Simple Metrics for Curricular Analytics

Xavier Ochoa
Escuela Superior Politécnica del Litoral
Vía Perimetral, Km. 30.5
Guayaquil, Ecuador
xavier@cti.espol.edu.ec

ABSTRACT

The analysis of a program curriculum is traditionally a very subjective task. Perceptions and anecdotes, faculty preferences and content or objectives check-lists are the main sources of information to undergo the revision of the structure of a program. This works proposes a list of simple metrics, that can be easily extracted from readily available academic data that contains the information about the actual interactions of students with the curriculum. These metrics, divided into time- and performance-related, are calculated at program level. The use of these metrics provides objective information in which to base discussions about the current state and efficiency of the curriculum. To exemplify the feasibility and usefulness of the metrics, this work presents some illustrative analysis that make use of the simple curriculum metrics.

CCS Concepts

•**Applied computing → Education;**

Keywords

Learning Analytics, Curriculum Analytics

1. INTRODUCTION

Learning Analytics has been traditionally applied to understand and optimize the learning process at course level. The learning process is analyzed through the captured interactions between students and instructor, content or tools. However, Learning Analytics are not restricted to act at this level. Adapted techniques, applied to different sources of information, could be used to understand and optimize learning at the program level, as exemplified by the works of Pechenizkiy et al. [8] and Gonzalo et al. [7]. Due to the interconnection between learning at the course and the program level, Program Analytics are an indispensable complement to traditional Learning Analytics in order to have an effective end-to-end learning process.

.

There are several sources of information that can be used to analyze a program curriculum. The first main categorization of this information responds to its level of objectivity. Surveys about needs, perceptions and sentiments are a common tool in curricula analysis. These surveys can be directed to students [6], faculty [7], alumni [3] or the labor market [5]. The result of these surveys provide subjective information. On the other hand, curriculum analysis could also employ factual data obtained from the curriculum and its usage. This data can be classified as objective information.

The objective information could be further classified in three main groups:

- Intrinsic: This is the information that is contained in the curriculum itself. For example, Sekiya et al., used the descriptions provided in the syllabi of several Computer Science curricula to compare their compliance to with the Computer Science Curriculum recommendation from ACM [9].
- Extrinsic: This is the information external to the program that influence its content or structure. For example, Sugar et al., [10] found required multimedia production competencies for instructional designers, compiling information from instructional design job advertisements.
- Interaction: This is information that is generated when the students interact with the curriculum. The most common interactive information is the course selection and the grades obtained by students. This information is commonly refereed as student academic records. For example, Bendatu and Yahya [1], inspired by the curriculum mining idea of Pechenizkiy et al. [8], use student records to extract information about the course-taking behavior of students.

From all this sources of data, this work will concentrate in the curriculum interaction data for three main reasons: First, it is automatically captured and readily available to any running program. Second, contrary to the intrinsic information, academic records are easier to analyze and understand. And finally, the relative uniformity in which this information is represented and stored make it an ideal target for analysis techniques that can be used between programs and institutions.

The structure of this paper is as follows: Section two proposes an initial list of useful and easy-to-obtain metrics that can be extracted from curriculum interaction data. Section three validates the ideas behind the metrics through it use

