Informatics in Education, 2024, Vol. 23, No. 3, 525–540

DOI: 10.15388/infedu.2024.23

Number of Program Builds: Another Criterium for Assessing Difficulty of a Programming Task?

Václav DOBIÁŠ, Václav ŠIMANDL, Jiří VANÍČEK
Department of Information Science, Faculty of Education,
University of South Bohemia in České Budějovice
České Budějovice, Czech Republic
e-mail: dobias@pf.jcu.cz, simandl@pf.jcu.cz, vanicek@pf.jcu.cz

Received: October 2023

Abstract. The paper discusses an alternative method of assessing the difficulty of pupils' programming tasks to determine their age appropriateness. Building a program takes the form of its successive iterations. Thus, it is possible to monitor the number of times such a program was built by the solver. The variance of the number of program builds can be considered as a criterion of the difficulty of the task. We seek to verify whether this variance is the greatest in the age group for which the task is most suitable. We created several series of programming tasks and offered them to 87000 pupils from 4th to 13th grade. For each task, we compared the optimal age group determined by the variance of the number of program builds method with the group determined by the correct answer ratio method. A strong correlation was observed in traditional microworlds Karel the Robot and Turtle. A moderate correlation was achieved in the new microworld Movie.

Keywords: task difficulty, programming task, number of program builds, teaching programming, primary school, lower secondary school.

1. Introduction

Currently, the beginnings of teaching programming as an educational instrument for developing computational thinking has been moved to a younger age of pupils (K-12 Computer Science Framework Steering Committee, 2016; CSTA, 2017; Ministry of Education, Youth and Sports of the Czech Republic, 2021). At a younger age, the pupil is unable to work with more complicated programming and other concepts, in a more complex environment, to use a more sophisticated problem-solving strategy. Teachers need to select age-appropriate tasks and integrate them into the curriculum so that their gradation was more suitable. For younger age groups, however, there is a lack of long-term experience with this. From this point of view, the search for effective methods that would allow assessment of the age appropriateness of a task is very important.

Programming tasks can be divided according to:

- The educational goal (to create a program, to recognize what the program does, to determine the final state after running the program).
- The programming concept (loop, condition, variable, parameter).
- The type of response.

When focusing on the type of response, we distinguish between convergent and divergent programming tasks. A convergent task has a single correct answer, which may be arrived at by a range of different methods. A divergent task is more open-ended and provokes a more diverse range of outcomes (Foster, 2015). In school knowledge contests, such as the informatics Bebras Challenge (Dagienė, 2008), convergent tasks are used in order to enable their automatic assessment. In case of divergent tasks, it is not possible to say in advance which solution is correct. They require assessment by an expert. Thus, this paper focuses on convergent tasks.

2. Background

2.1. *How Demanding are Programming Tasks?*

In textbooks and other teaching materials we come across easy tasks that can be solved by most pupils of the given age group and demanding tasks that are a hard nut to crack for a majority of them. When assessing the demands of a task, we distinguish between their difficulty and complexity. Although exact understanding of these concepts differs among authors, the common view is that complexity is concerned with intrinsic properties of a task itself (structure, components, their relations), whereas difficulty is a matter of person-task interaction (Effenberger *et al.*, 2019). This is in line with Yagunova *et al.* (2015) who claim that complexity is an objective factor and difficulty a subjective factor. Some authors use the term cognitive complexity instead of difficulty. Gupta *et al.* (2022) describes cognitive complexity as the effort it takes a person to complete a task or the difficulty encountered when trying to understand the source code. In this context, our paper focuses on difficulty of tasks.

The term complexity is, among other, used in the area of software engineering and cognitive informatics. Al-Batah *et al.* (2019) define complexity as the degree of difficulty in analysis, testing, design and implementation of software. A number of metrics has been developed for comparison of the complexity of a code. The majority of the metrics is based on statistics on the source code. For instance, according to Gupta *et al.* (2022):

- Halstead metrics is determined by counting the number of operators and operands.
- Chidamber and Kemerer metrics is determined by counting the number of functions and inheritance.
- McCabe's Cyclomatic metric calculates a program's complexity by examining its control flow graph.

These metrics are used when comparing different assessing code tools. For example, Dr. Scratch is compared to McCabe's Cyclomatic Complexity and Halstead's metrics (Moreno-Leon *et al.*, 2016). However, the results of individual metrics correlate only partially (Tavares Coimbra *et. al.*, 2018).

