K-012

Investigations of Difficulty Level Update in Recommendation Function for Element Fill-in-Blank Problem in Java Programming Learning Assistant System

Su Sandy Wint ‡ Nobuo Funabiki‡‡ Minoru Kuribayashi‡‡‡
pfpo5v7r@s.okayama-u.ac.jp funabiki@okayama-u.ac.jp

Abstract

A Web-based ***Java Programming Learning Assistant System*** **(*JPLAS*)** offers the ***element fill-in-blank problem*** **(*EFP*)** for novice students to study Java grammar and basic programming skills. By applying the ***blank element selection algorithm***, a large number of EFP instances have been generated to cover various grammar topic. Previously, we proposed the ***recommendation function*** to suggest a proper EFP instance among them to be solved next for each student. It considers the ***difficulty level*** that is increased or decreased depending on the solving performance of the previous instance. In this paper, we investigate the proper difficulty level update in the function. The experiment results show that the *quick increase* and the *slow decrease* can be a good choice for both high-level and low-level students.

Keywords: Java programming, JPLAS, recommendation function, element-fill-in-blank problem

1 Introduction

Currently, *Java* has been widely used from large scale enterprise systems to small-size embedded systems. To support *Java programming* studies, we have developed a Web-based *Java programming Learning Assistant System (JPLAS)* [1]. JPLAS offers several types of programming exercises to cover various learning stages, starting from grammatical study until practical code writing using object-oriented concepts.

In this paper, we focus on *the element fill-in-blank problem (EFP)* [2], which is mainly designed to learn Java grammar through code reading. By applying the *blank element selection algorithm*, a large number of *EFP* instances have been generated to cover various grammar studies. Therefore, we proposed the *recommendation function* to suggest a proper *EFP* instance among them for each student to be solved next [3]. It considers the *difficulty level* that is increased or decreased depending on the solving performance of the previous instance.

In this paper, we investigate the effects of the parameter values for the difficulty level control, in solving performances of students through experiments. We applied the same set of EFP instances to 10 students in Okayama University with four different values. The results show that a high -level student reached the highest level without mistakes at any parameter value, and a low-level student actually decreased the level after he/she could not solve EFP instances at middle levels when a smaller value is used.

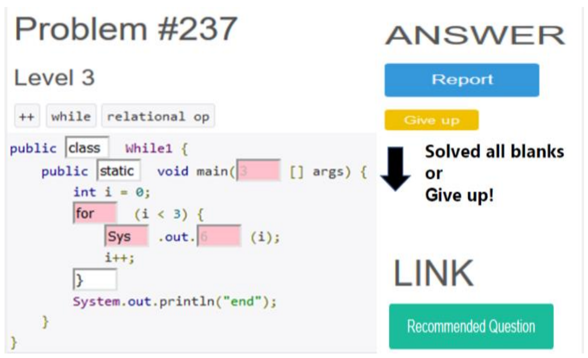
2 Review of EFP and Recommendation Function

In this section, we review the EFP answering interface and the recommendation function in JPLAS.

† Graduate School of Natural Science and Technology
† Okayama University, Okayama, Japan

2.1 Answering Interface for EFP

A student answers a selected EFP instance using the interface in Figure1. He/she needs to fill in the blank forms with the proper elements. When “Report” is clicked, the results are sent and marked at the server. When “Give up” is clicked, the correct answers of incorrect blanks are shown there. When “Recommended Question” is clicked after either of them, a proper EFP instance to be solved next is suggested.



```
public class   While1 {
    public static   void main(3     [] args) {
        int i = 0;
        for     (i < 3) {
            Sys   .out.6    (i);
            i++;
        }
        System.out.println("end");
    }
}
```

Fig.1 Answering interface for EFP instance

2.2 Difficulty Level Calculation

In the function, the difficulty level L is increased when all of the blanks in the previous instance are correctly answered. L is decreased when “Giving up” is clicked. The difficulty level change ΔL is controlled by the correct answer rate and the number of submissions as follows:

- If $x = 100$ and $y <= N$, $\Delta L = D$.
- If $x = 100$ and $y > N$, $\Delta L = \frac{D}{y-(N-1)}$
- If $x < M$, $\Delta L = D(\frac{x}{M} - 1)$
- Otherwise, $\Delta L = 0$.

Where x and y represent the correct answer rate (%) and the number of answer submissions at solving the previous EFP instance respectively. D does the maximum correct answer rate, which are given as parameters.

3 Investigation of Different Level Update Parameters

In this section, we investigate the effects of different level update parameters in the recommendation function.

3.1 Parameter Value Selections

The values of the three parameters D, N, and M are changed as in the four cases in Table 1.

Table 1 Parameter values for investigations

parameter	case1	case2	case3	case4
D	2	3	5	8
N	3	3	2	3
M	80	75	80	80

3.2 Application Results

We asked the students in Okayama University to solve EFP instances in JPLAS using the recommendation functions with the four-case parameter values. For case1-case3. 10 students actually solved them, and for case4, 48 students solved them. After that, we picked up typically results for a high-level student, a middle-level student and a low-level student for each case.