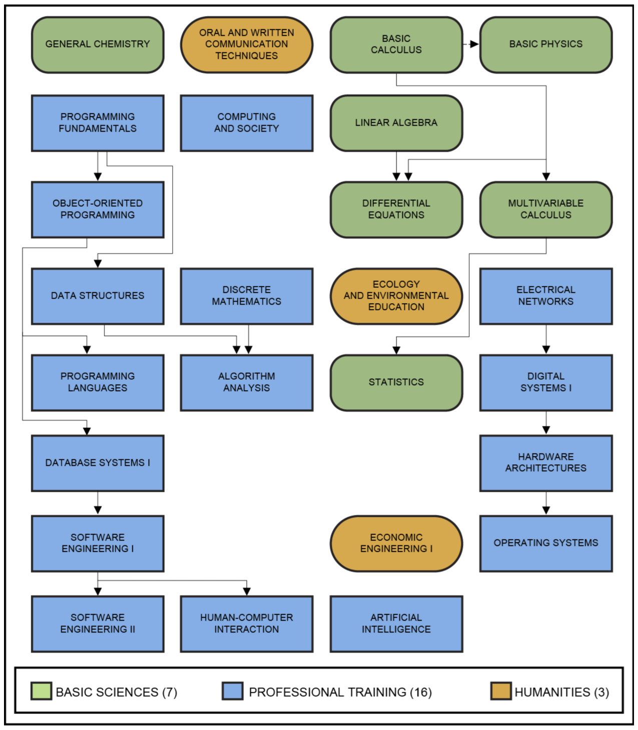

Figure 1: Courses in the CS Curriculum

to perform illustrative curriculum analysis. The paper closes with conclusions and recommendations for further work.

2. CURRICULUM METRICS

Metrics are objective measurements or calculations of the characteristics of an object that simplify their understanding and analysis. While obtaining the value for a given metric is not the same as performing an analysis, metrics are the base of quantitative analytics. This work proposes curriculum interaction metrics than can be used to perform quantitative analysis of the status and efficiency of program curricula. These proposed metrics will be calculated exclusively from curriculum interaction data (academic records).

Academic records can be seen as the capture of the interactions between students and the program curriculum. Table 1 present an example of the usual information present in the records of an academic institution. As a minimum, academic records contain information about two main interactions events: 1) the decision of the student to register in a given course during a given academic period, and 2) the level of success of the student within the chosen courses. Due to these two different interaction aspects, the curriculum interaction metrics will be grouped into two sets described in the subsections below.

To obtain a first insight on the values generated by the tool, they will be applied to a real Computer Science program in a medium-sized university. This program will serve as a case study. The curricula of this program can be seen in Figure 1

2.1 Temporal metrics

In academic programs where students have the flexibility to select courses at different temporal points during their studies, that selection could provide useful information for curriculum analyzers. This work propose three metric associated with the temporal information of the academic record.

Table 1: Example of Academic Records

Student Id	Course Id	Semester	Grade
200002608	ICM00604	2001-1S	6.75
200002608	FIEC04341	2001-2S	8.32
200225076	ICM00604	2002-1S	4.23
200300341	ICF00687	2003-2S	9.01

2.1.1 *Course Temporal Position (CTP)*

This simple metric measure the average academic period (semester or year) in which a course is taken by the students of a program. This information can be used to establish the real position of a course in the program.

To calculate this metric, the raw academic period information needs to be converted into a relative value. For example, in a semester based program, if a student started their studies during the first semester of 2004 and he or she took the relevant course during the second semester of 2006, the relative period will be 6, because the course was taken on the sixth semester relative to the student's first semester. To avoid to inflate the metric, only active periods, that is periods where the student has been actively pursuing the program, should be counted. Once relative period of a course is calculated for all the students that have approved the course N, the average is calculated according to Equation 1, where RP_s^c is the relative period of the analyzed course c for a given student s. Additionally, this metric can be configured to obtain the temporal position when a course was initially taken or when it was finally approved. Depending on the type of analysis, these two different metric versions could be useful.

$$CTP_c = \frac{1}{N}\sum_{s=1}^{N} RP_s^c \qquad (1)$$

When this metric is calculated for all the core courses of the Computer Science case study program (Table 2), it is clear that there are considerable differences between the semester when the course is programmed and the average semester when the students approve that course. The largest difference correspond to Object-Oriented Programming. This course is programmed to be taken during the third semester, however, students, in average, are approving this course during the sixth semester. On the other side of the spectrum, Discrete Mathematics is programmed to be taken during the fourth semester, but students are approving it earlier (third semester). This information could be used to restructure the curriculum.

