Regular Article

An Efficient Approach to Measure the Difficulty Degree of Practical Programming Exercises based on Student Performances

Huy Tran, Tien Vu Van, Hoang Nguyen Viet, Duy Tran Ngoc Bao, Thinh Tien Nguyen, Thanh Van Le

Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, VNU-HCM, Ho Chi Minh City, Vietnam

Correspondence: Huy Tran, huy.tran14@hcmut.edu.vn

Communication: received 23 August 2021, revised 20 September 2021, accepted 21 September 2021
Online publication: 23 October 2021, Digital Object Identifier: 10.21553/rev-jec.282
The associate editor coordinating the review of this article and recommending it for publication was Prof. Tran Manh Ha.

Abstract– **This study examines the generality of easy to hard practice questions in programming subjects. One of the most important contributions is to propose four new formulas for determining the difficulty degree of questions. These formulas aim to describe different aspects of difficulty degree from the learner's perspective instead of the instructor's subjective opinions. Then, we used clustering technique to group the questions into three easy, medium and difficult degrees. The results will be the baseline to consider the generality of the exercise sets according to each topic. The proposed solution is then tested on the data set that includes the results of the two subjects: Programming Fundamentals, Data Structures and Algorithms from Ho Chi Minh City University of Technology. The most important result is to suggest the instructors complete various degrees according to each topic for better evaluating student's performance.**

Keywords– **e-learning, difficulty degree, automatic question classification, student's perception, effective coverage exercise.**

1 Introduction

With the help of technology support, teaching and learning processes can be deployed entirely on the Internet [1]. People who join an online course can access all necessary resources with no restrictions and do the assessment tests. The instructors can observe and evaluate the learning process of course participants through well-designed tests.

The platform for online teaching and learning as above is normally called e-learning education system, becoming more and more popular in schools, particularly in universities [2]. The higher level of education, the higher requirement of self-study in the learning process of learners. Moreover, thanks to the online environment, learners can preview educational materials at home and take tests at any time. Overall, the process will become more effective when applying e-learning to teaching and learning. Realizing the above learning foundation's effectiveness, the Faculty of Computer Science and Engineering (CSE) of Ho Chi Minh City University of Technology (HCMUT) has developed and deployed an Auto Grading System (AGS) (see Section 3.1) starting since Semester 1 of 2019.

The world is currently experiencing the COVID-19 pandemic because of its widely substantial spread. To reduce the risk of disease spreading, the government requires keeping social distance and encourages the online education process. In HCMUT, the AGS can completely satisfy the above demand for the programming practice lessons. If the system has sufficiently good exercises, can encourage learners, and is supported by instructional materials, teaching practice programming can be done essentially online.

Besides the above benefits, the programming support system's common drawback is the lack of mechanisms to encourage the appropriate learning and provide a suitable learning path for each learner. Learners usually need to be instructed how to do exercises from easy to complex and the number of questions at each degree should be appropriate. In the case where the question bank has too many easy questions, learners will be bored with them. In the other case where there are many difficult questions, they will be discouraged and gives up. Therefore, an exercise with a sufficient number of questions and appropriate difficulty degree will provide an incentive environment for learners.

The difficulty degree of each question mentioned here should be based on the learners' view, although the actual result of learners depends on the evaluation process including solution designed and grading scale estimated solely defined by instructors. Typically, the average grade point of students is a factor that instructors frequently observe and refer to represent difficulty degree. However, if there is only one factor to consider, there will be a lack of perspective on other aspects, resulting in a one-sided evaluation. Therefore, in this paper, some other factors will be proposed and considered since difficulty degree of learner's point of view may also be revealed through problem solving progress indicators such as the number of submission trials, the solving time duration, etc.

The process of creating questions and assessment their difficulty degree is only from teacher's subjective opinion. The authors in [3] point out that teachers only accurately estimate a fraction of question difficulty compared to learners. That statement raises the question of whether a programming exercise, which is a group of questions, has covered enough different degrees for learners to practice. From our proposed methodology, taking one step further, we investigate this coverage problem. The results provide an opportunity for teachers to look back to exercises by difficulty level and give teachers some direction to improve the question bank.

