The *Problem Solving Genome*: Analyzing Sequential Patterns of Student Work with Parameterized Exercises

Julio Guerra[†], Shaghayegh Sahebi[⋆], Peter Brusilovsky[†], Yu-Ru Lin[†]

[†]School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15260, USA
{jdg60, peterb, yurulin}@pitt.edu

[⋆]Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260, USA
shs106@pitt.edu

ABSTRACT

Parameterized exercises are an important tool for online assessment and learning. The ability to generate multiple versions of the same exercise with different parameters helps to support learning-by-doing and decreases cheating during assessment. At the same time, our experience using parameterized exercises for Java programming reveals suboptimal use of this technology as demonstrated by repeated successful and failed attempts to solve the same problem. In this paper we present the results of our work on modeling and examining patterns of student behavior with parameterized exercises using the Problem Solving Genome, a compact encapsulation of individual behavior patterns. We started with micro-patterns (genes) that describe small chunks of repetitive behavior and constructed individual genomes as frequency profiles that show the dominance of each gene in individual behavior. The exploration of student genomes revealed the individual genome is considerably stable, distinguishing students from their peers. Using the genome, we were able to analyze student behavior on the group level and identify genes associated with good and poor learning performance.

Categories and Subject Descriptors

Information systems [**Information Systems Applications**]: Data mining

Keywords

sequential pattern mining, parameterized exercises

1. INTRODUCTION

Parameterized exercises have recently emerged as an important tool for online assessment and learning. A parameterized exercise is essentially an exercise template that is instantiated at runtime with randomly generated parameters. As a result, a single template is able to produce a large number of similar, but distinct questions. While parameterized questions are considerably harder to implement than traditional "static" questions, the benefits offered by this technology make this additional investment worthwhile. During assessment, a reasonably small number of question templates can be used to produce online individualized assessments for large classes minimizing cheating problems [12]. In a self-assessment context, the same question can be used again and again with different parameters, allowing every student to achieve understanding and mastery. The aforementioned properties of parameterized exercises made them very attractive for the large-scale online learning context. At the same time, parameterized exercises as a learning technology have their own problems. Our experience with personalized exercises for SQL [17] and Java [7] in the self-assessment context demonstrated that the important ability to try the same question again and again is not always beneficial, especially for students who are not good at managing their learning. The analysis of a large number of student logs revealed some considerable number of unproductive repetitions. We observed many cases where students kept solving the same exercise correctly again and again with different parameters, well past the point when it could offer any educational benefit. While it might increase self-confidence, students' time and effort might be spent better by advancing to more challenging questions. We also observed cases where students persisted in failing to solve the same, too difficult exercise, instead of focusing on filling the apparent knowledge gap or switching to simpler exercises.

The work presented in this paper was motivated by our belief that the educational value of parameterized exercises could be increased by a personalized guidance mechanism that can predict non-productive behavior and intercept it by recommending a more efficient learning path. The main challenge with predicting unproductive behavior is to examine the stability of behavior patterns in the problem solving process. If the patterns, such as specific unproductive sequences, appear at random, there is a slim chance to predict and prevent them. If, on the contrary, specific patterns are associated with certain features of the student (such as knowledge and individual traits), exercise complexity, or the learning process stage, there is a good chance to learn the association rules and use it for prediction. In this paper we performed an extended study of problem solving patterns in the context of parameterized exercises. We explored the connection between these patterns and the components of the learning process mentioned above. Our study produced a rather unusual result. While it was more plausible to expect that the patterns are related to the current level of student knowledge, our analyses revealed that the patterns are related to student problem solving tendency. More exactly, we discovered that every student has a specific combination of micro-patterns, a kind of problem solving genome. We observed that this genome is relatively stable, distinguishing every student from his or her peers; it changes very little with the growth of the student knowledge over the course. We also discovered that genomes are not randomly distributed, and instead, students with similar genomes form cohorts that perform relatively similarly in the problem solving process. We believe that our discovery of the problem solving genome is a very important step toward our goal of predicting and preventing unproductive behavior. Indeed, the stability of patterns on the personal level makes the task of pattern prediction feasible while the presence of cohorts opens the way to detect the student problem-solving genome early in the learning process. In this paper we present our approach of detecting student problem-solving genome and report our exploration of the genome on the level of individual students and cohorts.

The rest of the paper is structured as follows. The next section briefly reviews several areas of related work. Section 3 describes the dataset used in the study. Section 4 presents the method for building the Problem Solving Genome. In Section 5 we explore the Genome's stability and its relation with performance groups and the complexity of the exercises. Section 6 summarizes the contribution and discusses future work.

