Liu, C.-C. et al. (Eds.) (2014). Proceedings of the 22nd International Conference on Computers in Education. Japan: Asia-Pacific Society for Computers in Education

Algorithmic Thinking Learning Support System in Block Programming Paradigm

Mizue KAYAMA[a*], Hisayoshi KUNIMUNE[a] , Masaaki NIIMURA[a], Masami HASHIMOTO[a] & Makoto OTANI[a]
[a]*Shinshu University, Japan*
*kayama@cs.shinshu-u.ac.jp

Abstract: The goal of this study is to develop a learning support system for algorithmic thinking for novices. This paper describes some educational functions of this system. The main features of this system are 1) the teacher can control the usage of control elements for developing algorithms based on his educational philosophy, 2) a simple task-answer management system is included and 3) eAssesment functions based on the Student-Problem table method are implemented.

Keywords: Algorithmic thinking, education for beginners, student-problem score table analysis, problem reviewing.

1. Introduction

An algorithm is defined as "... a set of rules that precisely defines a sequence of operations such that each rule is effective and definite and such that the sequence terminates in a finite time"[Stone 1972, Knuth 1980]. As such, until recently, algorithms have been taught in programming courses. However, since about five years ago, algorithms have been taught before programming. There have been many papers on this kind of algorithm and/or algorithmic thinking education, e.g., Sarawagi 2010, Hubalovsky 2010, Futschek 2011.

We have proposed an educational method for algorithmic thinking in fundamental education for computer science course students in 2008[Fuwa 2009] and examined the effectiveness of our algorithmic thinking learning support system. We designed our system based on discovery learning[Bruner 1960] and guided discovery[Cook 1991]. This system uses only three flow structure elements: calculation, selection and repetition. Students describe the algorithm using these three elements.

"What kinds of problems are suitable for assessing the students' ability and evaluating achievement level?" This is our research question. To review some typical exercise questions, we introduce the Student-Problem (SP) Score Table Analysis method. This paper describes educational functions in our algorithmic thinking system and our eAssessment function.

2. Breif Description of Our Learning System

2.1 Educational Features

We have developed a web based learning system for basic algorithmic thinking education[Kayama 2014a]. This system has the following pedagogical features:

1. The algorithm is represented not by a flow chart, but by columns of blocks such as stacks that grow downward.
2. Input errors can be reduced.
3. The rules to describe the algorithm can be increased in pedagogical stages.

Feature 1 makes learners concentrate on block combinations without thinking about notation. Each block corresponds to all the elements for constructing the algorithm. Learners can insert, move or delete the blocks with their mouse.

To realize feature 2, the variables only need to be typed once. After that, they can be selected from a list. Only operations that are allowed can be selected from a list.

Feature 3 allows control of the blocks visible in proportion to the learner's learning progress. This makes learners think about the algorithm in a situation with limited available blocks. On the other hand the blocks that have not been learned are invisible, which helps learners think about the algorithm without stress.

2.2 Learning Support Functions

2.2.1 Overview

The system with the features explained in the previous subsection was developed using the Web2pf framework. The combination of some elements represents the structure of the algorithm. Each process is described with a combination of comment statements and a selection of items.

Fig.1 shows an example screen shot of our algorithm editor for learners. In this system, a user chooses an element for use in the algorithm by double clicking on the editor area. A learner constructs an algorithm with blocks which were chosen. Executable parts are connected to the "start" block (upper left area in Fig.1). The sets of blocks that are not used can also be put in this editor area. They are shown in gray. A learner can watch the change in the variable values that he/she uses in his/her algorithm. Moreover, some output data can also be watched in this editor.

