文章编号：1672-5913(2018)12-0048-09 中图分类号：G642
DOI:10.16512/j.cnki.jsjjy.2018.12.008

# Analyzing Student Behavior in Online Programming Courses

Xinyu You, Bohong Liu, Menghua Cao, Tao Wang, Yue Yu, Gang Yin

**Abstract:** Rather than maintaining the classic teaching approach, a growing number of schools use the blended learning system in higher education. The traditional method of teaching focuses on the result of students' progress. However, many student activities are recorded by an online programming learning platform at present. In this paper, we focus on student behavior when completing an online open-ended programming task. First, we conduct statistical analysis to examine student behavior on the basis of test times and completed time. By combining these two factors, we then classify student behavior into four types by using k-means algorithm. The results are useful for teachers to enhance their understanding of student learning and for students to know their learning style in depth. The findings are also valuable to re-design the learning platform.

**Key words:** educational data mining; learning analysis; student behavior; online programming; blended learning environment

## 1 Introduction

Online education has revolutionized traditional education in the past years. Many higher education schools use web-based learning system to deliver online education in a blended learning academic environment. Blended learning includes both face-to-face classroom style guidance as well as online methods [1]. This learning environment is especially suited to programming courses due to the high level of interaction and rich multimedia of online programming courses.

• Xinyu You, Bohong Liu, Menghua Cao, Tao Wang, Yue Yu, Gang Yin. Computer Science College, National University of Defense Technology, Changsha, China. E-mail: yxydiscovery@gamil.com, yorhaz40@gmail.com, floweropenbb@gmail.com, taowang2005@nudt.edu.cn, yuyue@nudt.edu.cn.

In online programming courses, each lesson typically involves coding tasks. Students code using a compiler online to finish a task and get a score. Most students get similar scores as soon as they finish all the tasks. In this situation, observing students' learning behaviors is difficult for instructors [2]. To solve this problem, students' activities are recorded by a system. The log data can then be used as a source for analyzing student behavior in the fields of educational data mining and learning analytics to investigate improvements of learning, teaching, and re-designing of the learning platform.

In our work, we focus on two factors, namely test times and completed time, that correlate with learning. By using these two factors, we analyze students' behavior during online programming and automatically classify them into different learning types.

The paper is structured as follows. Section

2 introduces related work. Section 3 provides methodology about the dataset and cluster algorithm, and Section 4 presents the analysis and clustering results. Finally, Section 5 offers concluding remarks and proposes directions for future research.

## 2 Related Work

Blikstein extensively used quantitative techniques to analyze students' behavior and categorize them by their different programming strategies. Students' programming actions were logged and collected by the offline Net Logo programming environment. Blikstein then focused on one student and explained her coding strategy by code size, compiler error, and time between compilation over time [3].

Based on Massive open online course, Anderson A et al. analyzed student engagement. They formed a taxonomy of personal behavior, examined students' behavioral patterns, and investigated how forum participation is associated with other factors during a course. The students' engagement styles are divided into three categories, namely, viewers, solver, and all-rounders, on the basis of watching lecture video and handling in assignments [4].

Estacio R R et al. used Moodle, which is a blended online learning management system, to filter and analyze student behavior data using a machine learning technique (i.e., vector space model). The result shows that data mining algorithm can quantify the data into a single numeric value that can be used to generate visualizations of students' level of activity [5].

With the online learning platform as basis, we focus on the test times and completed time of programming courses. We analyze these two factors and use k-means clustering algorithm to automatically classify students.

## 3 Methodology

In this section, we first give the method framework of the entire process. We then introduce the dataset and data preprocessing, including data cleaning and course filtering. Finally, we present how we apply k-means clustering algorithm to classify students.

### 3.1 Method framework

Figure 1 illustrates the method framework of analyzing and evaluating student behavior in online programming courses.

The method framework consists of three main processing steps.

