International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 4, August 2019, pp. 3262~3271
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i4.pp3262-3271

Predicting cognitive load in acquisition of programming abilities

So Asai[1], Dinh Thi Dong Phuong[2], Fumiko Harada[3], Hiromitsu Shimakawa[4]
[1, 4]Ritsumeikan University, Japan
[2]Paracel Technology Solutions Co., Ltd, Vietnam
[3]Connect Dot Ltd, Japan

Article Info

Article history:

Received Mar 2, 2019
Revised Mar 29, 2019
Accepted Apr 8, 2019

Keywords:

Data mining
e-learning
Machine learning
Programming learning

ABSTRACT

In this paper, we propose a method to predict cognitive load and its factors affecting the learning efficiency in programming learning from the learning behavior of learners. Generally, since the concepts of programming are difficult for learners, some of them suffer inappropriate cognitive load to understand them. Although teachers must keep cognitive load of such learners appropriate, it is difficult for them to find learners who has inappropriate cognitive load from a large number of learners. To find learners with inappropriate cognitive load, we construct models with the random forest algorithm, using learning behavior collected from learners solving fill-in-the-blank tests. An experiment shows the models can detect cognitive load for IL and GL along with their factors. Teachers must address adjustment of cognitive load of learners. This result clarifies the learning factors affecting cognitive load of learners, which enables teachers to address the adjustment with small burdens.

Copyright © 2019 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

So Asai,
Ritsumeikan University,
National Chung Cheng University,
Nojihigashi 1-1-1, Kusatsu, Shiga, 525-8577, Japan.
Email: asai@de.is.ritsumei.ac.jp

1. INTRODUCTION

Educational institutes to teach information technology provide programming exercise classes for many novice learners. Novice learners must understand many abstract concepts to acquire programming abilities. It is difficult for novice learners because they have never experienced how the concepts are realized on computers. There are not a few learners to drop out [1].

The reason why learners cannot understand the specific concepts of programming is that inappropriate cognitive load is imposed on them [2]. Cognitive load is one of the important elements to consider in order to promote the learners to understand the specific concepts. Cognitive load is closely involved in acquisition and fixing of their programming skills. Learners are more likely to acquire programming abilities if they have appropriate cognitive load. It is necessary to pay attention so that learners do not have inappropriate cognitive load. However, each learner has its own way to have cognitive load. Usually, in a usual exercise class, one or a few education stuff teach more than decades of learners. It is difficult for teachers to find learners who have inappropriate cognitive load.

To provide learners with a preferable learning environment, many methods have been proposed for instructional design [3]. Keller [4] values motivation for such environments.The work in [5] lists motivations and strategies in learning. Phuong [6] regards them as factors determining the learning behavior of each learner. Phuong proposes a data analysis method to figure out the factors, to determine which students should be supervised.However, almost all of the instructional design methods try to extract learner's mental factors

such as motivation to establish successful environments [7]. Even highly motivated learners need support from their teachers to overcome difficulties when they struggle with difficult learning tasks.A systematic method is necessary for teachers to detect learners with high cognitive load.

This paper proposes a method to predict factors to impose cognitive load on learners through analysis of learning behavior they show at solving programming assignments. There are several types of cognitive load [8]. The method leverages fill-in-the-blank test to determine what type of cognitive load learners have. To determine the type of cognitive load, the method generates a model of random forest analyzing learning behavior they take at finding correct answers to fill blanks in the program. The output of the model identifies the type of cognitive load on learners, while the importance of predictor variables indicates its factors. This clarifies the state and factors of cognitive load of each learner. Teachers can address adjustment of the excessive cognitive load of learners into an appropriate state with minimal effort.

2. COGNITIVE LOAD IN PROGRAMMING LEARNING

2.1. Requirements to obtain programming ability

In order to acquire the programming ability, it is essential to be able to read given programs and write appropriate programs, utilizing concepts specific to programming to be understood. To achieve it, learners are required to organize various kinds of knowledge on many concepts along with their usages. Many learners cannot solve programming assignments because of difficulties of abstract concepts and ways to utilize them [9]. Learners without enough understanding of the concepts and the ways do not know what program they should write when they engage in assignments. Even if learners grasp programming concepts and ways to utilize them, many of them cannot imagine actual behavior of given programs. Those learners cannot understand why the programs behave in specified ways. Such learners may fail to learn programming, which may cause them to escape from programming learning. Teachers must find learners who are likely to have understanding failures to prevent them from escaping. It is indispensable to identify what impedes learners to understand programming.