2. RELATED WORK

2.1 Parameterized Questions and Exercises

Recent studies in educational technology have demonstrated promising results by leveraging computer and Web abilities to deliver parameterized exercises worldwide, which has become one of the focusing topics in Web-enhanced education. One of the most influential systems, CAPA [9], was evaluated in a number of careful studies, providing clear evidence that individualized exercises can significantly reduce cheating while improving student understanding and exam performance. The CAPA technology has been later integrated into the popular LON-CAPA platform [12] and its functionality defined the assessment architecture of the MOOC platform eDX [1]. Due to the complexity of parameterized assessment, the majority of work on parameterized questions and exercises was done in physics and other math-related domains where a correct answer to a parameterized question can be calculated by a formula. There are, however, examples of using this technology in other domains. In particular, our team focused on parameterized exercises for teaching programming. We developed and explored the QuizPACK platform for C-programming [3] and the similar QuizJET platform for Java programming [7]. Problem solving repetition behaviors have been studied by psychologists in different ways, providing evidence that repetition behaviors have roots in cognitive, metacognitive and motivational aspects and explaining why some students quit and others persist when facing challenging problems [14]. Schunk [16] shows a positive correlation between persistency in repeating and *self-efficacy* (believe on self-capabilities to solve a problem). The attribution theory [19] describes how students that attribute performance outcomes (successes, failures) to effort tend to work harder than students who attribute them to ability. Grounded in the literature in educational psychology, we conjecture that patterns on problem solving repetition may be explained by individual learners' motivational traits that are part of learners' personality [15]. These theories provide insights into analyzing to what extent these behaviors are stable in students.

2.2 Sequential Pattern Mining in Education

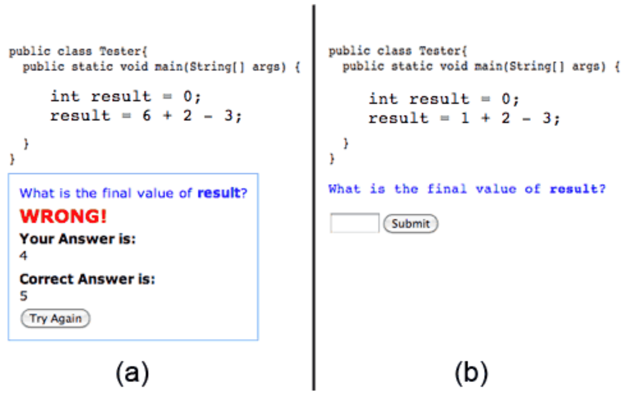
Mining sequential patterns of students actions has recently gained attention in educational data mining field. Using activity data collected from groups of students working with interactive tabletops, Martinez et al [13], mined and clustered frequent patterns to compare distinct behaviors between low and high achievement groups. The differential sequence mining method, introduced by Kinnebrew and Biswas [11] has been successfully used to differentiate behavioral patterns among groups of students (such as low and high performance students). The method uses *SPAM* [1] to find common patterns in the sequences of the whole dataset, and then applies statistical tests to reveal differences in the frequencies of the discovered patterns among different groups. The same authors have applied this technique in data collected from the system Betty's Brain to discovered patterns that can distinguish self-regulated behaviors in successful and non-successful students [2] and to analyze the evolution of reading behaviors in high and low performance students during productive and non-productive phases of work [10]. Herold, Zundel and Stahovich [4] have used the differential sequence mining on sequences of actions on handwritten tasks and proposed a model to predict performance on the course based on pattern features. Our work extends this prior work by utilizing and aggregating the mined sequence patterns to construct student activity profiles. Such profiles enable us to evaluate the statistical differences at the student, exercise, and group levels.

3. SYSTEM AND DATASET

We collected answers of students who worked with QuizJET [7] parameterized Java exercises in the context of an introductory object-oriented programming class at the School of Information Sciences in the University of Pittsburgh. The students accessed the exercises through the Progressor+ interface [6]. The system was provided for self-study and its use was not mandatory. Each QuizJET exercise was generated from a template by substituting a parameter variable with a randomly generated value. Exercises generated using the same template were equal from a semantics point of view. To answer the exercise the student had to mentally execute a fragment of Java code to determine the value of a specific variable or the content printed on a console. When the user answers, the system evaluates the correctness, reports to the student whether the answer was correct or wrong, shows the correct response, and invites the student to "try again". Next time, the exercise is be generated with other values and the correct answer will be different. In this way, the student can try the same exercise many times, leaving a trace of successes and failures. Figure 1 shows a simple parameterized

---

[1] http://www.edx.org/



```
public class Tester{
  public static void main(String[] args) {

     int result = 0;
     result = 6 + 2 - 3;

  }
}
```

What is the final value of **result**?

WRONG!

Your Answer is:
4

Correct Answer is:
5

```
public class Tester{
  public static void main(String[] args) {

     int result = 0;
     result = 1 + 2 - 3;

  }
}
```

What is the final value of **result**?

Figure 1: A parameterized problem in QuizJET. In (a) the student answers wrongly and then hitts "Try Again" button. In (b) the problem is reloaded with different numbers.