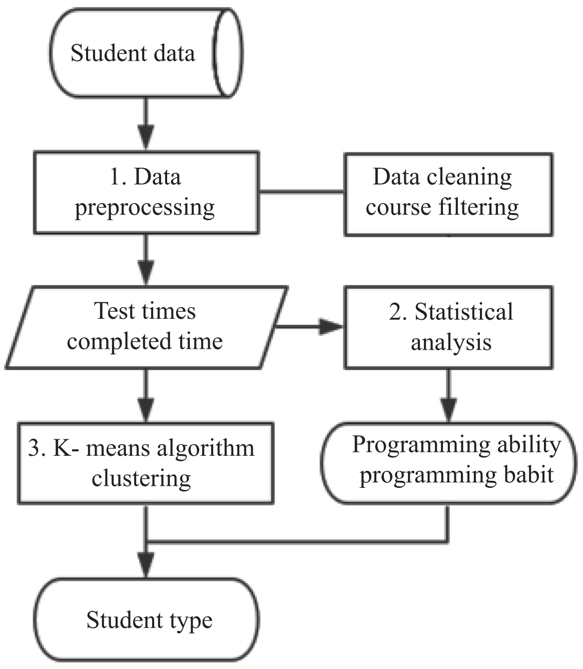

Student data
1. Data preprocessing
Data cleaning course filtering
Test times completed time
2. Statistical analysis
3. K- means algorithm clustering
Programming ability programming babit
Student type

**Fig. 1 Method framework.**

Step 1. Data preprocessing.

Before analyzing the student data, we first clean the data. We then select the experimental courses and extract test times and completed time from the relevant data table.

Step 2. Statistical analysis.

We conduct statistical analysis on the two factors to obtain the students' programming ability in class as well as detailed information on their programming habits.

Step 3. K-means algorithm clustering.

Using k-means algorithm, we cluster the students considering their test times and completion time. Combined with the statistical analysis results, the characteristics of each type of students are explained, providing a basis for personalized learning guidance.

## 3.2 Dataset

Our dataset came from Educoder, which is a web-based platform for online programming learning offered by the National University of Defense Technology, Changsha, China. The platform is mainly for higher education students to offer blended learning. Table 1 shows the essential concept of this platform.

The courses include Programming Languages, Subject Basis, and Data Structures. These courses are offered to undergraduate and postgraduate students. Each course can have two to eight lessons, and each lesson contains 1 to 10 levels. Each level has varied scores according to difficulty, and students are required to complete all levels to pass the course. A student can run codes multiple times to get the full score of any lesson before the deadline, but the student will get zero points if he/she reads the answer.

**Table 1 Related concept and description for research.**

| Concept | Description |
|---|---|
| Course | Course offered by a teacher |
| Student | Student who joined the course |
| Lesson | Lesson in the course |
| Level | Level in the lesson |
| Score | Score of each level |
| Test | Student writes or modifies the code and click submit, then the system reviews the program. |
| Test Times | Total clicks/times to submit |
| Pass Level | Test is successful. |
| Time | Time interval from opening a level to passing the level |

The size of the original dataset is 5G, and it contains 1,025 courses, 312 lessons, and almost 30,000 users.

A brief review of various significant data in Educoder database is presented in tabular form (Table 2).

**Table 2 Tables and relative information in database.**

| Name of Table | Information |
|---|---|
| courses | Information about a course |
| students_ for_courses | Correspondence information between students and courses |
| homework_ commons | Correspondence information between published homework and courses |
| homework_ commons_lessons | Correspondence information between homework and lesson |
| lessons | Information about a lesson |
| mylessons | Information about the lesson created by individuals |
| challenges | Correspondence information between challenges and lesson |
| levels | Information about the levels created by individuals |

## 3.3 Data preprocessing

Data preprocessing is a crucial step in our research because a large amount of datasets in Educoder have missing values, noisy data, or irrelevant and redundant information. The main step to filter the dataset is course selection.

Given a large set of courses and lessons have been published, we first chose the formally published lessons to get the finished and representative data. In addition, the number of visits in a lesson should be more than 500 and less than 5 000. We required every lesson to consist three to four levels. After filtering, 55 qualified lessons were left.

We then found the lesson-corresponding courses by using the table homework_commons. Taking into account the different university teaching styles, training modes, sufficient data samples, we used the following criteria to filter. According to the statistics, five satisfied courses were noted.

- School ID, which chooses School ID 117, i.e., National University of Defense Technology.
- Visits, which should be more than 1000 visits in a class.
- Course size, which selects more than 100 students in a course.