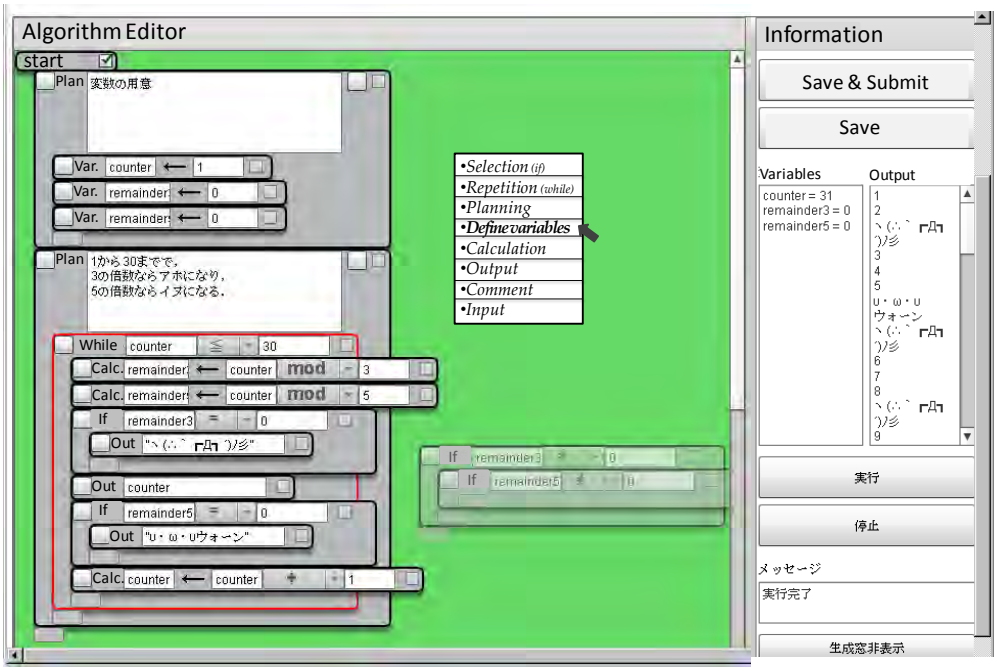


Figure 1. The algorithm editor of our tool.

There are two kinds of blocks, structural blocks and commentary blocks. The structural blocks are calculation, variable declaration, selection, repetition and break. The selection block and repetition block can be nested in other blocks. The commentary blocks are planning and comment. The planning block can be nested in other blocks including structural blocks.

Fig.2 shows the window where rules can be defined for describing the algorithm and the selectable items for the learners. A teacher can define the description rules for each algorithm question. He/she only needs to click the check boxes shown in Fig.2, and then saves this rule with a name. When he/she submits a question, one description rule can be attached.

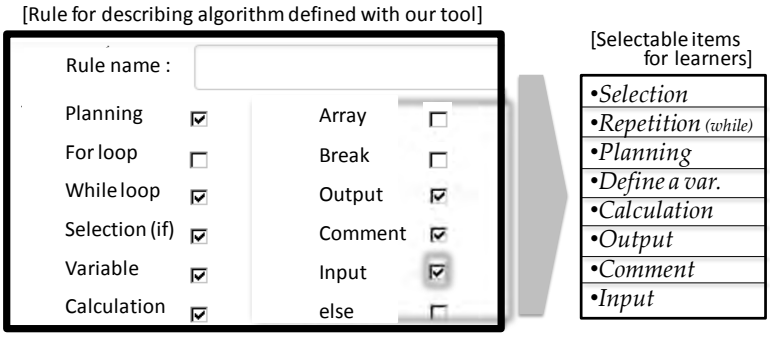


Figure 2. Definition of the rule for describing the algorithm.

The teacher usually plans the learning steps for his students. As an example, suppose that the first step is a calculation by assignment and/or arithmetic operations with abstract numbers and/or variables, the second step is a conditional test with relational operations and the third step is an iteration with termination conditions. Corresponding to these steps, the algorithm description rules can be defined. Fig.3 shows this set of learning steps and the description rules.

2.2.2 Element Usage of Control

Each teacher has his/ her own policy to teach algorithmic thinking. Each policy has to be appreciated because a teacher has a responsibility for his/her course. Therefore, by using our system, teachers can control which elements are visible according to the class rules. Different algorithm building elements are attached to the three steps in Fig.3. The rule set can be independently managed for each step. A teacher can set this rule before his/her lesson. This rule can be changed during the lesson.