2.1.2 *Temporal Distance between Courses (TDI)*

This metric establishes how many academic periods, in average, pass between a student taking two different courses. This information can be used to establish the actual sequence in which courses are taken.

While a simple way to calculate this metric will be to subtract the CTP of the second course from the first course, information about the actual time difference for each student is lost due to the average nature of the CTP. To calcu-

Table 2: Values of planned semester vs. CTP for all the core courses in the CS Program

Course	Planned Semester	CTP
OBJECT-ORIENTED PROGRAMMING	3	5.768965517
HARDWARE ARCHITECTURES	6	8.725
OPERATING SYSTEMS	8	10.51557093
PROGRAMING LANGUAGES	5	7.457478006
DIGITAL SYSTEMS I	5	7.303882195
ELECTRICAL NETWORKS	4	6.238329238
HUMAN-COMPUTER INTERACTION	8	10.19935691
SOFTWARE ENGINEERING II	8	9.97318612
SOFTWARE ENGINEERING I	7	8.920821114
ALGORITHM ANALYSIS	5	6.903743316
DIFERENCIAL EQUATIONS	3	4.868390129
DATABASE SYSTEMS I	6	7.845737483
ARTIFICIAL INTELLIGENCE	8	9.504983389
ORAL AND WRITTEN COMMUNICATION TECHNIQUES	1	2.498585573
MULTIVARIATE CALCULUS	2	3.471134021
GENERAL CHEMISTRY	1	2.294483294
PROGRAMMING FUNDAMENTALS	2	3.252823632
DATA STRUCTURES	4	4.946681175
STATISTICS	5	5.934782609
BASIC CALCULUS	1	1.846450617
BASIC PHYSICS	1	1.804273504
LINEAR ALGEBRA	2	2.791219512
COMPUTING AND SOCIETY	2	2.356042174
ECOLOGY AND EVIRONMETAL EDUCATION	4	4.025195482
ECONOMIC ENGINEERING I	7	6.876140808
DISCRETE MATHEMATICS	4	3.333333333

late TDI (Equation 2), the relative periods of the relevant courses ($c1$ and $c2$) are subtracted for each student. Then, the average is taken.

$$TDI_{c1,c2} = \frac{1}{N}\sum_{s=1}^{N}(RP_s^{c2} - RP_s^{c1}) \tag{2}$$

When applied to the CS case study program, it is now apparent that courses that should be taken in sequence, are actually taken with 2 or more semesters apart. For example, reviewing course position in Figure 1 and the values in Table 2, it is clear that subjects like Differential Equations should be taken immediately after Linear Algebra. In reality, they are taken, in average, two semesters apart. This information could be useful to better guide students in course selection.

2.1.3 Course Duration (CDU)

This metric measures the average number of academic periods that students need to pass a given course. This metric provides information about the effect that a course has in the length of the program.

CDU is obtained by subtracting the relative period of the first time each student took the course ($RPfirst_s^c$) from the relative period when the student finally passed ($RPpass_s^c$) it and then averaging these values between all the students (Equation 3). A variation of this metric only considers the periods where the course was taken. In this case, the metric is identical to the average number of times that students need to repeat the course before passing.

$$CDU_c = \frac{1}{N}\sum_{s=1}^{N}(RPpass_s^c - RPfirst_s^c) \tag{3}$$

When CDU is applied to the CS case study program, the values (Table 3) present some interesting results. Some courses perceived as difficult, for example Basic Calculus, takes 2 semesters to be approved. However, other courses, also considered difficult, for example Software Engineering, are always passed during the first attempt.

2.2 Difficulty metrics

Each time a student undertakes a course, performance information is captured and stored. The way in which this information is represented varies, but usually involved a grading scale. This scales could be categorical (letters, passing/not-passing etc.) or numerical (20 out of 100, 4 out of 5, etc.). The information stored in the student grades can be processed to produce useful information about the difficulty of different courses in the program. This work summarizes some simple difficulty metrics proposed by previous works and propose two new profile-base metrics.