2.2. Cognitive load affecting learning

People use working memory when thinking something. The amount of working memory represents the capability for process abilities to think. People must put many elements on their working memories when they learn new things. Since there are individual differences in working memory, the allowable amount of learning varies. When the same elements are repeatedly processed in working memory, they are organized as a schema. Once elements become a schema, learners can utilize them without cognitive load, because the elements have been organized with its usage. The goal of learning is that learners get able to solve problems never seen before without effort by combining elements they have understood. In other words, learning means to construct a schema into which elements are organized along with their usage.

Cognitive load affects understanding failure caused by difficulties in programming concepts [2, 10]. Cognitive load indicates how much working memory is assigned to tasks to understand unknown items and to utilize acquired knowledge in solving assignments [11]. The cognitive load theory classifies usage of the working memory into the following three types [8, 12].

Intrinsic Load (IL): IL occurs due to the inherent difficulty of the assignment against abilities of learners. IL is imposed when learners engage in problem solving under an unknown item and ways of consideration using it. This load gets high, when learners feel excessive difficulty for the assignment, because of the small amount of their working memory.

Extraneous Load (EL): EL is caused by surrounding learning environments and brought by the poor quality of teaching materials and lectures provided for learners. Teachers are required to design the materials and the lectures so as to reduce this load.

Germane Load (GL): GL is related to the schematization of contents to be learned. The imposition of GL implies learners are in the process of organizing given learning contents as schemata. Learners are encouraged to experience this load [13].

If it is an ideal learning situation that learners continue to acquire new knowledge, it is desirable that cognitive load is high in GL, in which knowledge is being schematized, while it should be low in IL and EL [8]. This work refers to it as an appropriate state of the cognitive load. Teachers must endeavor to keep cognitive load of learners an appropriate state. However, every learner feels different difficulties for each of the given learning content. There are various learners who have different effects of lesson design of programming lectures/exercises conducted as mass classes. It is difficult to estimate the cognitive load for each learner. Even more, it is nearly impossible to judge it from the appearance of learners on the spot during the class.

2.3. Estimation of cognitive load

A number of researchers have engaged in works to measure cognitive load [10, 14, 15]. Several methods are proposed to measure cognitive load. Their usefulness is confirmed in various fields. Morrison et al. [16] proposed a method to measure cognitive load of learners in the programming classes, without sensors. He extended the method Leppink et al. [17] established for statistics classes to programming learning. The method provides learners with several question items. Each of the question items is correlated to figure out either of IL, EL or GL. For each question item, learners present their answers by 11-scale. The Leppink's method assesses cognitive load for the whole learning process in the class, not factors causing it. The Morrison's method does not clarify factors of cognitive load of each learner, either.

Dividing visual information and character information in a programming class, Yousoof et al. [18] proposed a method to measure cognitive load from its accumulation, to reduce it. This method mainly focuses on EL. It does not fully consider IL or GL, which come from the utilization of working memory by learners. It is necessary to clarify essential factors of their understanding failures, considering learning behavior which appears for each type of cognitive load. Fridman et al. [19] measured the cognitive load during driving vehicle without wearing any sensors. This work investigates the cognitive load in real time, feeding video data of eye movements to the deep learning. This study does not distinguish the three types of the cognitive load.

In programming learning, it is essential to discriminate IL, EL, and GL. Since IL and EL decrease learning efficiency, they should be low. Meanwhile, high GL is preferable, because it implies the learner is working on the schematization of learning items. It should be avoided to attach physical sensors to learners, to obtain cognitive load in programming learning. There are many learners in the class. It brings huge costs to attach sensors to all of them. Sensors may also influence learning efficiency inappropriately. In addition to the investigation of cognitive load without sensors, it is necessary to clarify what learning behavior distinguishes the three types of cognitive load of learners. Furthermore, their factors should be identified for teachers to make them appropriate. A contribution of our work is to identify factors of cognitive load, as well as to investigate the three types of the cognitive load without specific sensors.

3. PREDICTING FACTORS AFFECTING COGNITIVE LOAD

3.1. Method overview

Our work aims to estimate cognitive load from learning behaviors at solving programming assignments, to predict each type of cognitive load along with factors causing it. A method is proposed to predict the cognitive load along with its factors when learners study programming with a procedural language like C. Figure 1 illustrates the method overview. To train classification models, learning behavior is collected from learners answering fill-in-the-blank tests. The method lets the learners specify their cognitive load for each assignment with the questionnaires explained in [16]. It trains models of random forest with the learning behaviors and the cognitive load. The models of random forest extract correspondences between the learning behavior and the cognitive load of the learners. When the learning behaviors of a new learner are provided, the model predicts learner's cognitive load along with its factors. The method helps teachers to confirm whether the new learners in programming learning are under appropriate cognitive load. The teachers can also address learners with inappropriate cognitive load. They can take measures to adjust their cognitive load, taking its factors into account.