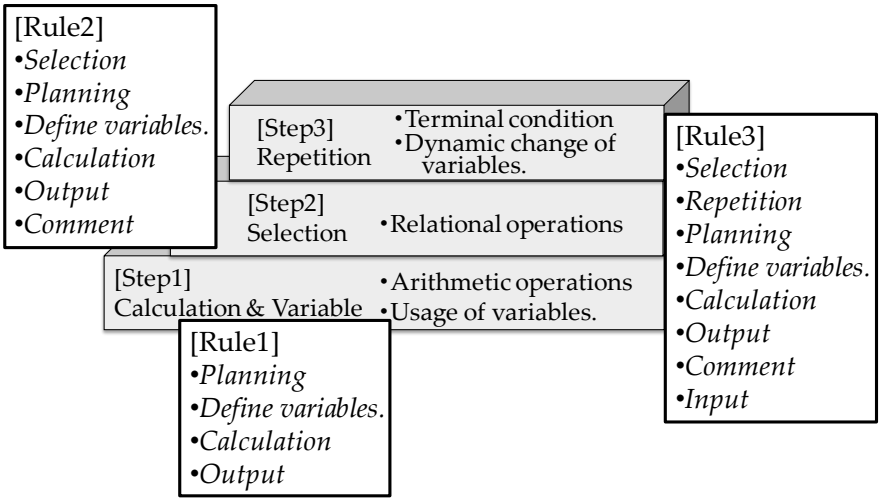


Figure 3. The learning steps and the algorithm description rules.

Moreover, a teacher can define which activities are permitted for constructing algorithms, i.e., permit to submit, save, execute, edit, syntax check, show program code, log operation history or not. In our system, these things are defined as controllable elements. By using this function, the teacher can define an algorithm reading task (not permit learners to edit and run), an algorithm reading and running task (not permit learners to run) and a scaffolding task (give a template algorithm using planning blocks with teacher's comments).

Teachers can also decide whether the trace function is available and its interval. Teachers can make the trace function available to debug the algorithm or to make it unavailable to make learners think about tasks and learning progress. If a teacher gives permission to use this function, learners can trace the execution of the algorithm and specify a variable when tracing. The right area in Fig.1 shows a variable trace area and the result of algorithm tracing (output area).

2.2.3 Template for each Task

This allows learners to see a part of the algorithm or the whole algorithm on their monitor. Therefore, combining this function with the learning activity control function, learners use this system to dissect the algorithm, finish incomplete parts of the algorithm, select a suitable algorithm from some choices, and describe the algorithm from scratch.

2.2.4 Simple Task-Answer Management System

a) State management of learners' reports

This system helps teachers and learners manage the report status for each learner and each task. For instance, saved but not submitted, submitted, rejected, accepted, unevaluated or evaluated. Students' answers are divided into two columns: unevaluated and evaluated. In the evaluated answers, a grade for each answer is shown.

b) Semi auto test for the submitted answers

If a teacher defines the test rules for a problem, our system tests students' answers automatically. There are three types of test results: 1) normal exit and the algorithm outputs the expected results (OK), 2) normal exit but the algorithm does not output the expected results (NG), 3) grammatical error or abnormal exit or time out (??). .

2.2.5 eAssesment functions based on Student-Problem Score Table Analysis

The student-problem (SP) score table analysis is an educational analysis method based on students' responses, i.e. test score patterns [Satoh 1974, Harnisch 1981]. The SP score table is a two-dimensional table where rows are student numbers and columns are problem (i.e., test question) numbers. In the table, if a particular student answers a particular problem correctly, the cell is filled with a 1. Otherwise, the cell is filled with a 0. This table is sorted by column and by row in decreasing order of occurrence of 1s. The students are sorted by their total score. If some students have the same score, they are ordered based on the sum of the number of correct answers to the problem which they answered correctly.

As a consequence, the upper-left triangular region is filled with nearly all 1s. Ideally, students with higher scores should solve those problems which are answered correctly by most students. Similarly, if a problem is solved by most of the students, a good student is able to solve the problem.

3. eAssessment with Our Tool

3.1 S Line and P Line

In an SP score table, there are two types of lines named the S line and the P line.

The S line is defined based on the scores of all students. Starting from the top student of the table, for each student, a short vertical line is drawn on the right of the score for the problem column which is the same as the student's score. The S line shows the frequency distribution of the students' scores. Ideally, there should only be 1s on the left side on this line and 0s on the right side. Teachers can capture the status of the achievement of each student.