2.2.1 Simple Difficulty Metrics

The most basic metrics of the difficulty of a course are the passing rate (PR), the number of students that have approved the course divided by the number of students that have taking the course, and the average grade (AG), the sum of the grades of all students (converted to a numerical value) divided by the number of students. These metrics, however, are not comparable between courses because they

Table 3: CDU values for all the core courses of the CS Program ordered from largest to smallest

Course	CDU
BASIC CALCULUS	2.213775179
PROGRAMMING FUNDAMENTALS	1.873074101
STATISTICS	1.804930332
BASIC PHYSICS	1.743679775
DIFERENCIAL EQUATIONS	1.730544747
ELECTRICAL NETWORKS	1.586794462
LINEAR ALGEBRA	1.534738486
DATA STRUCTURES	1.439759036
GENERAL CHEMISTRY	1.438584316
MULTIVARIATE CALCULUS	1.426287744
PROGRAMING LANGUAGES	1.415881561
OBJECT-ORIENTED PROGRAMMING	1.285101822
DISCRETE MATHEMATICS	1.268479184
DIGITAL SYSTEMS I	1.263420724
DATABASE SYSTEMS I	1.247706422
ARTIFICIAL INTELLIGENCE	1.236245955
ALGORITHM ANALYSIS	1.230769231
COMPUTING AND SOCIETY	1.207446809
OPERATING SYSTEMS	1.205042017
ECOLOGY AND EVIRONMETAL EDUCATION	1.149152542
ORAL AND WRITTEN COMMUNICATION TECHNIQUES	1.097040606
ECONOMIC ENGINEERING I	1.093867334
HUMAN-COMPUTER INTERACTION	1.05229794
HARDWARE ARCHITECTURES	1.037356322
SOFTWARE ENGINEERING II	1.026479751
SOFTWARE ENGINEERING I	1.017492711

depend on the group of students that taken the course. A course with relatively good students will have a better PR and AG than a course when only regular or bad students.

Calulkins et al. [2] proposed more robust difficulty metrics. Two metrics, Grading Stringency, also called β (Equation 4) and Multiplicative Magnitude, also called α (Equation 5) eliminate the bias introduced by the group of students taking the course by subtracting from the GPA of each student (GPA_s) the grade that he or she obtained in the course (r_{sc}) and averaging those values over all the students (N). However, the calculation of β and α metrics assume a normal distribution of grades that is usually not the real case.

$$\beta_c = \frac{1}{N_c}\sum_{s=1}^{N_c}(GPA_s - r_{sc}) \tag{4}$$

$$\alpha_c = \frac{\sum_{s=1}^{N_c} {GPA_s}^2}{\sum_{s=1}^{N_c}(r_{sc} * GPA_s)} \tag{5}$$

These metrics were applied to the CS case study and were reported in a previous work [7].

2.2.2 *Profile-Based Metrics*

Simple Difficulty metrics (PR, AG, β and α) reduce the difficulty of a course to a single number. However, as demonstrated by Mendez et al. [7], course difficulty is different for different types of students. To account for this difference, this work proposes a set of profile-based difficulty metrics.

The basic idea behind profile-based metrics is to divide the population of students in different groups according to their performance (usually their GPA). For example, in a program with grades between 0 and 10 and a passing grade of 6, students could be grouped in with following schema: students with [GPA higher than 8.5], [GPA of 7.5 to 8.5], [GPA of 6.5 to 7.5], [GPA of 5.5 to 6.5] and [GPA lower than 5.5]. Then the relevant metric for a course is calculated separately for each group using only information from the performance of its members.

The use of profile for the difficulty metrics reduce the bias of the PR and AG as it is calculated only for similar students in different courses. Also, the profile-based metrics preserve the basic grade distribution shape for β and α.

The proposed profile-based difficulty metrics are:

- Course Approval Profile (CAP): This is the profile-based version of the Passing Rate (PR) metric. For each student group, the number of students on that profile that have passed the course in a give period is divided by the number of students in the group that have taken the course in the same period.