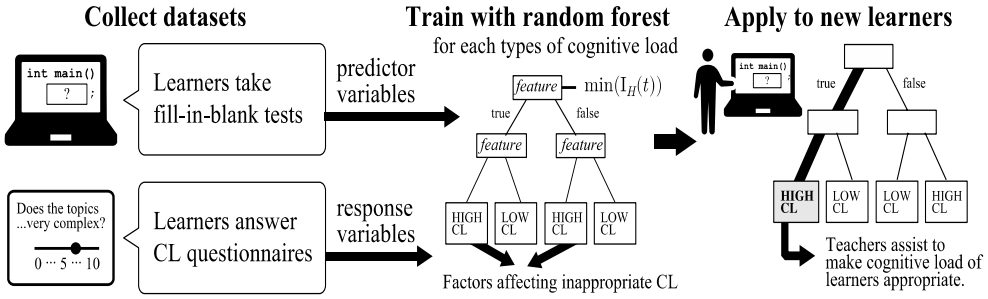

Figure 1. Overview of our method to predict factors affecting cognitive load

3.2. Collecting learning behavior

We focus on learning behavior when learners answer assignments of fill-in-the-blank test. Fill-in-the-blank test is frequently used to measure learner's understanding in programming classes [20]. Learners must fill code fragments suitable for blanks, considering coincidence with code fragments disclosed in other parts than blanks. Fill-in-the-blank tests reveal the understanding of learners because learners are requested to read the disclosed code fragment, understand them, and conceive code to fill the blanks. In fill-in-the-blank tests, it is less likely to speculate answers than in multiple-choice tests [21].

Fill-in-the-blank tests can examine learning achievements. Any types of cognitive load are not imposed on learners who have acquired programming abilities. Learners who are schematizing learning items have high GL because they are in the process of acquiring programming ability. High IL is imposed on learners when they engage assignments whose solution itself is hard to seek. Learners seem to have high EL by the assignments which bring unnecessary burdens such as sentences hard to read. Fill-in-blank tests, where the parts to be answered are limited, are useful for the measurement of the cognitive load as well as the understanding level of learners.

When answers which learners have convinced correct are judged to be incorrect, they consider the reasons, consuming their working memory. High IL occurs in this case. Recognizing their schema is wrong, learners reconstruct another schema. GL gets high in the reconstruction. Proper cognitive load of learners is collected, only if learners can receive the grading result on the spot when they solve fill-in-the-blank tests. In general learning with fill-in-the-blank tests, learners submit their answers on the sheet, with their grading results fed back after a few days. It is not expected proper cognitive load can be obtained with this learning procedure.

The method provides an automatic grading system [22]. It is implemented as a web application, with which learners can grade their answers interactively. The system grades an answer a learner gives for each blank on demand. It notifies the correctness of the answer immediately. Learners using the system are allowed to submit their answers many times until their answers become correct within the time limit.

In our method, one test corresponds to the code of a program, parts of which are blanked out. More than one tests are provided for learners. As learning behavior, the method collects 3 data items: the consuming time, change histories of answers, and the number of grading demands. More concretely, the system records the elapsed time from the time point a learner starts a specific test, a chronological list of answers the learner submits for each blank, and the number of grading demands transmitted to the system, respectively. We focus on 3 types of predictors explained below, to detect each type of cognitive load along with its factors. All of the predictors can be derived from the learning behavior the automatic grading system for fill-in-the-blank tests collect.

- Grading requests

Learners can check whether their answers are correct many times for each blank. High IL means learners have difficulties to fill blanks in the test because the test itself is hard to them. They have few candidate code fragments to fill the blanks. When none of them works well, the learners have nothing to do. Therefore, we assume that such learners demand to grade rarely. In the meantime, learners with high EL might fail to understand what the test requests or even how they should use the system, which leads them to do nothing. The system allows learners to demand to grade as many times as they want. It aims to cause learners to reach right answers after careful consideration. The method counts the number of grading for each blank. It does not count grading when the code fragment has not been modified from the previous demand, even if the learner demands to grade.