To the best of our knowledge, this is the first time the difficulty-related factors are explored in depth and tackled. Our contribution is fourfold:

1) The investigation of studying important factors that affect difficulty degree of programming practice questions.
2) The development of clustering questions based on the learner's perspective.
3) A comparison between difficulty-related factors and between the learner's and teacher's perspective in the view point of difficulty degree of question bank.
4) A real-life application of the problem to detect the lack of ease-to-difficulty degree of each subject in the question bank on the learner's perspective.

The rest of the paper is structured as follows. Section 2 presents related researches about factors that affect question difficulty. Section 3 focuses on our proposed approach to difficulty-related formulas and clustering approach; Auto Grading System will also be briefly introduced in this section. Section 4 shows the experimental results and our evaluation for question classification task. Section 5 will consider coverage of programming topics by making statistics on classification results. Finally, Section 6 concludes this study with results and future researches.

2 Related Work

2.1 Difficulty-Related Factors

Average score is a factor commonly used to evaluate the difficulty of questions. For instance, the authors Simon et al. only used students' average marks to measure difficulty of programming examination questions [4]. In addition, the ratio of students' marks to the number of students as a weight is proposed by Mahatme's group for categorizing questions in e-learning environment research [5].

Other factors are also considered to describe the difficulty degree of question. When predicting student performance using data from an Auto-grading system, the authors in [6] select four features: the individual passing rate of the best submission, individual testcase outcomes of the best submission, the time interval between the time of submissions and the task deadline, and the number of submissions for classification and regression tasks. In [7], the difficulty degree is also considered to be proportional to the total number of attempts for a problem. In 2018, Awat et al. did an item

Table I
Examined Difficulty-Related Factors

Factor	References
Average score	[5], [4]
Number of passed students	[8], [9]
Number of passed submissions	[9], [5]
Max score	[6]
Number of submissions	[6], [7]

analysis using the examination results of students [8]. One of the processes in performing item analysis is determining the difficulty level of an item. The item (question) difficulty is stated as the number of correct students divided by the total number of students.

With the objective to estimate the difficulty of programming problems, among information extracted from the dataset, Chowdhury et al. examined clustering by choosing the number of passed students, the number of passed submissions, and many others as clustering features [9].

Table I covers five difficulty-related factors that will be examined in this study.

2.2 Data Mining Techniques

The authors in [10], [11] have proposed a fuzzy genetic algorithm to estimate the real difficulty of the questions. *K*-means [12] algorithm is used to cluster difficulty degree in an e-learning environment [5] and *HackerRank* [7]. After examining many clustering techniques, the authors in [9] focuses on Fuzzy C-Mean Clustering algorithm to estimate the difficulty of programming problems which got high accuracy score on the testing set.

2.3 Programming Question Coverage

The authors Petersen et al. [13] have evaluated CS1 examinations from a range of schools across North America. They considered the distribution of question types and the average number of concepts besides question contents. The question contents include writing code, reading code, programming concepts and non-programming. The question types are multiple-choice, short answer, writing code, drawing diagrams. There are 28 question concepts in this research; some fundamental concepts are trivial syntax, variables, function structure and expressions.

3 Proposed Approach

3.1 Auto Grading System

Auto Grading System (AGS in short) is a system that supports practicing programming, built and used in the Faculty of CSE of HCMUT. In this research, we will focus on two key features of AGS, which are:

- Manage the questions bank (add/modify), and assign a set of questions to appropriate groups of students.

- Grade the submissions of students through an automatic mechanism.

The two sections below describe these two key features in detail.

3.1.1 Manage and assign the questions: The instructor, who manages the laboratory class, is required to setup an exercise for every topic of knowledge in a course. An exercise contains some relative questions to evaluate the students' understanding about the topic. One question in the AGS system must be configured with the main component, i.e., a suite of input as testcases for submissions. This suite is used in the grading process for submission.