Java problem answered incorrectly by the student (a) and then repeated (b). Note the differences in the numbers in the second attempt (b) which correspond to the same problem. Progressor+ provided access to 103 different parameterized exercises organized in 19 topics (Variables, Objects, Arrays, etc.). Exercises are labeled in terms of complexity and there are 41 *easy* exercises, 41 *medium* exercises and 19 *hard* exercises.

The dataset includes three semesters of student data (Spring 2012, Fall 2012 and Spring 2013) in which the use of the system was optional. Overall, 101 students used the system making 6489 incorrect and 14726 correct attempts. Easy exercises were attempted 10620 times, medium complexity exercises were attempted 7876, and hard exercises were attempted 2719 times. Once a student started to work with an exercise she might attempt it just once or try it several times in a sequence. The dataset includes 4212 single attempts (no repetition) and 4758 sequences with more than one attempt. Among these there are 2717 with more than two attempts, 1583 with more than three attempts, and 1016 sequences with more than four attempts.

4. BUILDING THE PROBLEM-SOLVING GENOME

The key idea of our "genome" approach is to build a compact characteristics of student problem-solving behavior on the level of micro-patterns. To build a genome we started by finding proper micro-patterns (genes) and then built a genome of a student as a vector representing the frequencies of different micro-pattern occurrences in the student problem-solving logs. An overview of the genome-building process is shown in Figure 2. To build the genes, we started by labeling students' attempts using time and correctness (Figure 2(a), Section 4.1). We then apply sequential pattern mining to extract sequential micro-patterns Figure 2 (b), Section 4.2). The most frequent micro-patterns were selected as *genes* and used as a basis for the *Problem Solving Genome*, which is a vector of gene frequencies (Figure 2(c), Section 4.3). This section presents the genome-building process in detail while the next sections report our exploration of the Genome.

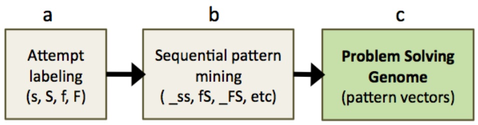


Figure 2: Steps for building the Problem Solving Genome.

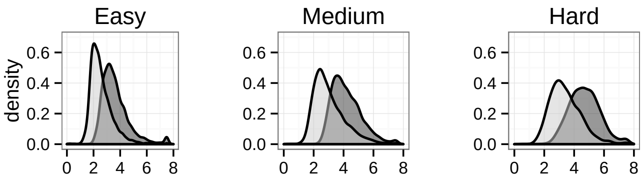


Figure 3: Time distributions (logarithmic) for easy, medium and hard exercises. The right curve is always the first attempt time distribution, showing that first attempts usually take longer times.

4.1 Attempts labeling

We use both time and correctness of each attempt to label it for further use in sequential pattern mining analysis. In this way, each action will convey more information than using correctness only. As shown in Figure 3, distribution of times for first attempts are different from other (non-first) attempts. This is reasonable if we consider that the user needs extra time the first time to read and understand the exercise. Additionally, time distribution is different for different exercises, as in general, complex exercises need longer times. Thus, for labeling the time factor, we used time information of historical records in our system to compute the median times for each exercise for both first and other attempts. Then, we labeled the attempt as *short* or *long* depending on the time being shorter or greater than the median of the distribution for the specific exercise. Combining correctness and time, we finally label the attempts using the letters 's' (lowercase s) for a short success, 'S' (uppercase S) for a long Success, 'f' for a short failure, 'F' for a long Failure.

The labeled attempts are organized in sequences by pairs student-question within a session in the system. Each sequence $s_{u,e}$ represent the sequential attempts of user u in the exercise e within a session. If the user attempted the same exercise in different sessions, there will be more than one sequence $s_{u,e}$. Additionally, we mark starting and ending points on sequences using '_' (underscore). For example, a sequence *_fSs_* means start with a short failure, make a long success and then finish with a short success.

4.2 Sequential pattern mining

To discover frequent patterns, we use the PexSPAM algorithm [5], which extends the fast SPAM algorithm [1] with gap and regular expression constraints. Given a sequence database $D = s_1, s_2, ..., s_n$, the support of a pattern α is the proportion of sequences of D which contains α as a subsequence at least once. If the support of α is bigger than a threshold, then α is considered a frequent pattern. Support measure does not inform for multiple occurrences of the pattern within a sequence. In this work, we set a small minimum support in 1% because even when a pattern oc-

Table 1: Top 20 patterns (genes) ordered by support (the percentage of sequences that contain the pattern). Observe the presence of many inefficient patterns like 'ss' or 'FF' among top 20.