- Page transitions

In fill-in-blank tests, teachers provide multiple assignments in one session in order to confirm their understanding for various programming concepts. Learners can solve assignments in an order of their own choice. They can also switch them halfway. The learners change assignments to other ones when they either complete right answers for all blanks of an assignment or give up answering because they cannot imagine any other answer. The more difficulty learners perceive for assignments, the more they transit assignments. Trying to reconsider previous assignments, learners move back to them. In other case, learners move forward to new assignments. The method counts page transition, distinguishing the next assignment, the previous one, and one ahead more than two.

- Time transitions for correct answer rate

When a learner engages in a test, the learner repeats to send a grading demand after specifying an answer for each blank. Let the correct answer rate as the ratio of blanks they fill with correct answers against all the blanks. As a whole, learners increase their correct answer rate, as the number of grading request grows. Learners who have enough understanding of the tests can answer all of them easily. Their correct answer rate quickly reaches to the full mark or one close to it. On the other hand, learners who lack understanding need a long time to find correct answers or give up to find them. It is expected the time

transition of the correct answer rate plays a vital role in discriminating the cognitive load. In order to represent the role in an integrated way, the method quantifies the time transition T_m for assignment m by the following equation:

$$T_m = \frac{1}{2}\sum_{k=1}^{n}(p_k + p_{k-1})(t_k - t_{k-1}) + p_n(t_l - t_n),$$

where n is the number of grading demands of the learner, p_k is the correct answer rate at the kth grading, t_k is the elapsed time at the kth grading from t_0, and t_l is the deadline of answering. The time is excluded while learners answer assignments other than assignment m. $k = 0$ stands for the state at the start time of answering, where $t_0 = 0$ and $p_0 = 0$. The quantification enables us to represent the accumulation of the correct answer rate of a learner over the elapsed time, as Figure 2 shows. In the case that the correct answer rate reaches high early, T_m gets large as shown in Figure 2(a). On the other hand, when the correct answer rate of a learner remains low for a long answering time, T_m is small as shown in Figure 2(b).

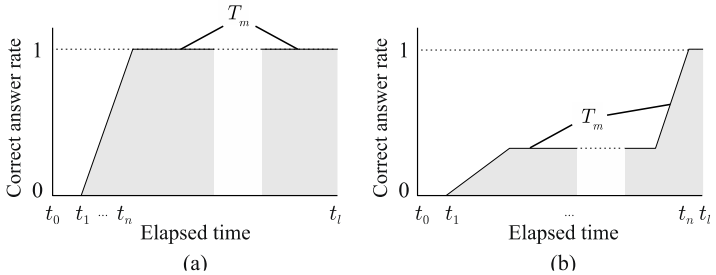

Figure 2. Examples of time transitions for correct answer rate

3.3. Investigating cognitive load

The method uses the cognitive load measurement questionnaire [16] in order to investigate three types of the cognitive load of learners. For the investigation, learners answer the questionnaire consisting of 10 questions. They are classified into 3, 3, and 4 questions corresponding to IL, EL, and GL, respectively. To clarify the target of each question, qualifiers are added to each statement of the cognitive load measurement questionnaire in the method. Figure 3 lists up the questions. Learners evaluate each of them by 11-scale. A larger number corresponds to a strong agreement for the question. When the sum of marks in questions corresponding to a specific type of cognitive load is large, its degree of the learner is judged to be high.

1. *The topics covered in the activity of solving fill-in-blank prob lems were very complex.*
2. *The activity of solving fill-in-blank problems covered program code that I perceived as very complex.*
3. *The activity of solving fill-in-blank problems covered concepts and definitions that I perceived as very complex.*
4. *The instructions and/or explanations of the website during the activity were very unclear.*
5. *The instructions and/or explanations of the website were, in terms of learning, very ineffective.*
6. *The instructions and/or explanations of the website were full of unclear language.*
7. *The activity of solving fill-in-blank problems really enhanced my understanding of the topics covered.*
8. *The activity of solving fill-in-blank problems really enhanced my knowledge and understanding of computing/programming.*
9. *The activity of solving fill-in-blank problems really enhanced my understanding of the program code covered.*
10. *The activity of solving fill-in-blank problems really enhanced my understanding of the concepts and definitions.*

Figure 3. Questions of cognitive load measurement questionnaire

3.4. Identifying cognitive load with random forest

The method generates models to associate characteristics of learning behavior with factors of the cognitive load evaluated by the questionnaire. The models found on the random forest algorithm. Predictor variables and response variables of the models are the learning behaviors and a specific type of cognitive load, respectively. Models based on the random forest algorithm present how important each predictor

variable is in detecting a target type of cognitive load, which contributes to identifying its factors. An individual model is generated for each of three types of cognitive load because our work aims to clarify factors for each of the three types of the cognitive load.