After finishing setting up the exercise, the instructor needs to assign it to groups of students. Some interesting fields of data to configured in assigning question are:

- The maximum number of submissions.
- The start and end time for submitting an answer.

When the assigning step is done, students can start doing this exercise.

3.1.2 Automate grading the submissions of students: Whenever AGS system records a submission for a question from a student, it compiles the submission source code then runs with the configured testcases to produce a set of output. The submission score is determined by the number of correct testcases that represented in the output set.

Following this phase, we can collect all the score from submissions of students which forms the data source for this paper, including some useful information inferred from the data source, such as:

- The average score of submissions for a question
- The number of students that passed a threshold
- The number of submissions that passed a threshold
- The best submission of a student for a question
- The number of submissions of a student for a question

3.2 Difficulty-Related Formulas

For readability, in this research, all following words including *Average score*, *Passed submissions*, *Passed submissions*, *Best submission*, *Number of submissions* will be used as a factor or a formula name interchangeably.

Our research proposes 4 formulas that related to the difficulty of programming questions and constructed based on student's submission results. By observing all 4 formulas, we aim to describe the difficulty of programming questions based on student's performances. To increase the reliability of our research, we simultaneously compare those formulas with the average score formula, which is a commonly used formula to determine the difficulty degree of questions.

3.2.1 Average score: Average score is a factor that is widely used to describe difficulty as the mean of student scores. Because programming questions often have many submissions (as for trying and correcting), a student's score is the mean of all his/her submission scores, which is then normalized to the range [0-1] based on max score, then calculate the mean score according to students. The formula is stated as

$$F_0 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{C_i}\sum_{j=1}^{C_i}\frac{c_{ij}}{C_{\max}}\right), \tag{1}$$

where N is the number of students answering the question, C_i is the number of submissions of the ith student, c_{ij} is the score of the jth submision of the ith student, and $C_{\max}$ is the maximum score of the question.

3.2.2 Passed students: Passed students refers to information about the number of students who passed the test. Since a difficult question will have few students finding a solution within the time limit, students who do not have the submission will not be included. The formula is then normalized to range $[0 - 1]$. The proposed formula is

$$F_1 = \frac{S}{S_{tot}}, \tag{2}$$

where S is the number of passed students, and S_{tot} is the number of students who had at least one submission for the question.

3.2.3 Passed submissions: Passed submissions refers to the number of passed submissions for a question. For programming questions, students typically stop submit when they have passed them. For a difficult question, students will have a few failed requests before reaching the passed one. Therefore, the ratio of the number of passed submissions to the number of total submissions will be low. Conversely, for an easy question, the number of failed submissions is low and that ratio will be high. Additionally, if a person is recognized as failed on *Passed students*, Passed submissions gives extra information about the number of failed requests. The proposed formula is

$$F_2 = \frac{U}{U_{tot}}, \tag{3}$$

where U is the number of passed submissions, and U_{tot} is the number of total submissions.

3.2.4 Best submission: Best submission addresses the student's submission score. The easier the question, the higher the student's score on the question. During the time the question is open, it is possible that the student did not get a good score at the beginning, but after a while, the submission improved, the score increased. Hence, *Best submission* suggests taking the highest score in a student's submissions for the question

$$F_3 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\max\{Ci*\}}{C_{\max}}\right), \tag{4}$$

where N is the number of students, $C_{\max}$ is the maximum score, and C_{i*} is the score set of the ith student's submissions for the question.