- Course Performance Profile (CPP): This is the profile-based version of the Average Grade (AG) metric. For each group of students that have taken the course, the AG is calculated.

- Course Difficulty Profile (CDP): This is the profile-based versions of the metrics proposed by Calulkins et al. It can be Additive (CDP-β) or Multiplicative (CDP-α), depending on the difficulty metric used for each group.

The result of the profile-based difficulty metrics is a vector. This representation enables the use of more sophisticated data mining techniques to compare and group courses according to their difficulty.

All the difficulty metrics could also be calculated for each Course-Instructor pair to provide a better difficulty estimation given that the characteristics and grade stringency of each instructor could bias the metric result if averaged over all instructors.

When applied to the CS case study, the profiled metrics are able to highlight different patterns among courses. For example, in Figure 2, courses perceived as easy, such as Oral and Written Communication have a very similar Profile Approval Rate for all but lowest performing students. On the other hand, difficult courses, such as Differential Equations and Programming Fundamentals, have a very steep decrease in Approval Rate for different type of students. Another example can be sen in Figure 3, where the Profiled Difficulty is represented. For courses perceived as easy, such as Economic Engineering, improve the GPA of all but the lowest performing students. Difficult courses, however, negatively affect the GPA of all students in a degree related to their actual GPA, as is the case for Programming Fundamentals.

3. CURRICULUM ANALYSIS

The main purpose of calculating a set of well understood metrics over the different courses of a program curriculum is able to easily find answers through more complex analysis based on a combination of the metrics' results. This section provides five illustrative examples of these analysis using only the temporal and difficulty metrics presented before.

3.1 Course Concurrency

One of the main tasks in curriculum analysis is to determine the workload that a student will receive over a given academic period. It is a usual practice that instructors from concurrent courses, that is, courses that are taken together in a period, interchange information about their course load (homework, projects, etc.) to avoid to overload the students over specific times during the period, for example near the exams). However, it is not always easy to determine which courses are actually concurrent, specially if the program if flexible.

This analysis can be performed mainly in two ways. Without previously calculated metrics, the recommended technique is to use a frequent itemset mining technique, such as FP-Growth [4]. This technique discover courses commonly taken together more than a given percentage of times (support). However, it is not easy for instructors to determine the right value of the support and the crisp sets that this algorithm return hide information about less frequent but also occurring course concurrences.

In the second method, the determination of concurrency between courses can be easily obtained from the Course Temporal Position (CTP) metric. For example, in a semester-based program, all courses with at CTP between 1 and 1.5 could be considered to be part of the first semester, while all the courses with a CTP between 1.5 and 2.5 could be considered to be in the second semester. Moreover, overlapping sets could be used to assure that less frequent, but also relevant concurrences are taken into account in the period workload discussions.

3.2 Neglected Courses

It is common to find curricula with small sequences of related courses. When those sequences are designed, it is expected that students follow the courses one after another in consecutive periods. This is specially important for difficult courses such as Calculus or Physics where concepts learned in a previous course are necessary to master the concepts of the next one. However, students, specially in flexible programs, could neglect taking some courses due to different factors (difficulty, personal preferences, reduced available time, etc.) If too much time pass between courses, some of the previously learned concepts could be forgotten by the time the next course requires them, generating lower than expected performance.

To find if there are courses that are consistently neglected by students, the Temporal Distance between Courses (TDI) can be used. TDI is applied to each pair of consecutive courses in the analyzed sequence. If a pair of expected consecutive courses have a TDI value higher than a threshold (for example 1.5) the second course could be considered neglected and actions should be taken to encourage the students to take them as originally planned (for example, adding the second course as a prerequisite to a course with TDI between 2 and 2.5 from the first course).

3.3 Bottlenecks Identification

Due to economic constraints, the time that a student takes in completing the program has been of great interest for academic institutions. However, it is not always clear which courses are the bottlenecks that reduce the overall throughput of the program.