The random forest algorithm constructs a multitude of decision trees from randomly chosen predictor variables and produces a model classifying learners according to the degree of cognitive load by majority voting of the outputs of those decision trees. Each node of a decision tree composing the model bisects states of learners specified with the predictor variable, from the learning behavior in solving fill-in-the-blank test. It is desirable that one of the divided nodes includes more learners of a target type of cognitive load. Namely, the impurity in each node of the decision tree should be small. The impurity is represented by entropy $I_H(t)$, which is calculated with the following equation:

$$I_H(t) = -\sum_{i=1}^{2} P\,(i \mid t) \log_2 P(i \mid t)$$

The difference of the entropy after the branch from the one before the branch should be small, which corresponds to maximizing the information gain. The information gain $G(t)$ at the node t is obtained from the following equation:

$$G(t) = I_H(t) - \sum_{i=1}^{2} \frac{|t_i|}{|t|} I_H(t_i)$$

Each node is divided so as to minimize the information gain. To prevent the decision trees from overfitting, nodes whose information gain is less than a threshold value are not divided anymore, regarded as leaves. Learners matching the branching condition in their cognitive load are classified in each node. Eventually, learners with specific characteristics in learning behavior fall into each of leaf nodes. The characteristics correspond to a response variable. Combination of branching conditions along a path from the root to a leaf corresponding to high cognitive load reveals factors of learning behavior which affect the cognitive load.

In the analysis using random forest, a lot of decision trees are used to judge whether cognitive load is imposed on learners. In the determination of a specified type of the cognitive load, the more frequently a specific predictor variable is used overall decision trees, the more important the predictor variable gets. Models based on random forest present contributions of each predictor variable to the judgement of cognitive load as the variable importance.

3.5. Predicting cognitive load of learners

A model is constructed through training data collected from many learners solving fill-in-the-blank tests with the automatic grading system. New learners also solve fill-in-the-blank tests as the learners for the training did. Their learning behavior is applied to the trained models. Each cognitive load of the new learners is determined with majority votes by the models. Teachers are notified of detected type of the cognitive load and learning behaviors of the learner. When IL or EL is high or GL is low, the teacher should follow up the learners to lead their cognitive load to an appropriate state.

Under appropriate states of all types of cognitive load, learners can acquire programming skills more effectively. It contributes to preventing learners from failing programming learning, suppressing burdens of teachers. Teachers can utilize saved efforts to prepare better lectures to provide higher educational effects.

4. EXPERIMENT

4.1. Overview

An experiment was conducted to confirm whether the method identifies factors of cognitive load. The purpose of this experiment is the followings:

- Collecting datasets of learning behavior of learners answering fill-in-the-blank test
- Verifying models of random forest generated from the datasets

Subjects are Vietnamese college students who are learning programming in C and information technology. They are the second year college students. A preliminary survey confirmed they have already learned C programming for beginners. At the time of the experiment, there are various students in terms of interests and abilities toward C programming. The materials in the experiment were provided in English because the preliminary survey has confirmed most of them can understand English fairly well. In the

experiment, the subjects solved five assignments of fill-in-the-blank tests, where several code fragments are blanked out. Table 1 shows these assignments and the number of their blanks. Concepts on the learning units are generally difficult in programming learning [23, 24]. Assignments regarding the concepts are expected to reveal the difference in understanding of learners. The assignments questioning the concepts were chosen so that unbiased datasets can be obtained. The quality of the assignments is guaranteed because these assignments are actually used in programming classes at Ritsumeikan University.

The subjects use a website implemented the automatic grading system for fill-in-the-blank test described in Section 3.2. They access the experimental website with a browser that they usually use, and log in with user ID and password given in advance to solve assignments. They can solve assignments in any order. They can switch an assignment to another on the way within the time limit. Learning behavior of subjects is stored on the server with asynchronous communication of Web beacon [25] immediately every time learners take predefined actions such as pressing buttons and reloading Web pages. After the time limit has elapsed, they finish solving the assignments. They subsequently answer the two kinds of questionnaires. The one is for cognitive load measurement, and the other is for assessing the degree of difficulty for each assignment. They evaluate cognitive load in 11-scale, while difficulties for each the assignments in 5-scale.