3.2.5 Number of submissions: Number of submissions refers to the number of times a student needs to work to achieve a question. The harder the question, the more times it has to be done. However, because AGS provides an editor to fill the code directly, students who do not pass a test will often change a little code

Table II
Summary of Formulas

Notation	Name	Range of values	Properties
F_0	Average score	[0-1]	The bigger the easier
F_1	Passed students	[0-1]	The bigger the easier
F_2	Passed submissions	[0-1]	The bigger the easier
F_3	Best submission	[0-1]	The bigger the easier
F_4	Number of submissions	[0-1]	The bigger the harder

right on the system and submit their code without checking carefully on the IDE. This approach increases the number of submissions but does not help improve student skills. The AGS system can provide and fulfill the following conditions:

- The number of tests is limited for learners to try and carefully work on each submission. Careful work helps the data reflect the student's work effort.
- The number of questions is large enough that learners switch to another question when one question is finished.
- The time to open the question is not too much for learners to spend time doing different questions, not too much time left to do many passes for one question.

The proposed formula is

$$F_4 = \frac{1}{N}\sum_{i=1}^{N}\frac{U_i}{U_{\max}}, \tag{5}$$

where N is the number of students, U_i is the number of submissions of ith student, $U_{\max}$ is the maximum number of submissions for the question.

3.2.6 Comments about difficulty-related formulas: Table II summarizes the five formulas introduced above with their range of values and properties.

To this study's concern, each formula F_1, F_2, F_3, and F_4 provides various aspects regarding the difficulty. Of all these perspectives, it can be seen that 3 main factors that affect difficulty are students, number of submissions, and grades. Furthermore, each formulation may have more than one contributing factor with varying degrees. Table III describes the above 3 factors with 3 contributing degrees 0, 1, 2.

- If a factor is not shown in the formula, its contributing degree is 0.
- If the formula is relevant to the factor, the factor contribution is at degree 1.
- If a factor is an inseparable part of the formula, its contributing degree is 2.

Table III also shows that each formula has a different combination of factors' contributing degree. This study aims to recognize difficulty on many different aspects, so our research model uses a feature vector $\langle F_1, F_2, F_3, F_4\rangle$ with four corresponding values of formula F_1 to F_4 to describe the difficulty for that question.

Table III
Factors that Affect Difficulty of a Practical Programming Question

Formula	Student	Submission	Score
F_1	2	0	1
F_2	0	2	1
F_3	1	0	2
F_4	1	2	0

3.3 Clustering Approach

Clustering is a technique of grouping similar data without being affected by a specific purpose other than data points themselves. K-mean is the well-known clustering technique published as a journal article in [12]. The algorithm performs the following steps:

- Select K cluster centers at random.
- For each data point, calculate the distance to each center and assign that point to cluster with the nearest center.
- Recalculate the new center for each cluster by calculating the new average point in each group.
- If the stopping condition is satisfied, then stop. Otherwise, repeat step (2).

The stopping condition may be the maximum number of iterations reached, or the displacement of the centers between two adjacent iterations is lower than a defined threshold.

This study does not focus on comparing and selecting the better clustering methods. K-means is appropriate to metric, easy to capture the structure of data and guarantee the convergence. Moreover, due to its frequent occurrence in categorizing questions [5], [7], we choose k-means as the clustering technique in this research.

After clustering, we choose Silhouette Score to measure the goodness of the result. The Silhouette Score is calculated as

$$\frac{b-a}{\max(a,b)},$$

where a is the mean distance from a sample to the other samples in the same cluster, b is the distance between a sample and the nearest cluster that the sample is not a part of [14]. The range of the Silhouette Score is between -1 to 1. The sample is considered to have been assigned to the correct cluster if close to 1. Whereas the value -1 implies that a sample has been assigned to the wrong cluster.

3.4 Coverage Approach

The authors in [13] give some directions to do coverage in terms of content and type of questions. However, in this study, the question content and the question type are both writing code; the concept (we call the topic) of a question is simply the topic it belongs to. Taking another direction, the main focus of our study is the difficulty degree of questions. Therefore, this study statistics the number of questions according to each of the difficulty degrees for each programming topic.

4 Question Difficulties Classification

This section describes the experiments and evaluation for question difficulties classification. *Pandas* tool [15] helps manipulate tabular data, which is used for preprocessing and calculating formulas' values for each question. *Scikit-learn* [16] package is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. Our study used k-means algorithm from *Scikit-learn* to cluster question's difficulty.