We checked the pass rate for students based on the selected courses and lessons. We defined N_g as the number of levels in a lesson and N_p as the number of pass levels for a student in the lesson. As shown in Eq. (1), we applied a variable named PassOrNot to determine whether a student passed the level or not. We also used Eq. (2) to calculate the pass rate of a course, in which N is the number of students in the course.

In Fig. 2, the horizontal axis is course ID linked with lesson ID, and the vertical axis is course pass rate. Figure 2 indicates a significant difference in the pass rate for our selected courses.

$$\text{PassOrNot} = \begin{cases} 1, N_p = N_g \\ 0, \ N_p < N_g \end{cases} \quad (1)$$

$$\text{CoursePassRate} = \frac{\sum_{i=1}^{n} \text{PassOrNot}_i}{N} \quad (2)$$

$$\text{CoursePassRate} = \frac{\sum_{i=1}^{n} \text{PassOrNot}_i}{N}$$

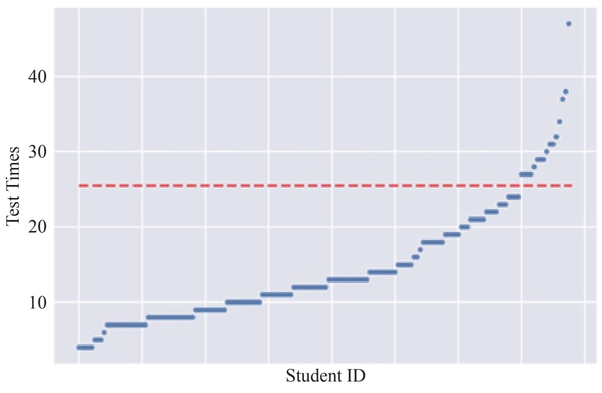


**Fig. 2 Pass rate of ClassID_LessonID in selected classes.**

Overall, we selected the Course 1144 “Computer Foundation in Fall 2017 by Ren Xiaoguang” and Lesson 157 “Tuples and Dictionaries for Getting Started with Python” to facilitate the analysis and eliminate the effect of objective factors. This lesson’s dataset contains 157 students, 4 levels, and 10,456 code submissions. Among the students, 98.0% succeeded at passing all levels, but 2% did not get the full score. Table 3 shows the detail of levels in Lesson 157.

**Table 3 Details of levels in Lesson 157.**

| Level | ID | Subject | Score |
|---|---|---|---|
| Level 1 | 417 | Use of tuple | 100 |
| Level 2 | 444 | Traverse dictionary | 200 |
| Level 3 | 445 | Nested | 300 |
| Level 4 | 472 | Use of dictionary | 100 |

### 3.4 K-means algorithm

K-means clustering algorithm is first proposed by Hugo Steinhaus in 1957 [6]. In 1967 [7], J. MacQueen formally proposed the initial definition of k-means in his paper. The most commonly used method of initializing clustering centers is proposed in 1979 [8] by Hartigan, J. A.

K-means clustering algorithms aim to divide n samples into k clusters so that the within-cluster sum of squares is minimized, which is based on distance concepts, to update the clusters iteratively. In this paper, K-means clustering algorithms were applied to group students based on two characteristics: test times and completed time.

Suppose we have n samples from the data set, for sample i, its test times is $C_i$, its completed time is $T_i$, so the feature vectors is $X_i$ ($C_i$,$T_i$ ). They can be divided into k clusters, k < n. Furthermore, we standardized the data set to preprocessing, because outliers are present in the set and scaled data have zero mean and unit variance. As shown in Eq. (3), $\mu$ is the mean of all the sample data, and $\sigma$ is the standard deviation of all the sample data.

$$x^* = \frac{x - \mu}{\sigma} \quad (3)$$

After the normalization, we applied k-means algorithm. The pseudocode of k-means algorithm is organized as follows:

```
Algorithm 1: K-means algorithms
  Input: Dataset D = $\{X_1, X_2, ..., X_n\}$
  Output: Cluster centers C = $\{C_1, C_2, ..., C_k\}$
Randomly select k cluster centers;
while the clusters are changing do
    for $X_i = X_1, X_2, ..., X_n$ do
        for $C_i = C_1, C_2, ..., C_k$ do
            $u_i$ represents the nearest distance of sample i to one class
             $u_i = argmin_{i \in \{1,2,...,k\}} \|X_i - c_i\|^2$
        end
    end
    for $C_i = C_1, C_2, ..., C_k$ do
        recalculate the centroid of the class $C_i = \frac{1}{n} \sum_{i=1}^{n} X_i^j$
    end
end
```

## 4 Analysis

In this section, we first analyzed the test number factor and then the completed time factor that a student spends in a course separately. We combined the two factors to classify the student into different types and investigated each type’s characteristic.