The P line is defined based on the score of all problems. From the highest score problem, for each problem, a horizontal line is drawn below the score for the student row which is the same as the number of correct answers for the problem This line shows the frequency distribution of the problems' score. Ideally, only 1s should be above this line and 0s below it. Teachers can capture the suitableness of each problem or the effect of his/her instruction.

These two lines are ideally located close together in the table. The difference between the S line and the P line shows some pedagogical agenda to be solved.

1. The suitability of problems for this learner group.
2. The effectiveness of the teacher's instruction for this learner group.
3. The achievement level of each student.
4. The variation in learning abilities of the learners.

3.2 Caution Indices

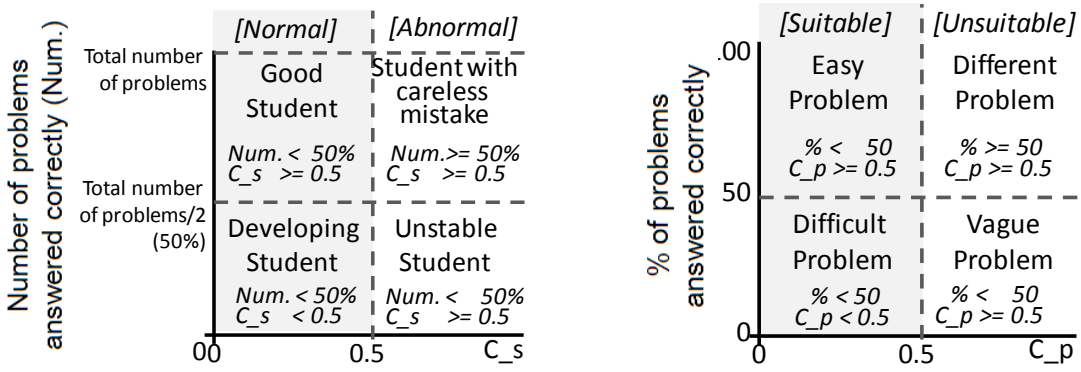
This method includes two index types: a caution index for students (C_s) and a caution index for problems (C_p) .

3.2.1 Caution Index for Student

By using the C_s index, we can easily detect the students who need attention. Theoretically, the threshold value is 0.5. If there are students whose values are under 0.5, they show abnormal response patterns for the problems, i.e., careless mistakes by high achieving learners and lucky guesses by low achieving learners. Moreover, teachers can diagnose students who cannot use the learning environment, especially the description method.

The combination of the C_s index and the percentage of questions answered correctly expresses the achievement level of students. We call this graph the Student Score Graph. Fig.4(a) shows our interpretation of the Student Score Graph. The vertical axis is the number of problems answered correctly (Num. for short) and the horizontal axis is the C_s index. This graph is separated into 4 sections by the vertical axis (Num.) threshold value of 50% of total problems and the horizontal axis (C_s) threshold value of 0.5. The left part of this graph is when the C_s index is less than 0.5. The students in this area seem to have normal reactions to the problems. If a student is in the upper left section (Num.>=50%), he/she is a "good" student. The lower left section (Num.<50%) is a "developing" student who needs more practice.

The right part is when the C_s index is more than 0.5. The students in this area seem to have abnormal reactions to the problems. Students in the upper right section, where Num. is high, tend to make careless mistakes for fundamental problems or to not fully acquire the related knowledge. When Num. is low and the C_s index is high, the comprehension of the students is quite uncertain. Students in the lower right section tend to have unstable answers. Therefore, they require special attention from the teacher. In our research, the C_s index has been used to identify students who need attention in each lesson.



(a) C_s (b) C_p

Figure 4. The relation of the Caution Indices and the number/percentage of the problems answered correctly

The right graph in Fig.5 is an example of the Student Score Graph. The radius of each circle represents the number of students located at that position. The center position of each circle shows the score combination. The five students mentioned above are located in the lower right section. The teacher may give these students particular attention.