4.1 Methodology

This study proposes two clustering models to categorize difficulty degree of programming questions into three degrees of easy, moderate and difficult. The clustering results of two models then will be statistics for evaluation in each programming topic.

The experiment dataset is two submission results of two laboratory courses: FP (Fundamentals of Programming) and DSA (Data Structures & Algorithms). These data are obtained from AGS exercise data in semester 2, academic year 2019-2020. Table IV gives more information on the dataset.

4.2 Clustering Models for Determining Question's Difficulty

This study proposes two clustering models:

1) Model 1: use k-mean of scikit-learn with the number of clusters is 3, other parameters are let as default. The information for a training point is a value from formula F_0.
2) Model 2: use k-mean of scikit-learn with the number of clusters is 3, other parameters are let as default. The information for a training point is a vector of 4 values $\langle F_1, F_2, F_3, F_4 \rangle$.

4.3 Clustering Results

The Silhouette Score of each model is relatively good (Table V).

The clustering results of two models of two courses are presented in Table VI and Table VII. Each degree of easy, moderate, and difficult is marked respectively in tables as E, M, D.

4.4 Method to Assign Degrees to Clusters

In model 1, after clustering, we find the center of each cluster represented by a scalar. According to the properties of average score in Table II: the biggest center corresponds to the easy cluster, the smallest center corresponds to the difficulty cluster, and the last center corresponds to the moderate one.

In model 2, we also find these three centers, but with a four values vector interprets each one, it is impossible to determine the smaller and greater relationship between these two vectors. We define variable *score* as the score of a permutation of these three vectors and proceed with the following steps:

1) Generate six permutations of three 4-dimensional vectors.
2) For every two adjacent vectors, consider all pairs of values belonging to the same formula; if these two values satisfy the properties in the Table II of that formula, then increase *score* by 1, assuming you need to sort these vectors with increasing difficulty.
3) Choose the permutation with the highest *score*, assign the clusters of the centers of this permutation with easy, medium, and difficult degrees, respectively.

The assignment manner for model 2 can fail if there is more than one permutation with the same highest score, then we don't know which permutation to assign the cluster. With experiment data, there is only one permutation with the highest score and cluster assignment is achieved.

Table IV
Dataset Information

	Number of questions	Number of students	Number of submissions	Number of topics
FP	32	673	38543	8
DSA	32	58	2604	7

Table V
Silhouette Score of Models

	Model 1	Model 2
FP	0.69	0.66
DSA	0.78	0.61

4.5 Evaluation of Clustering Result

In course FP, we can see that there are 7 questions with different results between two models: 6, 14, 17, 23, 24, 26, 28. Questions 6, 14 are clustered by model 2 as more complex than model 1. The remaining questions are clustered by model 2 as easier than model 1.

In course DSA, we can see that there are 6 questions with different results between two models: 6, 7, 8, 10, 29, 32. All of them are clustered by model 2 as easier than model 1.

There is no question in both FP and DSA that the two models give conflict results.

Generally, model 2 tends to rank the degree as easier than model 1. We can see it more evident in DSA, maybe because DSA is a more challenging course than FP. Students may not get high marks on the first few tries and even got 0 points, affecting the mean score to get lower. Nevertheless, after thinking and trying the test, maybe students will eventually pass the test with the highest score. At that point, the student's mean score will not be high, but other aspects such as the highest score, the number of submissions, and the number of accomplished students may give more

Table VI
Clustering Result of FP

Question	1	2	3	4	5	**6**	**7**	8	9	10	11	12	13	**14**	15	16	**17**	18	19	20	21	22	**23**	**24**	25	**26**	27	**28**	29	30	31	32
Model 1	D	M	M	M	E	M	D	M	M	M	E	E	M	E	M	M	M	M	E	E	E	E	M	M	E	M	E	M	M	M	M	E
Model 2	D	M	M	M	E	E	D	M	M	M	E	E	M	M	M	M	E	M	E	E	E	E	E	E	E	D	E	E	M	M	M	E