### 4.1 Test number

One of the key characteristics of a student is the number of tests, which reflects the number of attempts a student has made before passing a level.

On the one hand, we can analyze a student' s learning type through the test factor. On the other hand, we can estimate the difficulty of a level on the basis of number of tests. In this paper, we focused on the student' s learning type.

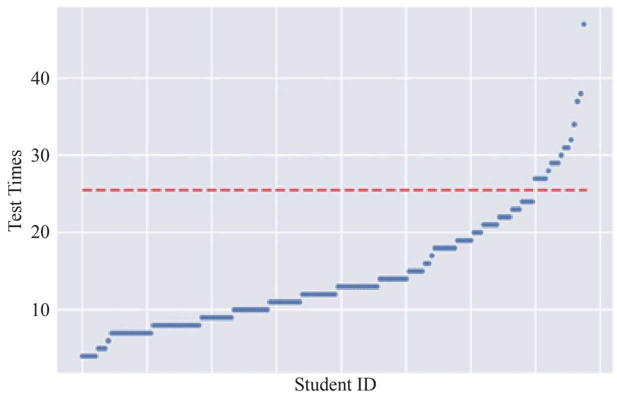


**Fig. 3 Test times of students in Lesson 157 .**

Figure 3 shows the Python lesson' s test times of the whole class. The abscissa in the figure represents the students' serial number, and the ordinate represents the total number of tests in the Python lesson sorted in ascending order. Each blue dot represents a specific student' s test number. The gray dashed line stands for the median level, which is 14.43.

The total test times for the Python lesson is 2265. Figure 4 indicates that the range of test number is between around 5 and 50.

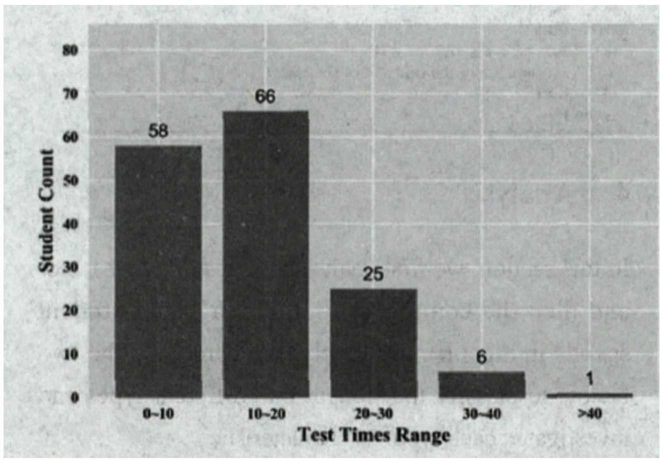


**Fig. 4 Distribution of students in test-time range with Lesson 157.**

Figure 4 shows the distribution of students in different test intervals. The number of tests in ranges 0~50 and 50~100 are 36% and 42.78%, respectively. The test number within 100 times accounts for 81.14%, but the rest is only 20%. Especially, only one student' s test number is over 200.

We then analyzed each level' s test-time situation. The total number of the first level to the fourth one is 427, 346, 582, and 810.

Figure 5 shows the details of each level' s test-time result. The first and third levels cost more test times.