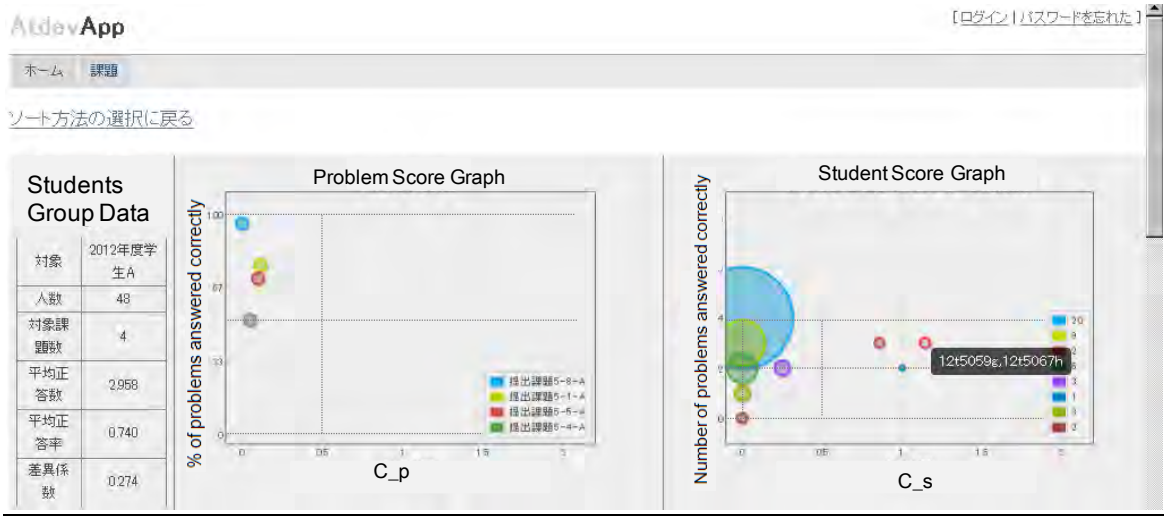


Figure 5. Problem score graph and Student score graph generated by proposed our tool.

3.2.2 Caution Index for problem

By using the C_p index, teachers can analyze the quality of a problem. We can detect inadequate questions which may include vague instructions or ill-structured tasks. Theoretically, the threshold value is 0.5. If C_p<0.5, teachers can modify or delete defective problems, and fit good ones into an item bank for future use. In Fig.5, there are no problems that need to be improved because their C_p Indices are all under 0.5.

The combination of the C_p index and the percentage of problems answered correctly expresses the appropriateness of the problem to evaluate the students' abilities. We call this graph the Problem Score Graph. Fig.4(b) shows our interpretation of the problem score graph. The vertical axis is the percentage of problems answered correctly (% in short) and the horizontal axis is the C_p index. This graph is separated into 4 sections by a % threshold value of 50% and a C_p threshold value of 0.5.

The left part of this graph is when the C_p index is less than 0.5. This means the problems in this area are "suitable" to evaluate students' abilities. If a problem is in the upper left section (%>=50), it is an "easy" problem. A problem in the lower left section (%<50) is a "difficult" problem which good students tend to answer incorrectly. The right part is when the C_p index is more than 0.5. This means the problems in this area are "unsuitable" problems to evaluate students' abilities. If a problem is in the upper right section, however, the % is high, so this problem includes some "different" factor to evaluate target abilities. A problem in the lower right section is a "vague" problem. The % is low and the C_p index is high. This means the problem includes ambiguity factors in its description. For example, the problem statement is difficult to read, or the way of answering the problem is not suitable for the learning environment. The C_p index has been used for evaluating the quality of each problem for each school year. The left graph in Fig.5 is a Problem Score Graph..

3.3 Practical Usage

We have been using this method to assess the problems in our course [Kayama 2014b]. In 2011, we made six groups of problems.

(A) Algorithm creation with guidance. A learner is given some variables with their initial values which can be used to create the algorithm, and guidance for creating the algorithm. He has to add appropriate blocks based on the given guidance.

(B) Algorithm creation from scratch. A learner has to add appropriate blocks without guidance. He is only given some variables with their initial values. If he does not want to use the given variables, he can define other variables.

(C) Algorithm creation by filling in blanks and adding explanations. A learner is given an incomplete algorithm with some blanks. He has to try to read the intention of the algorithm with the given blocks. After that, he adds some appropriate blocks to create the complete algorithm (C1). Then he is asked to explain the procedure by adding blocks (C2).

(D) Addition of the explanation for the procedures in the algorithm. A learner is given a complete algorithm with some "Plan" blocks. He is asked to explain each procedure in the blocks.