One way to identify the offending courses is to convert the curriculum into a graph. Each course will be a node in this graph. A edge will connect each pair of courses. The weight of each edge will be equal to the TDI between the courses it connects. All the edges with weights lower than 1 and higher than 2 are removed to leave only courses taken in sequence. Then the course with lowest CTP is selected as the initial node and the critical path is found in the graph. The critical path determines the longest sequential path from the initial course. For each of the nodes in the critical path, the course duration (CDU) is calculated. Those courses in the critical path with the higher CDU could be flagged as bottlenecks because they are likely to increase the number of periods that a student has to stay in the program.

3.4 Section Planning

Physical or regulatory limitations often determine the maximum numbers of students in a given class. When there are more students than places in a class, it is common practice to create additional sections of the course taught either by the same or a different instructor. Planning the number of sections needed for the next period, before the end of the current period is sometimes a challenge and usually provides unreliable results. This leads to wasting of resources (for example, two half-full sections) or under-served students (for example, students that can not follow the course during the period due to full sections).

The average passing rate it the usual way in which the forecast about the number of students that will be available to take the next courses is calculated. However, given that each period the composition of students varies, the passing rate does not remain constant, leading to inaccurate re-

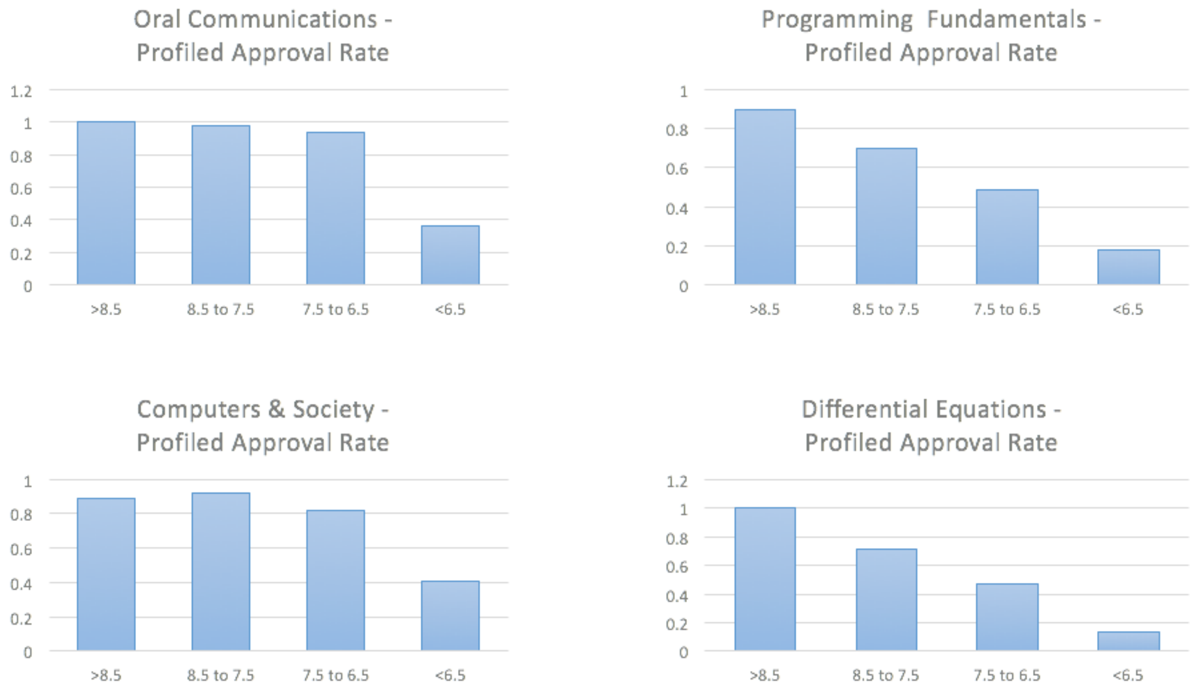

Figure 2: Profiled Approval Rate for different courses in the CS program

sults. The use of the profile-base approval metric (CAP) could provide a better way to forecast the actual number of students that will pass the course because it takes into account the different performance of the students taking the course. These CAP could be refined by using a combination of Course-Instructor to also take into account the grading stringency of the instructor.