	Pattern	Support		Pattern	Support
1	`ss_`	0.163	11	`_FS`	0.07
2	`ss`	0.107	12	`FS`	0.066
3	`Ss`	0.101	13	`FS_`	0.060
4	`SS_`	0.091	14	`FF`	0.059
5	`_FS_`	0.086	15	`SS`	0.058
6	`_FF`	0.083	16	`_SS`	0.054
7	`Ss_`	0.081	17	`_ss_`	0.053
8	`_fS_`	0.079	18	`_SS_`	0.052
9	`_fF`	0.077	19	`sss`	0.050
10	`sss_`	0.074	20	`_fS`	0.048

curs in overall few sequences, it can still make a difference when looking at the aggregation of pattern occurrences by student. Additionally, since we are interested in looking at patterns of 2 or more sequential attempts, we set the gap to 0 and considered only sequences with more than one attempt. After running the mining algorithm, we discover 102 common patterns occurring at least in 1% of the sequences. These common micro-patterns of student behavior play the role of genes in our approach. The top 20 genes and the corresponding support can be seen in Table 1.

4.3 The problem solving genome: characterizing students with pattern vectors

Using the 102 gene patterns discovered by the sequential pattern mining, we build individual frequency vectors that show how frequently each gene appears in student problem solving behavior. Since this vector captures in a compact form the specifics of student problem solving behavior, we call it student *Problem Solving Genome.* Note that the frequency-based approach allows building individual genome using any subset of gene sequences, for example, all sequences in the term, the first half of sequences of the student activity in the term, a random subset of sequences, etc. Since a pattern might occur more than once in a sequence, and more than one pattern may occur in a sequence, the frequency vectors are not summing to 1. Thus, we normalize the vectors for further analysis.

5. EXPLORING THE GENOME

In this section we analyze the pattern vectors within and between students, across problem complexity levels, and across different student performance groups. We use the same dataset for all further analysis: we select 68 students having pretest/posttest (see section 5.3.1) and a minimum amount of usage of the system of 20 sequences and two sessions. Addtionally, we exclude one outlier student with a very unusual number of repetitions in the first 6 sequences. At the end our dataset consists of 67 students.

5.1 Problem Solving Genome stability

The first step of problem-solving genome exploration is assessing its stability. To what extent the name "genome" that we assigned to the micro-pattern frequency vector is justified? Is it just a random mix of pattern which could be different for different time slots or, like a real genome, it is a stable characteristic of a user that distinguishes him or her from peers? A good approach to check genome stability is to randomly split sequences of user activity patterns into two equal sets and build the genome vector from each of two halves. If the genome is stable, then two random halves of the split genome should be significantly closer to each other than to half-genomes of other users. In contrast, if genome halves are no closer to each other than to half-genome vectors of other users, we can't consider genomes as stable user characteristics. To assess the stability hypothesis we built two half-genomes for each user by randomly splitting her observed sequences in half and compiling gene frequency vectors for each half. We then calculate pairwise distances between all half-genomes.

To compute distances, we use Jensen-Shannon (JS) divergence as it is a symmetric version of Kullback-Leibler divergence and has been widely used for computing distance between frequency distributions. We filter out all students with less than 60 sequences, limiting differences due to extreme difference on amount of activity. Among the 67 students in our dataset, there are 32 students with at least 60 sequences. In this analysis we use a paired samples *t-test* on the difference between the self and other distances. The normality assumption is met. Results are shown in Table 2 first row (a). Students self-distances are significantly smaller ($M = .2370$, $SE = .0169$) than distances to other students ($M = .4815$, $SE = .0141$), $t = -15.224$, $p < .001$, Cohen's $d = 2.693$.

While similarity of random half-genomes is a very strong argument in favor of genome stability, the random split has one weak aspect: since each of the random halves represents student micro patterns over the whole duration of the course, it is still possible that the student genome gradually changes over the course duration from one pattern frequency to another. To assess the temporal stability of the genome we need to use temporal split, i.e., to compare half-genomes built from the temporally first half (early) and second half (late) of student sequences. Results in Table 2 second row (b) confirm the temporal stability hypothesis: while the distance between temporary split half-genomes is larger than between randomly split halves ($M = .3211$, $SE = .0214$) it is still significantly smaller than between-student distances ($M = .4997$, $SE = .0164$), $t = -6815$, $p < .001$, Cohen's $d = 1.205$. This result confirms that frequencies of micro-pattern appearances act as a true problem solving genome "genome": it is considerably stable, characterizing each user as individual over the course progression, while sufficiently distinguishing this user from others.

5.2 Effect of complexity

While we discovered that the knowledge level and course stage doesn't affect the genome, it is still possible that behavior patterns are affected by exercise complexity. To understand how the complexity level of the exercises impacts on the pattern frequencies, we analyze distances between the genome of the exercises (i.e. pattern frequency vector for each exercise). Having the exercises' genome and the predefined classification in *easy*, *medium* and *hard*, we select pairs of exercises within and between complexity levels and compute distances using Jensen-Shannon divergence. We filter out all questions with less than 20 sequences and

Table 2: Statistical tests comparing students with themselves and others.

		self distances		dist. to others				
		M	SE	M	SE	t	*sig.*	Cohen's d
a)	randomly split genome	.2370	.0169	.4815	.0141	-15.224	$< .001$	2.693
b)	early/late genome	.3211	.0214	.4997	.0164	-6.815	$< .001$	1.205
c)	randomly split genome in easy exercises	.3736	.0214	.6065	.0128	-10.352	$< .001$	1.657

Table 3: Mean and standard error of distances within and between easy and hard exercises.