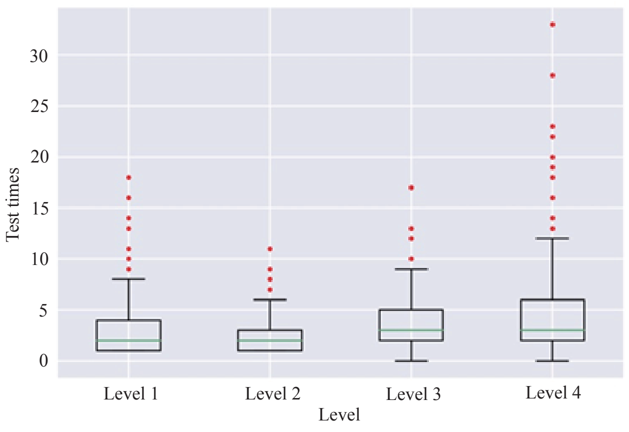


**Fig. 5 Test times in each level.**

For those who have passed the lesson and got full scores, we chose three students to explain how the test times implies their personal behavior. Table 4 provides the detailed information of their test times.

**Table 4 Detailed information of students' test times.**

| | Level 1 | Level 2 | Level 3 | Level 4 | Total |
|---|---|---|---|---|---|
| Zhang | 1 | 4 | 8 | 33 | 46 |
| Deng | 16 | 1 | 4 | 1 | 22 |
| Zhou | 1 | 1 | 3 | 3 | 8 |

Zhang, whose test number is 46, is the top student in the class. Table 4 indicated that he stalled in level 4. He tested much more times but finally got the scores. His programming ability may be weaker than others but he exhibits strong persistence.

Deng, unlike the other two, had spent more test times on the first level.

Zhou took the least times among the three students.

We then analyze the remaining four students who did not get the full scores.

Tables 5 and 6 show those who did not get the full scores. Meng and Xin did not try at all. Du and Li tried over 10 times but still failed.

**Table 5 Scores per level of four special students .**

| | G1 | G2 | G3 | G4 |
|---|---|---|---|---|
| Meng | 100 | 100 | 0 | 0 |
| Xin | 100 | 100 | 200 | 0 |
| Du | 100 | 100 | 0 | 300 |
| Li | 0 | 100 | 200 | 300 |

**Table 6 Test times per level and total scores of four special students .**

| | G1 | G2 | G3 | G4 | Total | Score |
|---|---|---|---|---|---|---|
| Meng | 1 | 7 | 0 | 0 | 8 | 200 |
| Xin | 6 | 2 | 3 | 11 | 22 | 400 |
| Du | 3 | 1 | 10 | 7 | 21 | 500 |
| Li | 3 | 2 | 9 | 4 | 18 | 600 |

## 4.2 Completed time

### 4.2.1 Total time

The total time a student spends on a course is one of the most significant indicators of student behavior. Total time varies from one student to another.

Figure 6 shows the Python lesson’s total time of the whole class. The x-axis is the student ID, and the y-axis means the total time, taking a logarithmic value. The median line of total time is 208.5 minutes, which is marked with the red dotted line. This line is regarded as the average level of total time in this lesson.

For example, the red dot shown in this figure is Yue, whose ID is 26112, who spent 1 472.58 minutes on this lesson. He completed this task with more time than the average level.

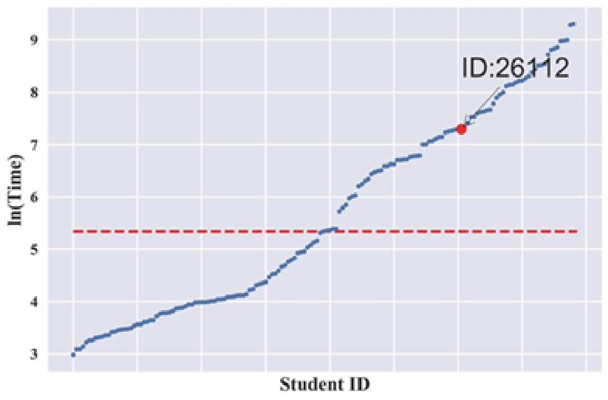


**Fig. 6 Total time of students in Lesson 157.**

### 4.2.2 Time in each level

Now the question is why he spent so much time on this lesson. By analyzing the time cost in each level, his coding characteristic in the time aspect can be explored in depth.

Figure 7 shows how much time Yue spend in each level.