Table VII
Clustering Result of DSA

Question	1	2	3	4	5	**6**	**7**	**8**	9	**10**	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	**29**	30	31	**32**
Model 1	M	E	M	D	D	M	M	D	M	M	M	M	D	M	M	E	M	M	M	M	E	M	E	E	M	M	M	M	M	E	E	M
Model 2	M	E	M	D	D	E	E	M	M	E	M	M	D	M	M	E	M	M	M	M	E	M	E	E	M	M	M	M	E	E	E	E

Table VIII
Formula Value for Question 7 and 8 of DSA

Question	F_0	F_1	F_2	F_3	F_4
6	0.70	0.94	0.51	0.963	0.36
7	0.71	0.96	0.50	0.996	0.35

Table IX
The Sameness and Similarity of Classification Results

	The sameness	The similarity
FP	78.13%	100%
DSA	81.25%	100%

Table X
Statistics of Degree in Course FP

		Model 1			**Model 2**		
Index	**Topic**	**Es**	**Ms**	**Ds**	**Es**	**Ms**	**Ds**
1	Loop & If ... Else	0	2	1	0	2	1
2	Array	1	2	1	2	1	1
3	String	0	3	0	0	3	0
4	Function and parameter passing	4	1	0	5	0	0
5	Recursion	3	2	0	2	3	0
6	Pointer	0	3	0	1	2	0
7	Link list	2	2	0	3	0	1
8	OOP	1	4	0	2	3	0

information to have a better perspective about the question.

Table VIII shows the calculated values of 5 formulas corresponding to questions 6 and 7 in DSA. According to *Average score*, we see the average score is about 7/10 points and ranked by model 1 as moderate. However, the rate of passed students is about 95%, the highest average score is 9.6/10 with question 6 and nearly 10/10 for question 7. The submission rate is also relatively low; if students have a maximum of 5 submissions, they only need 2 times to pass that question. Also, *Passed submissions* has a reasonable value: in two submissions, the first one is unsuccessful and the second one succeeds, so the rate of submission is about 50%.

Although there is no clear conclusion about difficulty, questions with different results as above should be considered further with our four proposed formulas in addition to the *Average score*. They are assisting teachers in doing reviews and deciding appropriate actions.

We propose two measures for evaluating the clustering result:

- **The sameness**: the percentage of the number of questions that two models have the same degree.
- **The similarity**: the percentage of the number for which the degree from two models is similar. Two degrees are considered similar if they are the same or less likely close but not contradictory.

Table IX records two above measures of courses FP and DSA. We can see that the results of model 2 have high matching with one of model 1.

5 Analysis of Programming Topic's Coverage

5.1 Statistics of Degree for Each Programming Topic

To observe the programming topic's coverage, we use classification results from 2 models and does statistics on the number of questions of each difficulty degree. The statistical results are shown in Table X and Table XI. The columns *Es*, *Ms*, and *Ds*, respectively correspond to the number of easy, medium and difficult sentences in each topic.

5.2 Evaluation of Statistical Results

In Table X, 6 out of 8 topics with different difficulty coverage are topics 2, 4, 5, 6, 7, 8. Consider topic 7 (Linked list): according to model 1, the classification has 2 easy questions, 2 moderate questions; according to model 2, there are 3 easy questions and 1 difficult question. The coverage of model 1 is plausible if the instructor wants the learners to do basic exercises. Teachers want students to get used to Linked lists, since Linked lists are quite advanced topics for an introduction course. The coverage of model 2 shows students' suitability if the teachers want learners to challenge some difficult questions. Also, it offers some lack the average question.