It is useful for a teacher or a designer of contest tasks to know how difficult a task will be. It can help them estimate the age appropriateness of the task. Researchers try to determinate the difficulty of a contest task in advance. An example of this is the Bebras Challenge, which contains a series of programming and algorithmic tasks. Organizers of this contest desire to be able to predict accurately which of the tasks will be seen as demanding and which will be considered easy (Tomcsányi, 2009). Van der Vegt (2013) defines task difficulty in two areas: the question answering process and the size of the problem. Vaníček and Křížová (2014) describe several factors that affect a task difficulty, indicators that describe the difficulty and also set a task difficulty index. Effenberger *et al.* (2019) estimate difficulty as the average failure rate, the median solving time, the median number of edits, and the median number of executions.

Determining the difficulty of a pupil's programming task is related to the need to assess pupils' solutions. For automatic evaluation by the test software, it is necessary to create a task whose solution is easy to assess. Then, of course, the question is whether such a task is not too far from real programming. Another possibility is the development of a sufficiently robust automatic task evaluation system (Lokar, 2020; Vaníček *et al.*, 2022). This is not easy, especially since pupil programming tasks are often based on graphics and therefore it is a graphic output that needs to be assessed. In some environments, this output may have problems with accuracy, caused, for example, by the way the canvas is rendered or by floating-point division (Marbach *et al.*, 2022; Vaníček *et al.*, 2022). Common programming environments for children do not automatically evaluate the correctness of the solution and leave this to the problem solver. This is the case of Computing with Emil (Kalaš and Horváthová, 2022), Scratch or Makecode. In such environments, the teacher should point out an error even where pupils do not detect it on their own. This requires a very good level of pedagogical content knowledge from the teacher. However, there is a shortage of qualified informatics teachers who would be able to do it.

2.2. *Assessing Convergent Programming Tasks in the Bebras Challenge*

Common test tasks, such as those in the Bebras Challenge, do not provide feedback after their solution; the solver chooses or constructs the answer and submits it. For programming tasks, this way of answering is unsuitable because it does not support the way programming is commonly done. During their work, programmers receive feedback on how programs they have just built work and where errors appear in the running. It is possible that the error is not of a programming nature, i.e. that its cause is not a misconception or programming inexperience. It can be, for example, a careless mistake while reading the task instructions, insufficient familiarity with the environment, or a mistake in space orientation on a graphic screen (typically left-right turning of the sprite) (Vaníček *et al.*, 2023).

One of the methods of assessing the difficulty of Bebras tasks is the success rate of the solvers. As van der Wegt (2013) states a task that is solved by the majority of the contestants is definitely easier than a task that is solved by a small proportion of them. This approach is used by a number of researchers focusing on the contestants' performance (Stupurienė *et al.*, 2016; Izu *et al.*, 2017; Budinská *et al.*, 2018; Bavera *et al.*, 2020; Delal and Oner, 2020; van der Vegt and Schrijvers, 2013). A similar approach is determining the proportion of unsuccessful task solvers (Jašková and Kostová, 2020) and the proportion of task solvers who decided not to answer it. Vaníček (2016) or Vaníček and Šimandl (2020) found that the proportion of successful solvers and the proportion of solvers who decided not to answer are indicators of a task difficulty. The share of unsuccessful solvers does not work as an indicator here.

Another indicator could be the solving time (Stupurienė *et al.*, 2016). However, if we consider it as the difference between the time of subjecting the answer to the previous and the currently solved task, this may not be the time in which the contestant was really solving the given task (Vaníček, 2016). That is why some researchers focus on the time the contestant spends actively solving the given task (Bellettini *et al.*, 2020).

Another way of assessing tasks and their difficulty is to interview the contestants after the test is finished (Vaníček and Šimandl, 2020). A similar way is to observe how the pupils estimate the difficulty of the task based on their success in solving a previous similar task. Tomcsányiová and Tomcsányi (2014) observed that pupils who did not solve a task correctly later considered the same task difficult and decided not to solve it.

2.3. *How to Compile Didactical Series of Programming Tasks of Growing Difficulty*

Series of programming tasks can grow in terms of programming difficulty given, for example, by new commands, used programming concepts and their combination. However, they can also gradate in other directions. Miková and Hrušecká (2021) studied these topics in tasks for the Blue-Bot robot. They found that the tasks can gradate in the way of work with the robot (direct control, program with delayed execution), in the programming concepts used, the complexity of the situation (given by the situation on the robot pad) and the state of the robot (turning). Vaníček and Miková (2023) define four areas of difficulty gradation in series of programming tasks based on turtle graphics: programming concepts, new commands, geometric aspects of the microworld, the solver's cognitive operations. It follows that in series of programming tasks the difficulty can also be gradated in the non-programming component of the task. The method of gradation depends on the specific order of tasks, e.g. when:

- The task was preceded by a very similar task (and thus the performance is influenced by transfer) (Effenberger *et al.*, 2019).
- There is too much of a cognitive leap between two successive tasks, making the subsequent task appear harder (Vaníček *et al.*, 2023).