Table 1. Assignments of fill-in-the-blank test in the experiment

Assignment No.	Learning units	No. of blanks
F1	2-dimensional array	3
F2	2-dimensional array	7
F3	Structure, pointer, and linked list	7
F4	2-dimensional array, and function	9
F5	Sorting algorithm	8

4.2. Result

Datasets of learning behavior are obtained from 54 subjects. 3 subjects are excluded due to the insufficient number of learning behavior. The proposed method constructed models of random forest [26] for the 3 kinds of cognitive load using the datasets. The model construction reveals important variables to detect IL, EL, and GL. The important variables [27] are determined based on Gini index [28]. Predictor and response variables for training with random forest are followings.

- Predictor variable: 43 features based on the 3 kinds of learning behavior described in Section 3.2
- Response variable: high or low of IL, EL, and GL obtained from the questionnaire

The datasets are divided into data for the model construction and the verification. In order to verify the accuracy of the decision tree, we adopted 6-fold cross-validation [29]. Table 2 shows the results of the accuracy which represents the correct rate of the verification data with the cross-validation. The predictor variables are arranged in descending order of the average of the importance. We choose the top 10 variables of them. The top 10 variables and their importance for each cognitive load are shown in Tables 3, 4 and 5. Variable importance means how much contribution to the model of random forest. The sum of them is 1. Table 6 indicates the difficulties for each assignment obtained by the questionnaire.

Table 2. Accuracies of the generated models

Cognitive Load	Training data	Test data
Intrinsic	0.907	0.740
Extraneous	0.870	0.537
Germane	0.944	0.870

Table 3. The top 10 important variables for IL

Variable	Importance
Grading requests for blank 4 in F4	0.074
Grading requests for blank 1 in F3	0.065
Time transitions for F3	0.059
Time transitions for F4	0.046
Grading requests for blank 3 in F4	0.045
Time transitions for F5	0.043
Time transitions for F1	0.039
Grading requests for blank 5 in F5	0.038
Grading requests for blank 5 in F4	0.038
Grading requests for blank 5 in F2	0.037

Table 4. The top 10 important variables for EL

Variable	Importance
Time transitions for F2	0.086
Page transitions to previous	0.077
Grading requests for blank 3 in F3	0.052
Time transitions for F3	0.050
Sum of page transitions	0.046
Grading requests for blank 5 in F5	0.042
Time transitions for F1	0.041
Grading requests for blank 1 in F3	0.037
Page transitions to next	0.037
Grading requests for blank 3 in F1	0.035

Table 5. The top 10 important variables for GL

Variable	Importance
Page transitions to previous	0.116
Grading requests for blank 1 in F2	0.093
Time transitions for F4	0.076
Grading requests for blank 3 in F1	0.070
Time transitions for F1	0.066
Time transitions for F2	0.049
Time transitions for F3	0.045
Grading requests for blank 3 in F4	0.043
Grading requests for blank 3 in F2	0.036
Page transitions ahead more	0.035

Table 6. Evaluation of difficulty level for the assignments

Assignment No.	Mean	Variance
F1	2.63	0.74
F2	3.14	0.74
F3	4.10	0.42
F4	3.87	0.58
F5	3.30	0.97

5. IMPORTANT PREDICTORS

This section assesses the usefulness of the model of random forest for cognitive load along with the influence of the important variables on the 3 types of cognitive load of the subjects. It also discusses measures teachers should address against inappropriate states of cognitive load. It can be said that variables important to distinguish subjects of high cognitive load from ones of low cognitive load take largely different values for the two kinds of subjects.

– Intrinsic load

Many of the important variables for IL are variables related to assignment F3 and F4. As shown in Table 6, the subjects evaluated F3 and F4 the most difficult among the five assignments. This result implies the difficulty of assignments strongly influences to detect IL. Let us check the code fragments blanked out in the assignments. The important variables are the number of grading requests for blanks, which should be filled with multiple statements or variables in value settings. Subjects must seek a correct answer for the blanks, considering the influence on other parts of the program. The top 10 variables include the time transitions for correct answer rate, which is explained in Figure 2, for the assignments except for F2. Since this learning behavior indicates how short subjects could have answered correctly, it is directly linked with the difficulty of the assignment.

These facts suggest that learners with high IL excessively consume working memory to IL because they engage in the assignments for a long time. On the other hand, learners with low IL can easily solve the assignment in a short time. Teachers can predict IL of learners, focusing on the learning behavior against assignments with high difficulty. Teachers can mitigate IL, providing assignments of low difficulty for learners with high IL.