	Mean	SE
within easy	.3311	.0031
within hard	.3478	.0085
between easy-hard	.4145	.0050

perform comparisons between extremes groups, i.e. *easy* and *hard* complexity levels to extreme the differences. Normality and homogeneity of the variance on pair distances are not met on all levels, thus a non-parametric test is applied. Results of the Krustal-Wallis test shows significant differences between distances within and between levels, $\chi^2(2, N = 1596) = 160.359$, $p < 001$. Mean and standard error of distances within easy, within hard, and between easy and hard groups are shown in Table 3. A Mann-Whitney test is performed to test differences among the levels. Distances within easy exercises (mean rank = 626.16) are significantly smaller than distances between easy and hard exercises (mean rank = 909.77), $z = -12.564$, $p < .001$. Similarly, the distances within hard exercises (mean rank = 277.20) are significantly smaller than distances between easy and hard exercises (mean rank = 383.13), $z = -4.733$, $p < .001$. These results show a clear dependency of the pattern behaviors with the complexity level of the questions. This is reasonable given that hard questions, which need more time, are expected to discourage repetitions.

The impact of exercise complexity on the patterns suggest that the genome is as much impacted by the unique exercise difficulty profile than by individual differences of the students. We re-examine the analysis on Section 5.1 now considering randomly split genome built only from activity on easy exercises, to control for differences of students amount of activity on different complexity exercises. We perform this analysis with 39 students having at least 20 sequences in easy questions. Results shown in last row (c) in Table 2 confirm the stability of patterns: students are more similar to themselves (self distance $M = .3736$, $SE = .0214$) than to others (distances $M = .6065$, $SE = .0128$), $t = -10.352$, $p < .001$, Cohen's $d = 1.6569$, even within exercises of the same complexity.

5.3 Patterns of Success within student groups

Since one of the goals of this paper is using behavior analysis to identify and prevent inefficient patterns, it would be valuable to use the genome to identify which patterns make groups of students more or less successful in the learning process. The easiest approach to do it is to split students into performance-related groups and find unique genome aspects in this group. This simple approach, however, might not work since for students with very different genomes, different behavior patterns might be related to success. In this case, to find a connection between patterns and performance, we should group students into groups with similar behavior and contrast most and least successful students within each group. In this section we perform both kinds of the analysis.

Table 4: Number of students in each predefined performance group (PPG).

	Pretest (total=67)	Posttest (total=65)	Learning gain (total=65)
low	24	22	22
medium	16	19	20
high	27	24	23

5.3.1 Patterns for Predefined Performance Groups

Predefined Performance Groups (PPG) are defined based on pre and posttest scores that we collected. The pre and posttest were highly similar among different semesters (small variation on questions) and the scores were further normalized as score / max_score (min score is 0). Additionally, we compute a normalized learning gain score as (normalized post score) - (normalized pre score). For each of the pretest, posttest, and learning gain measures, students are classified in three groups using the percentiles 33.3 and 66.7: *low*, *medium* and *high*. For example, a student with pretest lower or equal than the percentile 33.3 in the pretest score distribution is classified as *low* pretest student. Summarizing, we have 3 PPG (low, medium, high) for each performance measure (pretest, posttest and learning gain). As explained before, the dataset contains 67 students with pretest and 65 students with both pre and posttest. Table 4 shows the number of students in each PPG.

Do students with similar performances have similar patterns for solving parameterized exercises? Is this similarity, between the students of the same predefined performance group, more than the similarity we can find between the students from different groups? For this analysis we contrast the genome built using all the term activity (all problem solving sequences) of the students classified in the performance groups described before. We sample 50% of all possible pairs of students within and between PPGs and compute the distances (Jensen-Shannon divergence) of all within and between group pairs. Then, we compare the average of distances within and between groups to see if students inside each group are more similar to each other than to students in other groups. Normality and homogeneity of variance is not met for all groups, thus we use Krustal-Wallis non-parametric mean rank test and Mann-Whitney test for single comparisons. We constrained the analysis to PPGs *low* and *high* to see extreme differences.