(E) Addition of the explanation for whole algorithm and the block sets in the algorithm. A learner is given a complete algorithm with some "Plan" blocks as in (D). He is asked to explain the whole algorithm (E1) and each procedure in the blocks (E2).

(F) Addition of the estimated value for the output and explanations of the block sets in the algorithm. A learner is given a complete algorithm with some "Plan" blocks as in (D) and (E). He is asked to explain the estimated value for the output (F1), and each procedure in the blocks (F2).

(A) and (B) are algorithm creation problems. On the other hand, (D), (E) and (F) are algorithm reading problems. Only (C) is categorized as both.

In each problem group, there is one calculation problem, two selection problems, two repetition problems and one combination problem, for a total of six problem types.

Therefore, we prepared 36 problems to analyze the appropriateness of the problems to evaluate the learners' abilities in algorithm creation and reading.

The subjects were 96 freshmen, who had finished our "algorithmic thinking" course. These students were divided into 6 groups. Each group was assigned six problems, one from each problem group and problem type. They had to answer all the problems in 90 minutes.

Table 1 shows the analysis result using our tool. In this table, each cell is marked in one of three categories: blank, difficult and different. Blank indicates that the problem is in the "Easy Problem" area in Fig.5(b). Difficult indicates that the problem is in the "Difficult Problem" area. Different indicates that the problem is in the "Different Problem" area. There are no "Vague Problems" in this test.

As for the algorithm reading problems, the "Different Problems" tend to be "explain the procedure" group problems (especially D, E2 and F2). As for the algorithm creation problems, (C1) asks students to fill in blanks in a given algorithm. This problem is the most difficult of the 6 problem groups. The percentage of students who answered this problem correctly shows a statistically significant difference between (B), which asks students to create an algorithm from scratch and (F), which asks students to estimate the output value from a given algorithm.

Table 1: The problem assessment with the SP score table analysis in 2011.

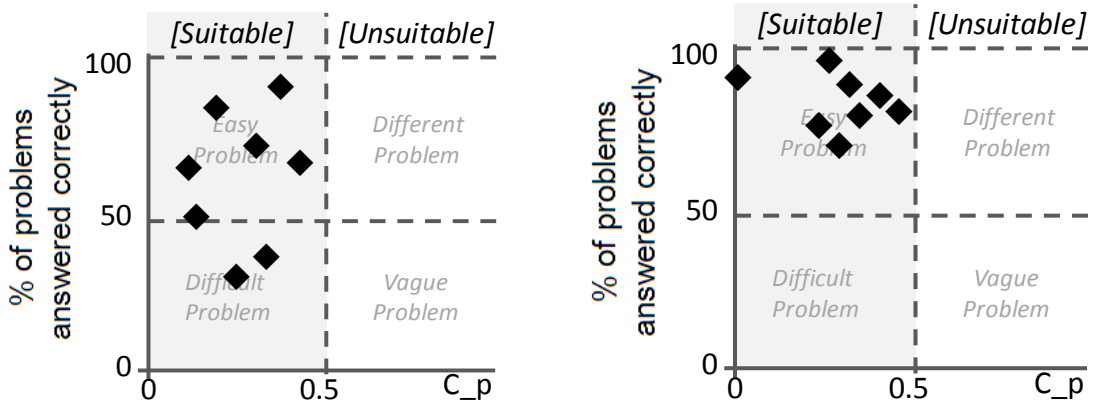
		Calculation	Selection1	Selection2	Repetition1	Repetition2	Combination
Algorithm CreationProblems	(A) Creation with guidance				Different		
	(B) Creation from scratch		*Difficult*				
	(C1) Creation by filling in blanks		*Difficult*		*Difficult*		*Difficult*
Algorithm Reading Problems	(C2) Addition of the explanations					*Difficult*	*Difficult*
	(D) Explaining the procedure in blocks			Different			
	(E1) Explaining the whole algorithm						
	(E2) Explaining the procedure in blocks	Different			Different		
	(F1) Explaining the estimated value for the output				Different		*Difficult*
	(F2) Explaining the procedure in blocks		Different			*Difficult*	*Difficult*

Based on these facts, we chose two groups of questions: (B), which is algorithm creation from scratch, and (F1), which is the estimated value of the output, to evaluate the students' abilities.