Figures 2-5 illustrate the difficulty level transition results for each case respectively. It was found that at any case, a *high-level student* reached the highest difficulty level in the shortest way, a *middle student* also reached the highest level but stayed at a certain level for a while, and a *low-level student* could not reach there. Actually, the last student decreased the level after he/she could not solve an EFP instance at the middle level, when a smaller value is used for *D*. It may happen since this student could not solve problems for certain grammar topics. The recommendation function continuously selects EFP instances covering such topics.

Thus, we conclude that for *D*, a larger value should be used at increasing the difficulty level, and a smaller value should be at decreasing the level. The quick increase can reduce time of solving EFP instances by a *high-level* student who has the sufficient programming ability and can skip solving EFP instances. The slow decrease can give more time for solving EFP instances to a *low-level* student who needs studying fundamental Java programming. However, the large *D* may skip the EFP instances covering important topics in Java programming to be studied [3], because it only selects a small number of instances that can cover limited ones. Thus, it is necessary to consider the keyword coverage at the EFP instance selection besides the difficulty level, which will be in future works.

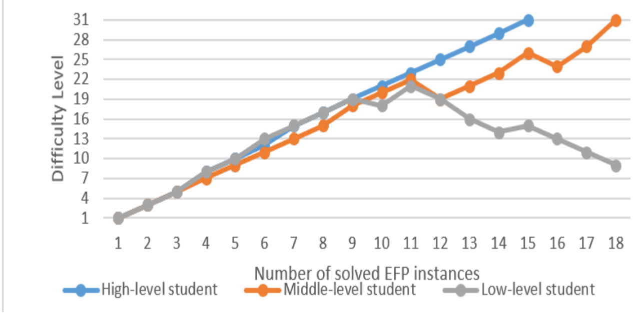


Fig.2 Level transition result for case1

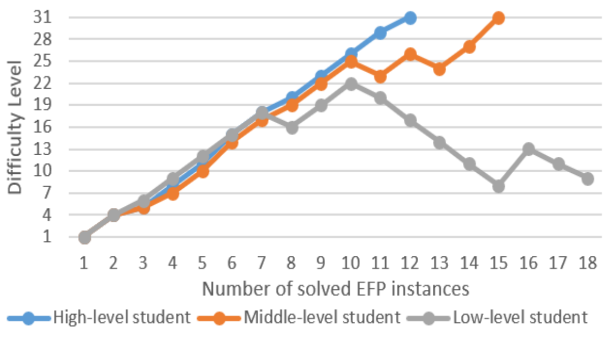


Fig.2 Level transition result for case2

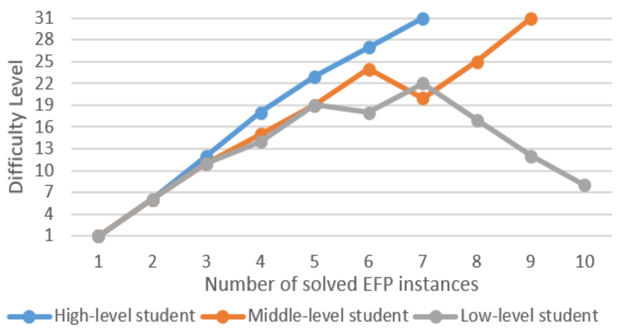


Fig.2 Level transition result for case3

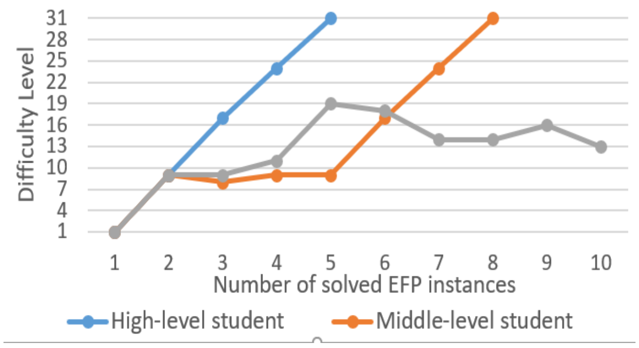


Fig.2 Level transition result for case4

4 Conclusion

This paper investigated the proper difficulty level update in the recommendation function in JPLAS. The experiment results showed that the *quick increase* and the *slow decrease* can be a good choice. Future works will be the topic coverage consideration at the EFP instance selection and their evaluations by a lot of students.

References

[1] S.Ao et al. edited, IAENG transactions on engineering sciences-special issue for the international association of engineers conferences 2016 (volume II), pp. 517-530, World Sci. Pub., 2018.

[2] N. Funabiki, Tana, K. K. Zaw, N. Ishihara, and W.-C. Kao,"A graph-based blank element selection algorithm for fill-in-blank problems in Java programming learning assistant system", IAENG Int. J.Comp. Sci., vol. 44, no. 2, pp. 247-260, May 2017.

[3] N. Funabiki, S. Matsumoto, S. S. Wint, M. Kuribayashi, and W.-C. Kao, "A proposal of recommendation function for solving element fill-in-blank problem in Java programming learning assistant system," to appear in Proc. NBiS-2019, Sep. 2019.