3.5 Course Similarity

One of the main curricular decisions that students make is the selection of the course load for each period. The number and difficulty of the courses has a been found to have direct impact on the performance of the students [7]. This decision is so important that it is common for academic institutions to provide course-selection counseling for students that seems to be struggling with their workload. The counseling session, however, only transfer the burden of course selection to instructors or professors that do not necessarily have a current picture of the difficulty and load of all the courses in the program. The decision is taken with a better background knowledge, but still perceptions and beliefs are the main sources of information.

The vector nature of the profile-based difficulty metrics could be exploited to apply straight-forward clustering techniques to group the courses according to their type of difficulty. These groupings could provide an easier way to characterize courses. For example, courses with the same passing rate AG, could be grouped separately according to their Difficulty profile (CDP). Difficult courses, with a linearly decreasing negative β for students with lower GPAs, will be clustered together. The same will happen to easy courses that have a constant β value among the groups. Courses with other distributions (for example, very easy for good performers, but hard for bad performers) will also be clustered with similar courses. Presenting this information for all courses in the program could help instructors to associate the difficulty of known courses to new or unknown courses. This potentially could lead to a better recommendation to the student.

4. CONCLUSIONS AND FURTHER WORK

Differently from data produced at course-level, program-level data tend to be more homogeneous between programs and institutions. This similarity could lead to the development of a sub-field of Learning Analytics with a common set of metrics and methodologies for Program Curriculum analysis that could be called Curricular Analytics. This work is one of the first steps towards the creating this sub-field.

Even simple metrics, when well defined and transferable between programs, have the capacity to improve the way in which curricula are analyzed and improved. The list of metrics presented in this work is by no means comprehensive, but provides a starting point from which more advanced and informative metrics could be created.

The presented illustrative analysis served as an initial validation of the feasibility and usefulness of the metrics. However, a series of evaluation studies with real data from existing programs is needed before these metrics could be safely used by practitioners to draw conclusions from their programs. The operational complexity of these studies is very low given that only the raw data and simple computational tools (for example a spreadsheet) are needed to obtain the metrics. On the other hand, measuring the informational value of the metrics to solve real-world questions requires a more complex quantitative and qualitative analysis.

The relative homogeneity of the data could also lead to the creation of Curricular Analytics tools or plugins that could