– Germane load

The most important variable is the page transitions to the previous assignment. Transitioning to the previous assignment means that the subjects retry to solve the assignments. In our method, the subjects can solve any assignments of fill-in-the-blank test many times within the time limit. Because subjects have solved the assignments repeatedly, it seems they achieved to establish a schema related to the blank and the contents of the assignment.

The next important variables are related to assignment F1, F2, and F4. The contents of the assignments are related to linear algebra. They should be solved using 2-dimensional array. All of the subjects have obtained skills of linear algebra calculation before they learn programming. They have achieved each skill of programming and linear algebra calculation. Because they need to solve the assignments using both skills concurrently, the assignments seem to reveal the difference in schematization of programming knowledge.

On the other hand, predictor variables related to assignment F5 are smaller. Assignment F5 is solved with a sorting algorithm. Because the sorting algorithm is unknown for most of the subjects, they showed low GL in the assignment F5. Therefore, in order to detect GL on learners, it is effective to solve assignments utilizing knowledge learners have already obtained or assignments similar to the learning unit. In addition, learning behaviors predicting GL is important to confirm how many learners have benefited from the learning contents and supervision. In case that few learners have been benefited, teachers should review the learning contents and supervision, so that the learners can establish schemata for programming abilities.

– Extraneous load

The model of random forest in our experiment did not show sufficient accuracy for the testing data. This means the model cannot predict learners' EL from their learning behaviors in solving assignments of fill-in-the-blank test. It is difficult to figure out variables of learning behavior to detect EL because there are various kinds of variables in the 10 important variables. The cause of the result is considered that the subjects were unfamiliar with the use of the experimental website which implements the automatic grading system. The subjects have practiced with a simple assignment of fill-in-the-blank test to grasp usage of the website. However, it might not be enough. As another cause, it is also conceivable that a language used in the experiment was not the primary language for the subjects. Most of the subjects understand English. However, it is not everyday languages. Answering in non-primary language seems to affect EL.

In order to improve the accuracy of the model to detect EL, it is necessary to provide an environment where learners are familiar with learning environments. We should have the subjects to be accustomed to the website, letting them try the automatic grading system more times. It is also required to revise the website to be easy to use for the learners.

Our method detects cognitive load from learning behavior without attaching any sensors to learners. It clarifies cognitive load of learners at an early stage, avoiding extra burdens not only on teachers but also on learners. It is difficult to find out learners who have IL or GL inappropriately without our method. Teachers can address to make each cognitive load of learners appropriate on a light burden.

6. CONCLUSION

In this paper, we propose models to detect cognitive load along with its factors, founding on the random forest algorithm. We also discuss the usefulness of the models. The models are constructed with predictor variables representing three kinds of learning behavior. They are effective to detect IL and GL. The learning behavior during solving difficult assignments is useful to detect IL. It is effective in the detection of GL to analyze learning behavior of learners engaging in assignments they have already learned. On the other hand, EL can be detected if we improve the usability of the automatic grading system. Teachers can find learners who have inappropriate cognitive load early because the proposed method clarifies cognitive load of learners from learning behavior. In the future, we elaborately clarify the accuracy of learners' cognitive load and its factors, extending areas of learning contents.