In Table XI, 3 out of 7 topics with different difficulty coverage are topics 2, 3, 7. In topic 2, model 1 shows a lack of easy questions, while model 2 shows a lack of difficult questions. In topic 7, model 1 shows a lack

Table XI
Statistics of Degree in Course DSA

		Model 1			Model 2		
Index	**Topic**	**Es**	**Ms**	**Ds**	**Es**	**Ms**	**Ds**
1	C/C++ Review, Recursion by C/C++	1	2	2	1	2	2
2	Implement AList Class with Template Programming	**0**	3	1	2	2	**0**
3	Implement singly linked list and its applications	1	5	1	2	4	1
4	Implement Stack and Queue with its applications	1	4	0	1	4	0
5	Implement Binary Tree and Binary Search Tree	2	3	0	2	3	0
6	Implement AVL and its applications	0	2	0	0	2	0
7	Implement heap, hash and their applications	2	2	0	4	**0**	0

of difficult questions, while model 2 shows a lack of moderate and difficult questions.

Therefore, two models can give different coverage for a topic. Although there is no clear conclusion as to which model is better in the above coverage, two observations can be made as follows:

- Firstly, for topics with similar coverage in two models, which states that the coverage has a high consensus, we can consider that coverage is reliable for representing the difficulty degree of the topic.
- Secondly, if the number of questions on the same difficulty degree in the two models is the same and equal to zero, it shows the lack of questions in this difficulty.

The instructor can rely on the above two observations to:

- Observe coverage if there is a similar coverage.
- Adding more questions to the difficulty degree of a topic where the number of questions is 0 in both models.

6 Conclusion

In this study, we aim to fulfill the difficulty degrees coverage of practical programming questions. We have proposed formulas related to the average score, passed students, passed submissions, best submission and the number of submissions to determine the difficulty of each question. Having proposed formulas, we use clustering techniques to group questions into three groups. The results will be an opportunity to look at the coverage of the topic-by-topic practice exercise set. Moreover, the results are directed to suggest the instructor to supply the questions with the missing difficulty degree according to each topic. These results are analyzed not based on the subjective opinions of the instructor but by observing the information set that stores not only the results of the learners but also the whole process of the work to submit.

The proposed solution has been measured across two subjects: Fundamentals of Programming and Data Structures & Algorithms. Data collected from these subjects through the second semester of 2019 at Ho Chi Minh City University of Technology. In view, the results of the study are as follows:

- Classify questions difficulty based on 2 models: one uses the formula using the average score formula and the other uses 4 formulas that we proposed. With the results, although there are some questions with different degrees of difficulty, in general, the two models give similar classification results, and they are reasonable.
- Conduct difficulty degree statistics on each programming topic. The results showed that there are differences in the difficulty degree coverage between model 1 and model 2. The study proposes to produce both classification results, along with the value of 4 proposed formulas and 1 average score formula. From there, the instructor will have many different perspectives to make decisions to add the necessary questions to the topic.

In the future, we still focus on these directions for this study:

- Enrich the dataset to enhance the accuracy of clustering models.
- Widen the scope of the fulfilling question difficulty coverage problem. This study is only focused on fulfilling the degree to which its amount of questions is zero. However, with a more significant number of questions due to enrichment, the zero value mostly will not be appeared. Therefore, the fulfilling method will not depend on the zero value and need a threshold-based configuration.
- Utilize the classification results for recommending the question with suitable difficulty for students. Furthermore, the next stage is the system to suggest programming practicing path for student.
- Expand the degrees of question difficulty, determine a number of degrees for having sensible jumps between degrees.

Acknowledgment

This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number DS2022-20-07. We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for this study.

References

[1] C. Coman, L. Țîru, L. Mesesan Schmitz, C. Stanciu, and M. Bularca, "Online teaching and learning in higher education during the coronavirus pandemic: Students' perspective," vol. 12, no. 2020: 10367, pp. 1–24, 2020.
[2] Ł. Tomczyk, K. Potyrała, A. Włoch, J. Wnek-Gozdek, and N. Demeshkant, "Evaluation of the functionality of a new e-learning platform vs. previous experiences in e-learning and the self-assessment of own digital literacy," vol. 12, no. 2020: 10219, pp. 1–22, 2020.

[3] G. van de Watering and J. van der Rijt, "Teachers' and students' perceptions of assessments: A review and a study into the ability and accuracy of estimating the difficulty levels of assessment items," *Educational Research Review*, vol. 1, no. 2, pp. 133–147, 2006.