About the problem types, this table shows that the Selection1 problem type is more difficult than the Repetition1 type. Moreover, we found that students who can solve Selection1 can get a higher score in the repetition problems. Repetition1 has "Different" marked in 3 problem groups. This result shows that Repetition1 in this test is not a suitable problem type to evaluate students' abilities.

In 2012, we introduced two of the groups of problems mentioned above. For each problem group, we prepared eight types of problems. The subjects were 95 freshmen who had finished our "algorithmic thinking" course. These students were divided into two groups. Each group was assigned eight problems (four algorithm reading problems, and four algorithm creation problems), one from each problem type. They had to answer all problems in 90 minutes.

The relationship between the C_p index and the percentage of problems answered correctly is shown in Fig.6(a) (algorithm creation problems) and Fig.6(b) (algorithm reading problems). The suitability of the problems was confirmed for all problems. They are located in the "Suitable" area. Though some problems are "Difficult Problems" for this group, overall these problems are suitable to evaluate our students' abilities.



(a) Algorithm creation problems (b) Algorithm reading problems

Figure 6. The relation of the C_p index and the percentage of problems answered correctly for 2012 problems.

4. Conclusion

In this paper, we described the pedagogical features and main functions of a learning support system for algorithmic thinking education. We have used this system since 2008 in our computer science course. The ability to control the usage of algorithmic elements such as input, output, selection or repetition is a unique function that no other system has had before. Teachers can define the element usage rule depending on the learning progress of their learners and their educational philosophy.

Moreover we show our eAssessment function based on the SP score table analysis. By using this function, we have reviewed the problems used in our course. As a result, the pedagogical quality of the problems has improved. In the future, we will pedagogically analyze and design learning scenarios with this supporting system.

Acknowledgements

This work is supported by MEXT/JSPS Grant-in-Aid for Scientific Research (KAKENHI) : 22300286.

References

Bruner, J. : The Process of Education, Harvard University Press, MA, 1960.

Cook, M.E. : Guided Discovery Tutoring: A Framework for Icai Research, Van Nostrand Reinhold, NY, 1991.

Fuwa, Y., Kunimune, H., Kayama, M., Niimura, M., & Miyao, H.,: Implementation and Evaluation of Education for Developing Algorithmic Thinking for Students in Computer Science, The Institute of Electronics, Information and Communication Engineers, Technical Report of ET, 109(268), 51-56, 2009.

Futschek, G. & Moschitz, J.: Learning Algorithmic Thinking with Tangible Objects Eases Transition to Computer Programming, Proc. of 5th International Conference on Informatics in Secondary Schools, 155-163, 2011.

Harnisch, D.L. & Linn, R.L.: Identification of Aberrant Response Patterns: Final Report, Education Commission of the States, 1981.
Hubalovsky, S., Milkova, E., Prazak, P., Hubálovský, Š., Milková, E. & Pražák, P.: Modeling of a real situation as a method of the algorithmic thinking development and recursively given sequences, WSEAS Transactions on Information Science and Applications , 7(8), 1090-1100, 2010.
Knuth, D.E.: Algorithms in modern mathematics and computer science, Stanford Department of Computer Science Report No. STAN-CS-80-786, 1980.
Kayama, M., Satoh, M., Kobayashi, K., Kunimune, H., Hashimoto, M., & Otani M.: Algorithmic Thinking Learning Support System based on Student-Problem Score Table Analysis, International Journal of Computer and Communication Engineering, 3(2), 134-140, 2014.
Kayama, M., Satoh, M., Hashimoto, K., Otani, M., Kunimune, H., & Niimura, M., : Algorithmic Thinking Learning Support System with eAssesment Function, Proc. of the 14th IEEE International Conference on Advanced Learning Technologies, 185, 2014.
Sarawagi, N.: A general education course – "Introduction to Algorithmic Thinking" - using Visual Logic, J. of Computing in Sciences in Colleges, 25(6), 250-252, 2010.
Satoh, T.: A classroom information system for teachers, with foocus on the instructional data collection and analysis, Proc. of ACM '74 annual conference, 1, 199-206, 1974.
Stone, H.S.: Introduction to Computer Organization and Data Structures, McGraw-Hill, New York, 1972.