7 Jun. 2004 Available at: <https://link.springer.com/chapter/10.1007/978-3-540-30188-2_12>

MUNOZ-GAMA, J. **Conformance Checking and Diagnosis in Process Mining**. [s.l.] Springer International Publishing, 2014.

NORRIS, D. M.; LEONARD, J. **What Every Campus Leader Needs to Know About Analytics.** Strategic Initiatives, Inc, Jan. 2008. Available at: <https://www.blackboard.com/cmspages/getfile.aspx?guid=cd33d46f-5230-4d1b-b81c-052312af9cf2>

OBLINDER, D.; CAMPBELL, J. **Academic Analytics.** EDUCASE Publications, , 15 Oct. 2007. Available at: <https://library.educause.edu/resources/2007/10/academic-analytics>

PAPAMITSIOU, Z.; ECONOMIDES, A. A. **Learning Analytics and Educational Data Mining in Practice: A Systematic Literature Review of Empirical Evidence.** Journal of Educational Technology & Society, v. 17, n. 4, p. 49–64, 2014.

PARACK, S.; ZAHID, Z.; MERCHANT, F. **Application of data mining in educational databases for predicting academic trends and patterns**. 2012 IEEE International Conference on Technology Enhanced Education (ICTEE). **Anais**. In: 2012 IEEE INTERNATIONAL CONFERENCE ON TECHNOLOGY ENHANCED EDUCATION (ICTEE). Jan. 2012

PHILLIPS, R. et al. **Exploring Learning Analytics as Indicators of Study Behaviour**. In: EDMEDIA: WORLD CONFERENCE ON EDUCATIONAL MEDIA AND TECHNOLOGY. Association for the Advancement of Computing in Education (AACE), 26 Jun. 2012. Available at: <https://www.learntechlib.org/p/41174/>.

RAJARAMAN, A.; ULLMAN, J. D. **Mining of Massive Datasets**. New York, NY, USA: Cambridge University Press, 2011.

RAMASWAMI, M.; RATHINASABAPATHY, R. **Student Performance Prediction Modelling: A Bayesian Network Approach**. 2012.

ROZINAT, A.; VAN DER AALST, W. M. P. **Conformance Checking of Processes Based on Monitoring Real Behavior.** Inf. Syst., v. 33, n. 1, p. 64–95, Mar. 2008.

SAA, A. A. **Educational Data Mining & Students' Performance Prediction**. International Journal of Advanced Computer Science and Applications (IJACSA), v. 7, n. 5, 2016.

SIEMENS, G. **What are Learning Analytics?** Elearnspace, 2010. Disponível em: <http://www.elearnspace.org/blog/2010/08/25/what-are-learning-analytics/>. Access in: 29 May. 2017

TAIR, M. M. A.; EL-HALEES, A. M. **Mining Educational Data to Improve Students ' Performance: A Case**. 2012.

VAN DER AALST, W. M. P. et al. **Workflow Mining: A Survey of Issues and Approaches**. Data Knowl. Eng., v. 47, n. 2, p. 237–267, Nov. 2003.

VAN DER AALST, W. M. P. **Process Mining: Discovery, Conformance and Enhancement of Business Processes**. 1st. ed. [s.l.] Springer Publishing Company, Incorporated, 2011.

VAN DER AALST, W. M. P. **Distributed Process Discovery and Conformance Checking**. Proceedings of the 15th International Conference on Fundamental Approaches to Software Engineering. Anais: FASE'12. Berlin, Heidelberg: Springer-Verlag, 2012 Available at: <http://dx.doi.org/10.1007/978-3-642-28872-2_1>.

VAN DER AALST, W. M. P.; VERBEEK, H. M. W. **Process Discovery and Conformance Checking Using Passages**. Fundam. Inf., v. 131, n. 1, p. 103–138, Jan. 2014.

VAN DER AALST, W. M. P.; WEIJTERS, A. J. M. M.; MARUSTER, L. **Workflow Mining: Which processes can be rediscovered?** Eindhoven University of Technology. Anais.2002

VAN DER AALST, W. M. P.; WEIJTERS, A. J. M. M.; MARUSTER, L. **Workflow mining: discovering process models from event logs**. IEEE Transactions on Knowledge and Data Engineering, v. 16, n. 9, p. 1128–1142, Sep. 2004.

VO, B. et al. **Mining frequent itemsets using the N-list and subsume concepts**. International Journal of Machine Learning and Cybernetics, v. 7, n. 2, p. 253–265, 1 Apr. 2016.

WARD, J. H. **Hierarchical Grouping to Optimize an Objective Function**. Journal of the American Statistical Association, v. 58, n. 301, p. 236–244, 1 Mar. 1963.

WEIJTERS, A. J. M. M.; MEDEIROS, A. K. A. D. **Process Mining with the Heuristics Miner Algorithm**, 2006.

WEIJTERS, A. J. M. M.; RIBEIRO, J. T. S. **Flexible heuristics miner (FHM)**. In Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on. IEEE, 2011. Anais.2010

WEN, L. et al. **Mining process models with non-free-choice constructs**. Data Mining and Knowledge Discovery, v. 15, n. 2, p. 145–180, 1 Oct. 2007.

WEN, L. et al. **Mining Process Models with Prime Invisible Tasks**. Data Knowl. Eng., v. 69, n. 10, p. 999–1021, Oct. 2010.

WIKIPEDIA. **Jensen–Shannon divergence**. Available at: <https://en.wikipedia.org/w/index.php?title=Jensen%E2%80%93Shannon_divergence&oldid=780000951>. Access in: 26 Jul. 2017a.

WIKIPEDIA. **Conformance checking**. Available at: <https://en.wikipedia.org/w/index.php?title=Conformance_checking&oldid=788799268>. Access in: 2 May. 2017b.

WIKIPEDIA. **Euclidean distance**. Available at: <https://en.wikipedia.org/w/index.php?title=Euclidean_distance&oldid=801348305>. Access in: 26 Jul. 2017c.

YUKSELTURK, E.; OZEKES, S.; TUREL, Y. K. **Predicting Dropout Student: An Application of Data Mining Methods in an Online Education Program**. European Journal of Open, Distance and E-Learning, v. 17, n. 1, p. 118–133, 2014.

ZAKI, M. J.; JR, W. M. **Data Mining and Analysis: Fundamental Concepts and Algorithms**. New York, NY, USA: Cambridge University Press, 2014.