[4] B. Simon, D. D'Souza, J. Sheard, J. Harland, A. Carbone, and M.-J. Laakso, "Can computing academics assess the difficulty of programming examination questions?" in *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, Nov. 2012, pp. 160–163.

[5] V. P. Mahatme and K. K. Bhoyar, "Questions categorization in e-learning environment using data mining technique," *International Journal of Information, Control and Computer Sciences*, vol. 9.0, no. 1, 1 2016.

[6] H. Chen and P. A. Ward, "Predicting student performance using data from an auto-grading system," *arXiv preprint arXiv:2102.01270*, 2021.

[7] S. Vamsi, V. Balamurali, K. S. Teja, and P. Mallela, "Classifying difficulty levels of programming questions on hackerrank," in *Proceedings of the Advances in Decision Sciences, Image Processing, Security and Computer Vision*. Springer, 2020, pp. 301–308.

[8] K. A. S. Awat and M. A. Ballera, "Applying k-means clustering on questionnaires item bank to improve students' academic performance," in *Proceedings of the IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, 2018, pp. 1–6.

[9] I. Chowdhury and Y. Watanobe, "Cluster analysis to estimate the difficulty of programming problems," in *Proceedings of the 3rd International Conference on Applications in Information Technology*, Nov. 2018, pp. 23–28.

[10] E. Verdú, L. Regueras, M. Verdú, and J. P. De Castro, "Estimating the difficulty level of the challenges proposed in a competitive e-learning environment," in *Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, 2010, pp. 225–234.

[11] E. V. Pérez, L. M. R. Santos, M. J. V. Pérez, J. P. de Castro Fernández, and R. G. Martín, "Automatic classification of question difficulty level: Teachers' estimation vs. students' perception," in *Proceedings of the Frontiers in Education Conference*, 2012, pp. 1–5.

[12] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[13] A. Petersen, M. Craig, and D. Zingaro, "Reviewing cs1 exam question content," in *Proceedings of the 42nd ACM technical symposium on Computer science education*, 2011, pp. 631–636.

[14] "sklearn.metrics.silhouette_score," last access: 19h30 28/05/2021. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

[15] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2021.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Huy Tran is an undergraduate student and pursuing his Master's Degree in Computer Science at Ho Chi Minh City University of Technology (HCMUT), VNU-HCM, Vietnam. His research interests are educational analysis and adaptive learning system.

Tien Vu Van is going to graduate with a Bachelor degree in Computer Science and continue to pursue a Master Program in Computer Science at Ho Chi Minh City University of Technology (HCMUT), VNU-HCM, Vietnam. His current research interests are machine learning, data analysis, and educational system.

Hoang Nguyen Viet is currently undergraduate at Computer Science from Ho Chi Minh City University of Technology (HCMUT), Vietnam. His research interests are software architecture for microservices-based system, workflow processing between services for automation in industry.

Duy Tran Ngoc Bao is a lecturer in Faculty of Computer Science and Engineering in Ho Chi Minh City University of Technology (HCMUT), VNU-HCM, Vietnam. He obtained an M.Eng in Computer Science (HCMUT, VNU-HCM) in 2020. His current interests are programming languages, compiler, code generation and applying the power of machine learning and deep learning in these relevant fields. He has also worked for some national projects in computer vision and data science.

Thinh Tien Nguyen is a lecturer and a researcher at the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT). He successfully defended his Ph.D. in Mathematics at Gran Sasso Science Institute (Italy) in 2018. At the moment, he is interested in physics-informed learning and applications.

Thanh Van Le received her Master degree in University Paris 8 in 2004 and achieved the Ph.D. diplomat in Computer Science from University Lyon 1, France in 2008. Since 2011, she has been a senior lecturer at the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Vietnam. Her current research interests include machine learning, data mining, big data and AI applications for public transport, image analysis, health care system, educational analytic, and blockchain mining.