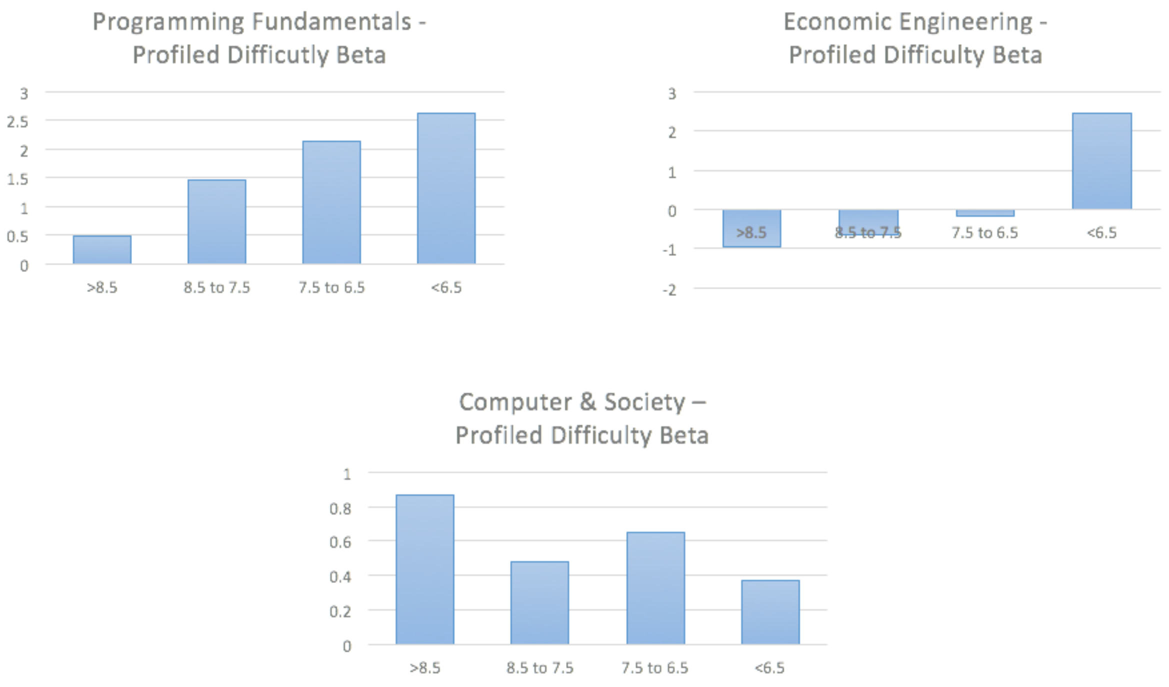

Figure 3: Profiled Difficulty for different courses in the CS program

incorporate all the tested metrics and analysis developed inside this sub-field. The existence of this kind of easy-to-use tools could help in transferring the research results into the practitioners field much faster than what has happened in Learning Analytics in general, where research results are much harder to make inroad in the day-to-day operation of academic institutions.

Finally, this work is a call to other Learning Analytics researchers to start focusing on the different levels of learning and education and the interrelation between those levels. While the focus on course-level analytics could help to improve the learning process in the classroom, only a holistic approach could ensure that these improvements are also reflected in the efficiency and effectiveness of learning programs and that society will receive the benefit of better prepared individuals.

5. REFERENCES

[1] L. Y. Bendatu and B. N. Yahya. Sequence matching analysis for curriculum development. *Jurnal Teknik Industri*, 17(1):47–52, 2015.

[2] J. Caulkins, P. Larkey, and J. Wei. *Adjusting GPA to Reflect Course Difficulty*. H. John Heinz III School of Public Policy and Management, 1996.

[3] R. Davis, S. Misra, and S. Van Auken. A gap analysis approach to marketing curriculum assessment: A study of skills and knowledge. *Journal of Marketing Education*, 24(3):218–224, 2002.

[4] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM, 2000.

[5] J. D. Lang, S. Cruse, F. D. McVey, and J. McMasters. Industry expectations of new engineers: A survey to assist curriculum designers. *Journal of Engineering Education*, 88(1):43–51, 1999.

[6] H. Lempp, C. Seale, et al. The hidden curriculum in undergraduate medical education: qualitative study of medical students' perceptions of teaching. *BMJ*, 329(7469):770–773, 2004.

[7] G. Mendez, X. Ochoa, K. Chiluiza, and B. de Wever. Curricular design analysis: A data-driven perspective. *Journal of Learning Analytics*, 1(3):84–119, 2014.

[8] M. Pechenizkiy, N. Trcka, P. De Bra, and P. Toledo. Currim: curriculum mining. In *Educational Data Mining 2012*, 2012.

[9] M. Sahami, S. Roach, E. Cuadros-Vargas, and D. Reed. Computer science curriculum 2013: reviewing the strawman report from the acm/ieee-cs task force. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 3–4. ACM, 2012.

[10] W. Sugar, B. Hoard, A. Brown, and L. Daniels. Identifying multimedia production competencies and skills of instructional design and technology professionals: An analysis of recent job postings. *Journal of Educational Technology Systems*, 40(3):227–249, 2012.