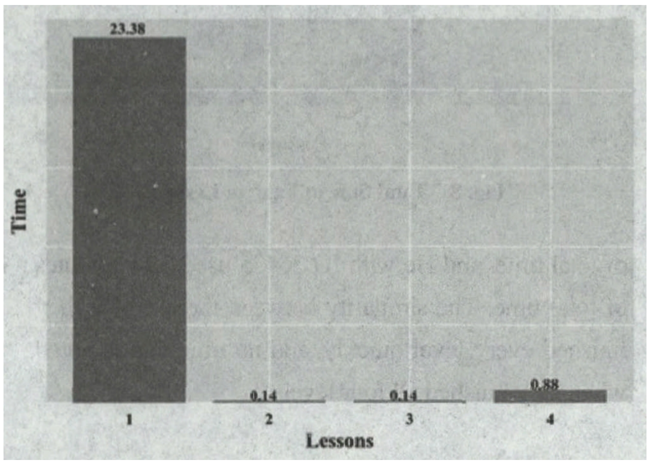


**Fig. 7 Total time of Yue in Lesson 157.**

(A)Yue started his first level in 2017-11-03 at 18:05, and he finished it at 17:28 in next day. He could not spend all these 23 hours on this level. This student must have gotten into trouble with the first level or did other things when he opened it. Then, he chose to find another time to finish it.

(B)When he finished the first level, he immediately opened the next level. This time he had finished it quickly. Then, he completed all of the four levels in this lesson without a break.

Using this analysis, we can examine the time characteristics of other students in search of differences. In the following, we show figures(Fig. 8–Fig. 11) of four different students (Yang, Wang, He, and Ma), which include their times in each level.

First, we examined Yang. His ID is 26512 and total time is 3 727.25 minutes, which is longer than Yue’s. He finished the first level quickly, but after that, he did not start the next level immediately. He had a time gap between every time he finished a level.

However, Wang with ID 26367 used 28.62 minutes

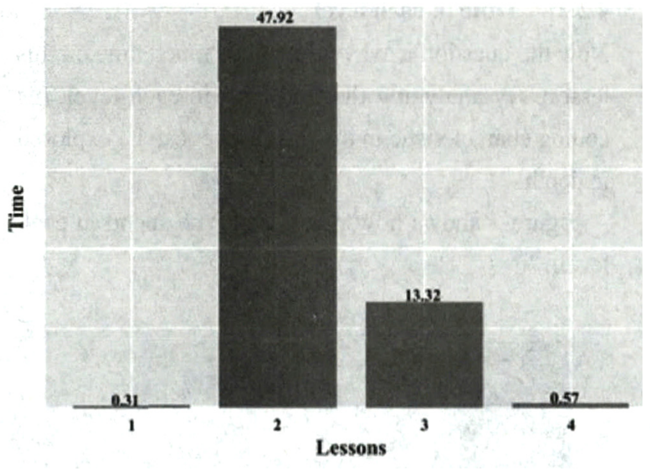


**Fig. 8 Total time of Yang in Lesson 157.**

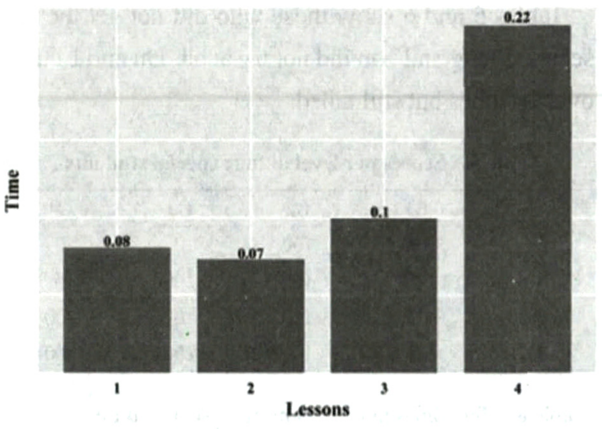


**Fig. 10 Total time of Wang in Lesson 157.**

of total time, and He with ID 26425 used 35.4 minutes of total time. The similarity between them is that they finished every level quickly, and no time gap is noted when they finished all four levels.