Results are shown in Table 5. Mann-Whitney comparison is reported only where significant differences among groups were found (pretest). For pretest groups, distances within the low group (mean rank = 222.70) are significantly smaller than distances between low and high groups (mean rank =

Table 5: Statistical tests on differences on distances between pairs of students within low, within high, and between low and high PPGs.

	low		high		low-high		Krustal-Wallis test			Mann-Whitney test		
	M	SE	M	SE	M	SE	Mean Ranks (low,high,low-high)	χ^2	*sig.*	Mean ranks	z	sig.
Pretest	.465	.014	.547	.017	.512	.010	294.68, 368.67, 341.51	11.926	.003	222.70, 258.21 low < low-high	-2.537	.011
Posttest	.486	.016	.516	.018	.511	.011	256.41, 271.97, 273.69	1.061	.588	-	-	-
L. Gain	.507	.019	.470	.018	.517	.013	242.32, 216.57, 251.35	5.276	.071	-	-	-

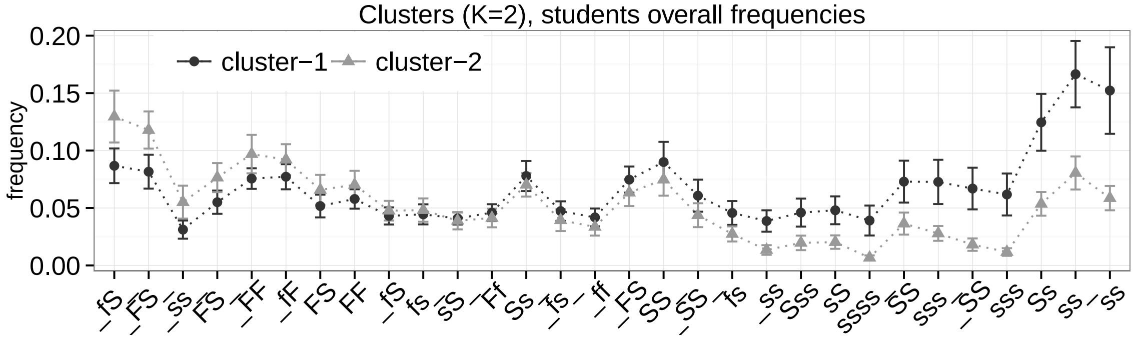


Figure 4: Top 30 patterns and their frequencies in each cluster. Patterns are ordered by the difference on frequencies between cluster two (non-confirmers) and one (confirmers).

258.21), $z = -2.537$, $p = .011$. This suggests that student with no previous experience tend to behave differently than students with stronger background. There is no significant difference between high and low-high distances, though, meaning that the high group behaved more heterogeneously than low group. For posttest and learning gain groups there are no significant differences on distances within and between groups. These results are intriguing, as we will expected to find clear differences among performance groups. Since we could not find those differences, we could hypothesize that specific behavior patterns can't be easily characterized as universally helpful or harmful for student performance, instead, the impact of each micro-pattern on student behavior might depend on the whole profile of micro-patters, i.e., the genome. Thus, to find connections between genome and performance, we need to start from the opposite side: cluster the students based on the genome, characterize the clusters in terms of the distinguishable patterns, and find helpful and harmful patterns within each class. We describe these analyses in the following sub sections.

5.3.2 Clustering students by their genome

We use the genome as a feature vector and cluster students using the spectral clustering technique [18] as it gives a better separation of the students. We choose two clusters (K=2) as we observe that two clusters give the largest eigen-gap, suggesting there are two intrinsic groups in the data. Figure 4 shows the top 30 frequent patterns in both of the clusters. Each point represents the average frequency of seeing a particular pattern in the cluster. Error bars are included to indicate significance. We order the patterns in x-axis by the differences between clusters two and one. As we can see in this figure, some of the patterns, such as *_fS_*, *_FS_*, *_ss*, *Sss*, etc., occur with significant frequency difference in the two clusters and some other patterns, such as *_fS*, *fs_*, *Ff*, etc., do not show significant differences. If we look more closely, the sequences that start with failure are mostly related to the students in cluster two and the sequences that start with success are mostly related to the students in cluster one. Also, we can see that the students in cluster one tend to repeat their successful attempts more and more frequently (e.g. the *ssss_* pattern). In other words, even when they get the right answer to the question, they will insist on confirming knowing the question by repeating it again and again. Unlike students in cluster one, the students in cluster two are much less prone to this "confirmation" behavior. Instead, they are more prone to stop working with an exercise early, frequently right after figuring out the first right answer to the question, even if they have struggled for the correct answer in their previous attempts (e.g. *_fS_*, *_FS_*, and *FS_* patterns). Thus, using the student genome, we can identify two major types of student behaviors in solving parameterized exercises. Based on these observations above, we call the first cluster of the students *the confirmers* and the second cluster *the non-confirmers*.