3. Aim of the Research, Research Questions

In this paper, we interpret the term "to build a program" as the assembly of blocks by the task solver, including the run of that program. We do not interpret this term as the compilation of a program by the computer.

In addition to determining the success rate of solvers, more advanced approaches such as Classical Test Theory (Kalelioğlu *et al.*, 2022) or Item Response Theory (Belletini *et al.*, 2015; Lonati *et al.*, 2017; Wiebe *et al.*, 2019) can also be used to determine the difficulty of informatics tasks. However, they are suitable for use mainly with closed (multiple choice) test tasks. Their use in assessing the difficulty of interactive tasks with building a program is not appropriate. That is why we looked for other ways of assessing the difficulty of these tasks.

It can be assumed that the same task will be too easy for an older pupil and too difficult for a younger one. It therefore makes sense to look for the age of the pupil for whom the difficulty of this task will be appropriate. If the task is easy, solvers will need only a few iterations to find the solution, while if it is difficult, many iterations will be needed. One of the possible indicators of the difficulty of a programming task could be the number of iterations it takes the solver to solve the task. This indicator would thus provide another possibility of automatic assessment of the difficulty of a task. A number of common programming environments can evaluate, for example, syntactic correctness but cannot assess whether the solver's program fulfils the requirements of the task. For such environments, it would not be difficult to add the functionality of counting the number of builds of the program.

Age appropriateness of a task may be well reflected by the variance of the number of builds of the program. If this variance is small, the task is either very difficult or very easy. In contrast, large variance means that in the group of solvers there is a subgroup for whom the task is easy and a subgroup for whom the task is difficult. If we set one such task to pupils of different ages, we will be able to observe in which age group the number of program builds has the highest variance. This group is then the most suitable for the given task.

With regard to the above considerations, we decided to use the method based on variance of the number of program builds by successful solvers. We assessed the difficulty of several series of tasks of growing difficulty. The aim of the research was to verify the applicability and appropriateness of the above-described method for determining the optimal age of pupils in relation to the difficulty of the task. Based on the stated objective, we posed the following research questions:

- **RQ1:** How can the method based on variance of the number of program builds (VPB for short) be used to identify the optimal age of pupils in relation to the difficulty of the task?
- **RQ2:** To what extent do the results of this method correlate with the results of the reference method based on correct answer ratio?

4. Methodology

In order to answer the research questions, we conducted quantitative research whose respondents were elementary and secondary school pupils. This research was experimental in nature. We recorded and analysed the performance of the respondents in solving a series of tasks based on the above-mentioned method. We compared the results of the analysis according to our method of variance of the number of program builds with the results of the reference method based on correct answer ratio.

4.1. *Series of Tasks*

To verify our hypothesis, we created series of tasks which gradually increase in difficulty and tested them on solvers of different ages. For this purpose, we have developed an environment that allows assessing the difficulty of the task according to our approach and the reference method. The environment should allow:

- The researchers to construct convergent programming tasks.
- The pupils to build (i.e. to edit and run) the program iteratively.
- To automatically assess pupils' solutions.
- To give pupils feedback on whether they succeeded in solving the tasks.
- To record the number of times a pupil has built the program.

We were aware that only an environment familiar and accessible to a large number of pupils at schools would guarantee a sufficient number of respondents. That is why we implemented the environment as a software module into the Bebras Challenge test application.

Several programming microworlds were created within this module:

- Traditional Karel the Robot microworld.
- Traditional Turtle microworld.
- Less traditional Movie environment, see Fig. 1 and (Vaníček and Šimandl, 2023).

This module made it possible to create series of tasks of growing difficulty that we presented to pupils in the form of a test (Vaníček *et al.*, 2022). The series contained from 11 to 14 tasks. From a programming point of view, the tasks in each series ranged in difficulty from simply creating a sequence of blocks through using loops, chaining and nesting them, using conditions when making decisions. The gradation also consisted in adding other restrictions, e.g. a limit to the number of blocks to be used or certain situations in a particular task (objects in the stage, etc.). In the case of using parameters, the gradation consisted in building expressions and combining several parameters.