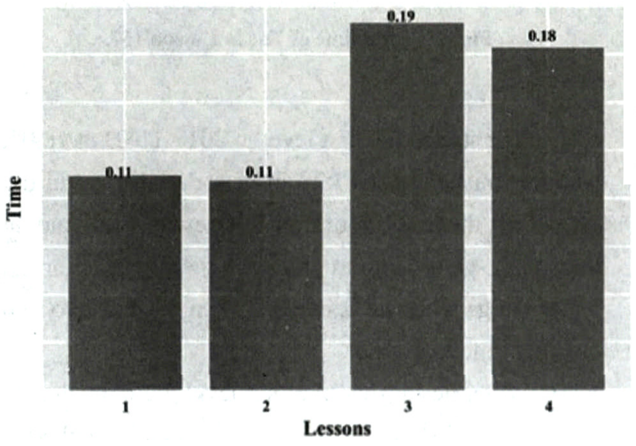


**Fig. 9 Total time of He in Lesson 157.**

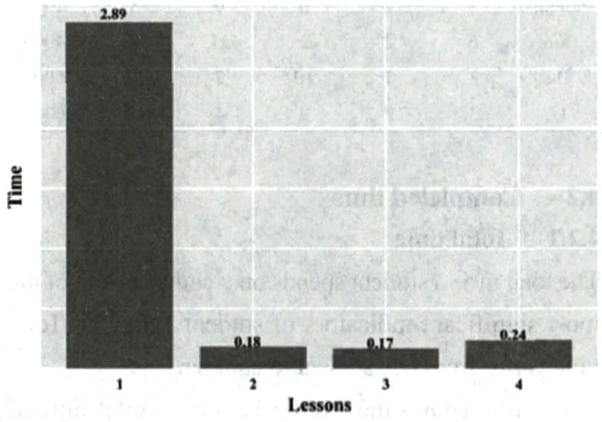


**Fig. 11 Total time of Ma in Lesson 157.**

Finally, we examined Ma, whose ID is 26717, with the median total time. This student started the first level at 20:42 and had trouble with it. Nevertheless, he quickly found a way to deal with it, and then completed this lesson with no break.

These analyses indicate that the reason for large total time is the existence of time gap. Four types of time strategies can be inferred.

(1) Solve problems at the starting point and finish all of the tasks without break.

(2) Solve problems at starting point but shelve when experiencing difficulty. Find tasks easy when finished immediately.

(3) Have the habit of dealing with problems by dividing them into different stages and taking a break upon completing a stage.

(4) Find solving the problem difficult and take a long time with no progress.

## 4.3 General analysis

Before the cluster, a normalization was applied to the index set. K-means Clustering was then employed to classify students on the basis of their time and test times characteristics.

Figure 12 and Table 7 show that students are classified into four clusters. The result suggests the following classification of students:

Type A: With little time and large commit numbers, students in this block complete a lesson rapidly.

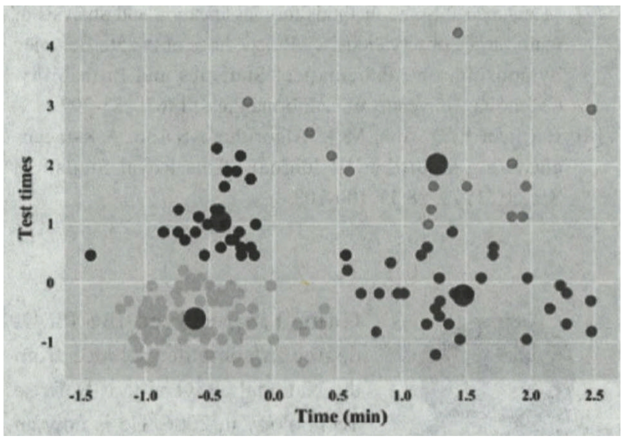


**Fig. 12  Cluster results based on their time and test times.**

**Table 7  Coordinate of four clusters.**

| | Cluster1 | Cluster2 | Cluster3 | Cluster 4 |
|---|---|---|---|---|
| Time | − 0.61 | − 0.41 | 1.27 | 1.47 |
| Test times | − 0.60 | 1.04 | 2.01 | − 0.21 |

However, before finding the correct answer, they change the codes so many times. They are in the majority.

Type B: With little time and small commit numbers, students in this block quickly finish a lesson and they do not change the codes that much. They appear to be at the top of their class.

Type C: With long time and large commit numbers, students in this block spend relatively more time to finish a lesson. They also change their codes many times. They may experience difficulty in the lesson.

Type D: With long time but small commit numbers, students in this block spend substantial time finish a lesson, but they do not change their codes that much. Despite the time, they concentrate on the quality of their codes.