5.3.3 Performance differences among clusters

Once two clusters of students that are similar in their overall behavior are identified, we can re-examine the connection between student success and behavior patterns on the cluster level. We study pattern by pattern differences between different PPGs within each cluster and describe the patterns that distinguish them. Both of the clusters have students from all PPGs. As a result, we cannot say that the student's genome has a direct impact on the performance of the student. Both *confirmers* and *non-confirmers* can have high or low performance. To look at the clusters deeply and to see if there are any differences in the patterns, within each cluster, that can drive students' performance, we repeat the first analysis within each cluster looking at the learning gain. For each of the clusters, we look at the patterns and the difference between their average frequencies for the students

Figure 5: Top 30 patterns and their frequencies for low and high learning gain PPG by cluster.

with low and high learning gain. The result is shown in Figure 5. The upper diagram shows the students in cluster one (the confirmers) and the lower diagram shows the students in cluster tow (the non-confirmers). The red line with round markers show the pattern frequencies for low learning gain students and the blue line with the triangle marker is representative of high learning gain students.

If we look at the patterns in cluster one (the confirmers), we can see that there are some patterns that show significant difference between the low and high learning gain students. Each of these patterns starts with a failure: *_FS_* and *Ff* have long failures in the beginning of the patterns and *_fF*, *fs_*, and *_ff*, have short failures at the beginning of the patterns. Among these patterns, only *_FS_* is practiced more by the high learning gain students. This indicates that, among the confirmer students, the ones that put a good amount of effort to answer a question correctly after a long failure and stop repeating the same question learn more. The low learning gain group shows more frequent use of the *Ff*, *_fF*, *fs_*, and *_ff* patterns. The common element of all of these patterns is short failure (*f*). If we look at Figure 5 for confirmers, we can see that all of the patterns that include a short failure, are practiced more by the low gain students. This can indicate that the low gain confirmer students do not spend enough time and thought on the questions to which they do not know the answer.

The non-confirmers show more pattern differences between the low and high learning gainers. We can see that the high learning gain group follow the patterns of *_FF*, *FS*, *_FS*, *SS_*, *_SS*, *SS*, and *Ss* more frequently. This means that the high learning gain, non-confirmer students tend to continue trying a non-parameterized exercise and spending time on it after they failed in it or it took them a long time to get to the correct answer for that exercise. In this sense, these students are closer to the confirmer group of students (cluster 1) but only at the times that they are not sure if they have learnt the solution to an exercise. On the other hand, the low learning gain group tend to develop the *fs_*, *_fs_*, and *_ff* patterns in their sequences. The first two indicates that they give up practicing the exercise after having a short success that comes after a short failure. Also, they tend to repeat short failures on the same exercise more often.

Comparing beneficial and harmful patterns for the two clusters, we can make an interesting observation that the increased use of several beneficial patterns for each cluster make students more similar to the opposite cluster. For example, while confirmers have a generally low tendency to stop after first hard success *_FS_*, successful confirmers demonstrate this pattern much more frequently. On the other hand, while non-confirmers generally tend to stop after first hard success, successful non-confirmers have higher tendency to continue after hard success as shown by significantly increased frequencies of such patterns as *SS_*, *_SS_*, and *Ss*. In other words, while the two clusters are considerably different by their behavior overall, the "centrist" students that are closer to the opposite cluster tend to be more successful, while the extreme behavior that distinguishes the cluster is frequently related to less successful performance.

6. CONCLUSIONS AND FUTURE WORK

In this paper we explored patterns of student repetitive work with parameterized exercises for Java programming domain. The goal of this work was to understand the connections between micro- and macro-level behavior patterns and factors that might be responsible for this behavior such as exercise difficulty, student personality, level of knowledge, or position in the course. In turn, we hoped that this understanding could help us predict how a specific student would work with an exercise and prevent inefficient behavior such as repetitive successful attempts when the exercise become too easy to contribute to student knowledge growth. To explore the impact of students' personal features on their work with exercises, we built the student *problem solving genome*, a compact representation that encapsulates the specifics of individual behavior patterns. To build the genome, we started with micro-patterns (genes) that describe small chunks of repetitive behavior based on correctness and duration of each attempt. We then constructed a genome as a frequency profile that shows the dominance of each gene in the student behavior.

Using the genome approach we analyzed the stability of behavior patterns for students and groups and explored their connection with student success in the course. The most interesting finding was a considerable stability of the genome on individual level. As our analysis showed, the genome uniquely identifies a user among other users over the whole duration of the course despite a considerable growth of student knowledge over the course duration. While the problem complexity does affect the behavior patterns as well, we demonstrated that the genome is defined by some inherent characteristics of the user rather than a difficulty profile of the problems she solves.

To find a connection between the problem-solving genome and student performance, we examined genomes for various groups of students. Since a direct attempt to associate genome with performance-related groups (a typical way to group students in educational contexts) was not successful, we started from the opposite side and formed student groups on the basis of their genome similarity. As it appears, all students could be most reliably split into just two cohorts that differ considerably by their behavior. After that split, we were able to contrast successful and less successful learners by their behavior and identify "beneficial" and "harmful" genes for each cohort. In particular, it was interesting to observe that the behavior of successful learners in one cohort was somewhat closer to the behavior of the opposite cohort.