REFERENCES

[1] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming," *SIGCSE Bull.*, vol. 39, no. 2, pp. 32-36, Jun. 2007.
[2] M. Okamoto and H. Kita, "A study of novices missteps in shakyo-style learning of computer programming," in *Memoirs of the center for educational research and training, shiga university 22*, pp. 49-53, 2014.
[3] R. A. Reiser and J. V. Dempsey, "Trends and issues in instructional design and technology," 3rd ed. New York: Peason, 2012.
[4] J. M. Keller, "Motivational design for learning and performance, the arcs model approach," New York: Springer, 2010.
[5] P. Pintrich, "A manual for the use of the motivated strategies for learning questionnaire (mslq)," Ann Arbor: National Center for Research to Improve Postsecondary Teaching; Learning, 1990.
[6] D. T. D. Phuong and H. Shimakawa, "Grasping motivation and strategy of current students referring to past programming course," *IEEJ Transactions on Fundamentals and Materials (A)*, vol. 136, no. 12, pp. 787-796, 2016.
[7] R. Gagné, W. Wager, K. Golas, and J. Keller, "Principles of instructional design," 5th ed. Belmont: Wadsworth Pub., 2005.
[8] J. Sweller, "Element interactivity and intrinsic, extraneous, and germane cognitive load," *Educational Psychology Review*, vol. 22, no. 2, pp. 123-138, 2010.
[9] S. Shuhidan, M. Hamilton, and D. D'Souza, "Understanding novice programmer difficulties via guided learning," in *Proceedings of the 16th annual joint conference on innovation and technology in computer science education*, pp. 213-217, 2011.
[10] J. Sweller, P. Ayres, and S. Kalyuga, "Cognitive load theory," *Springer*, 2011.
[11] W. Schnotz and C. Kürschner, "A reconsideration of cognitive load theory," *Educational Psychology Review*, vol. 19, no. 4, pp. 469-508, Dec. 2007.
[12] K. E. DeLeeuw and R. E. Mayer, "A comparison of three measures of cognitive load: Evidence for separable measures of intrinsic, extraneous, and germane load," *Journal of Educational Psychology*, vol. 100, pp. 223-234, Feb. 2008.
[13] J. Sweller, J. van Merrienboer, and F. Paas, "Cognitive architecture and instructional design," *Educational Psychology Review*, vol. 10, no. 3, pp. 251-296, Sep. 1998.
[14] E. Haapalainen, S. Kim, J. F. Forlizzi, and A. K. Dey, "Psycho-physiological measures for assessing cognitive load," in *Proceedings of the 12th acm international conference on ubiquitous computing*, pp. 301-310, 2010.

[15] F. Paas, J. Tuovinen, H. Tabbers, and P. W. van Gerven, "Cognitive load measurement as a means to advance cognitive load theory," *Educational Psychologist*, vol. 38, no. 1, pp. 63-71, Jan. 2003.
[16] B. B. Morrison, B. Dorn, and M. Guzdial, "Measuring cognitive load in introductory cs: Adaptation of an instrument," in *ICER '14 proceedings of the tenth annual conference on international computing education research*, pp. 131-138, 2014.
[17] J. Leppink, F. Paas, C. P. M. Van der Vleuten, T. Van Gog, and J. J. G. Van Merriënboer, "Development of an instrument for measuring different types of cognitive load," *Behavior Research Methods*, vol. 45, no. 4, pp. 1058-1072, Dec. 2013.
[18] M. Yousoof, M. Sapiyan, and and Khaja Kamaluddin, "Measuring cognitive load-a solution to ease learning of programming," *World Academy of Science, Engineering and Technology International Journal of Computer and Systems Engineering*, vol. 1, no. 2, pp. 32-35, 2007.
[19] L. Fridman, B. Reimer, B. Mehler, and W. T. Freeman, "Cognitive load estimation in the wild," in *Proceedings of the 2018 chi conference on human factors in computing systems*, pp. 652:1-652:9, 2018.
[20] K. Chang, B. Chiao, S. Chen, and R. Hsiao, "A programming learning system for beginners – a completion strategy approach," *IEEE Transactions on Education*, vol. 43, no. 2, pp. 211-220, May 2000.
[21] R. Medawela, D. Ratnayake, W. Abeyasinghe, R. Jayasinghe, and K. Marambe, "Effectiveness of 'fill in the blanks' over multiple choice questions in assessing final year dental undergraduates," *Educación Médica*, vol. 19, no. 2, pp. 72-76, 2018.
[22] S. Asai and H. Shimakawa, "Automatic scoring system of fill-in-the-blank tests to measure programming skills," in *Proc. Of the 6th the international conference on information technology and its applications*, pp. 23-29, 2017.
[23] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," in *Proceedings of the 10th annual sigcse conference on innovation and technology in computer science education*, pp. 14-18, 2005.
[24] I. Milne and G. Rowe, "Difficulties in learning and teaching programming–Views of students and tutors," *Education and Information Technologies*, vol. 7, no. 1, pp. 55-66, Mar. 2002.
[25] J. C. Sipior, B. T. Ward, and R. A. Mendoza, "Online privacy concerns associated with cookies, flash cookies, and web beacons," *Journal of Internet Commerce*, vol. 10, no. 1, pp. 1-16, 2011.
[26] J. Han, M. Kamber, and J. Pei, *Data mining, concept and techniques*, 3rd ed. Waltham: Morgan Kaufmann, 2010.
[27] T. Hastie, R. Tibshirani, and J. Friedman, "The element of statistical learning: Data mining, inference, and prediction," 2nd ed. Springer, 2009.
[28] K. P. Murphy, "Machine learning, a probabilistic perspective," Cambridge: MIT Press, 2010.
[29] T. M. Mitchell, "Machine learning," New York: McGraw-Hill, 1997.