For different types of students, teachers can give appropriate advice to help them improve their learning habits.

For types C and D, teachers should closely supervise the time spent on a lesson. Types A and C have large commit numbers, we can also give hints like knowledge tips or error message in our future platform design. For type B with high capacity, teachers can give them challenging lessons. At the same time, our platform can encourage students under this type to help others who are weak. Type B students can provide solutions and answer questions in the discussion area and get bonus points.

## 5 Conclusion

The goal of this study is to obtain a deep understanding of student behavior in blended online programming courses. The study indicates that although the final scores of the students are similar, their behaviors significantly vary. By statistical analysis, we find two factors, namely, test times and completed time, that can indicate different behaviors of students. To better understand students' behaviors, we use k-means clustering to classify them. We arrive at four classifications of behavior types by combining the two factors. According to the study results, teachers can have a direct view of their students' situation. Furthermore, they can take different teaching activities based on the students' behavior types in their class. We hope this study can improve the quality of teaching and the performance of students in the future.

## Acknowledgement

The research is supported by the National Grand R&D Plan (Grant No. 2016YFB1000805) and National Natural Science Foundation of China (Grant No.61702534, 61432020, 61472430, 61502512).

**References**

[1] Osguthorpe R T, Graham C R. Blended learning environments: Definitions and directions[J]. Quarterly Review of Distance Education, 2003, 4(3): 227-233.
[2] Mor E, Minguillón J, Carbó J M. Analysis of user navigational behavior for e-learning personalization[J]. WIT Transactions on State of the Art in Science and Engineering, 2013（2）: 227-246.
[3] Blikstein P. Using learning analytics to assess students' behavior in open-ended programming tasks[C]//Proceedings of the 1st International Conference on Learning Analytics and Knowledge. Banff, Alberta: ACM, 2011: 110-116.
[4] Anderson A, Huttenlocher D, Kleinberg J, et al. Engaging with massive online courses[C]//Proceedings of the 23rd

International Conference on World Wide Web. Seoul: ACM, 2014: 687-698.

[5] Estacio R R, Raga Jr R C. Analyzing students online learning behavior in blended courses using Moodle[J]. Asian Association of Open Universities Journal, 2017, 12(1): 52-68.

[6] Steinhaus H. Sur la division des corps matériels en parties[J]. Bull Acad Polon Sci Cl III, 1957, 4(12): 801-804.

[7] MacQueen J. Some methods for classification and analysis of multiVariate observations[C]//Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability. California: University of California Press, 1967: 281-297.

[8] Hartigan J A, Wong M A. Algorithm AS 136: A K-means clustering algorithm[J]. Journal of the Royal Statistical Society, 1979, 28(1): 100-108.

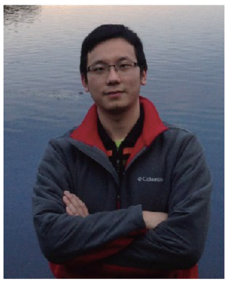
**Xinyu You** received the B.S. in software engineering from the Chongqing University, China in 2017. At present, she is a graduate student at Computer College of National University of Defense Technology. Her current interest is educational data mining in higher education.

**Bohong Liu** received the B.S. in computer science from Lanzhou University in 2017. At present, he is a graduate student at Computer College of National University of Defense Technology. His current interests focus on the improvement of online programing courses.

**Menghua Cao** received the B.S. from Yunnan University in 2017, who is major in Digital media technology. Now, she focuses on blended learning as a graduate student at National University of Defense Technology.

**Gang Yin** received the Ph.D. degree in computer science from the National University of Defense Technology in 2006. He is now an associate professor in NUDT. His current research interests include distributed computing, information security and software engineering, and machine learning. He is a member of the IEEE.

**Tao Wang** received both his M.S. and Ph.D. in computer science from National University of Defense Technology, in 2010 and 2014. His work interests include open source software engineering, machine learning, data mining and knowledge discovering in open source software.



**Yue Yu** received his Ph.D. degree in computer science from National University of Defense Technology in 2016. His current research interests include software engineering, spanning from mining software repositories and analyzing social coding networks.

（Publishing Editor：Yuanhong Peng）