Note that the reported finding are limited to a specific context - non-mandatory work with Java programming exercises. It is not clear whether problem solving behavior patterns in other domains or the same domain with mandatory exercises will exhibit the same properties. We also believe, however, that the "genome" approach provides a new way for exploration of student problem solving behavior and plan to explore to the stability of the "genome" and the presence of behavior cohorts in other domains and contexts. In addition, we would like to proceed to our ultimate goal of recognition and prediction of inefficient behavior. The discovery of a stable genome provides a good ground for developing a recognition engine and the presence of behavior cohorts indicates that some good guidance (encouraging "beneficial" patterns and discouraging "harmful" ones) could be provided even in the early stage of student work when it might be harder to build a reliable genomic profile.

7. ACKNOWLEDGMENTS

Julio Guerra is supported by Chilean Scholarship (Becas Chile) from the National Commission for Science Research and Technology (CONICYT, Chile), and the Universidad Austral de Chile.

8. REFERENCES

[1] Ayres J, Flannick J, Gehrke J, Yiu T (2002) Sequential pattern mining using a bitmap representation. KDD 2002, 429-435

[2] Bouchet, F., Kinnebrew, J. S., Biswas, G., and Azevedo, R. (2012). Identifying Students' Characteristic Learning Behaviors in an Intelligent Tutoring System Fostering Self-Regulated Learning. In EDM (pp. 65-72).

[3] Brusilovsky, P. and Sosnovsky, S. (2005). Individualized exercises for self-assessment of programming knowledge: An evaluation of quizpack. ACM Journal on Educational Resources in Computing, 5(3):Article No. 6, 2005.

[4] Herold, J., Zundel, A., and Stahovich, T. F. (2013). Mining Meaningful Patterns from Students' Handwritten Coursework. In EDM 2013 (pp 67-73).

[5] Ho, Joshua, Lior Lukov, and Sanjay Chawla. Sequential pattern mining with constraints on large protein databases. In Proc. of COMAD 2005b, pp. 89-100. 2005.

[6] Hsiao, I-H. and Brusilovsky, P. (2012) Motivational Social Visualizations for Personalized E-learning, In: Proc. of ECTEL 2012, Springer-Verlag, Volume 7563/2012, pp.153-165.

[7] Hsiao, I. H., Sosnovsky, S. , and Brusilovsky, P. (2009). Adaptive navigation support for parameterized questions in object-oriented programming. In ECTEL 2009, volume 5794 of LNCS, pages 88-98. Springer-Verlag.

[8] Kashy, D. A., Albertelli, G., Ashkenazi, G., Kashy, E., Ng, H.-K., and Thoennessen, M. (2001). Individualized interactive exercises: A promising role for network technology. In 31st ASEE/IEEE Frontiers in Education Conference. IEEE, 2001.

[9] Kashy, E., Thoennessen, M., Tsai, Y., Davis, N. E., and Wolfe,S. L. (1997). Using networked tools to enhanse student success rates in large classes. In 27th ASEE/IEEE Frontiers in Education Conference, volume I, pages 233-237. Stipes Publishing L.L.C., 1997.

[10] Kinnebrew, J. S., and Biswas, G. (2012). Identifying Learning Behaviors by Contextualizing Differential Sequence Mining with Action Features and Performance Evolution. In EDM (pp. 57-64).

[11] Kinnebrew, J. S., Loretz, K. M., and Biswas, G. A. U. T. A. M. (2012). A contextualized, differential sequence mining method to derive students' learning behavior patterns. Journal of Educational Data Mining.

[12] Kortemeyer, G., Kashy, E. , Benenson, W., and Bauer, W. (2008). Experiences using the open-source learning content management and assessment system lon-capa in introductory physics courses. American Journal of Physics, 76(438), 2008.

[13] Martinez, R., Yacef, K., Kay, J., Al-Qaraghuli, A., and Kharrufa, A. (2011). Analysing frequent sequential patterns of collaborative learning activity around an interactive tabletop. In EDM 2011 (Vol. 13, No. 2, pp. 111-120).

[14] Mayer, R. E. (1998). Cognitive, metacognitive, and motivational aspects of problem solving. Instructional science, 26(1-2), 49-63.

[15] McAdams, D. P. (1995). What Do We Know When We Know a Person? Journal of Personality, 63(3), 365-396.

[16] Schunk, D. (1991). Self-efficacy and academic motivation. Educational Psychologist 26: 207-231.

[17] Sosnovsky, S., Brusilovsky, P., Lee, D. H., Zadorozhny, V., and Zhou, X. (2008) Re-assessing the Value of Adaptive Navigation Support in E-Learning. In proc. of AH 2008, Springer Verlag, pp. 193-203

[18] Von Luxburg, U. (2007). A tutorial on spectral clustering. Statistics and computing, 17(4), 395-416.

[19] Weiner, B. (1986). An Attributional Theory of Motivation and Emotion. New York: Springer-Verlag.