Pupils solved the task by building a program iteratively. They received feedback from the test application as to whether they had fulfilled the task or which of the requirements had not been fulfilled. At the same time, they could see the progress of the program execution in the graphic window, including any runtime errors (e.g. the character

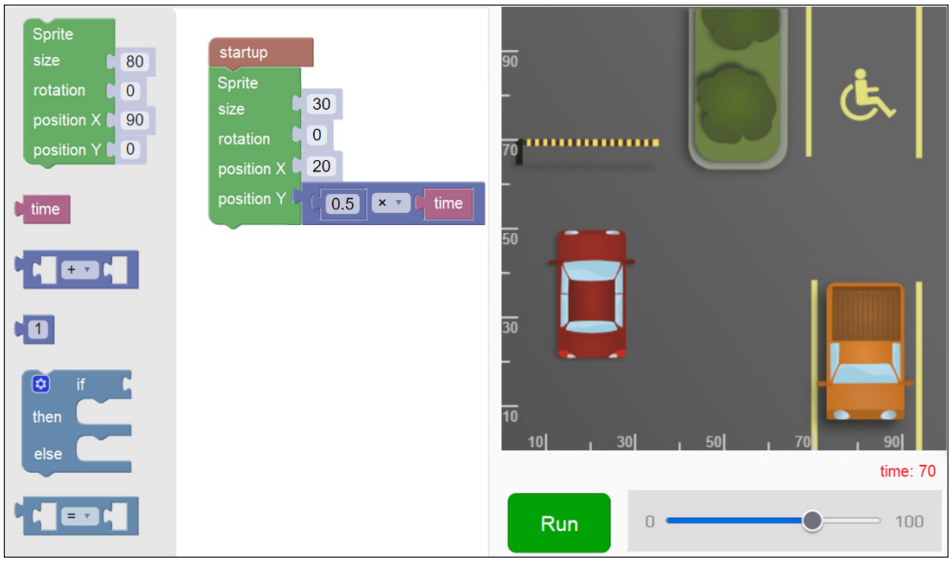

Fig. 1. The microworld Movie. The pupil's task is to program an animation of a sprite. The red car moves according to the pupil's program (in the middle). The program gradually renders the sprite with respect to the time parameter which changes by 1 from 0 to 100.

hitting an obstacle). After solving a task, they moved on to the next task in the series; they could go to it even without solving the previous task. Pupils solved the problems during lessons of informatics in the time limit, so not all pupils managed to solve all the tasks of the series.

4.2. *Respondents, Data Collection and Analysis*

From September 2021 to January 2023, we offered the above series of programming tasks to schools to use them in lessons. During this piloting, we collected data to determine how many times a certain respondent built the program and whether the task was solved successfully. Based on this data, we compiled an anonymized table of all respondents for each series of tasks. Table (see Fig. 2) shows the number of times respondents built their programs and the correctness of their solution.

Based on this table, it is possible to monitor the progress of solving a series of tasks by individual respondents. For illustration, let us study the data of respondent 62969 (let's say Eva) in table (see Fig. 2): She solved the first five tasks successfully – easily accomplishing the first three tasks (on the 6th, 7th and 7th build of the program) but struggling with the next two tasks, needing 16 builds for the fourth and 47 for the fifth. She failed to solve the sixth task, although she built the program 31 times. She did not solve any other tasks, she probably ran out of time given by the teacher.

In total, we gained data from 87.453 respondents from grade 4 (9–10 years old) to the final year of upper secondary school (18–19 years old); see details in Table 1.

Respondent	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14	Task 15
62922	1	1	3	7	24	4	36	6	7	11	18	18	5	1	1
62946	3	6	2	20	25	10	4				1		1		
62952	7	8			4	1	14	5		6	1				
62969	6	7	7	16	47	31									
62957	2	9	3	33	21	65	18	7							
62961	5	12	8	26	41	39	16	5	8	25	17	11	1	4	2
62964	1	1	1	1	5	2	5								
62970	1	3	3	6	9	3	7	6	3	7	3	8	11	31	30
62972	1	4	1	5	11	15	13	4	2	8	8	9	5	39	15
62974	1	7	3	3	60	12	5	15	7	9	9		2		
62982	1	2	4	2	13	8	8	9	7	5	4	4	6	11	30
62986	1	3	1	8	1	1	1	1	1	1	1	16	1		
62987	2	3	3	2	2	7	3	4	6	4	5	2	4	11	19

Fig. 2. Sample of source data on solving the series of tasks in the Movie microworld. Each row provides information about one respondent. The number in the cell means the number of builds of the program. Green coloured cell indicates that the task was solved. Red coloured cell indicates not solved task. An empty cell means that the respondent skipped the task or had no time to solve it.

Table 1

Numbers of respondents in the different series of tasks according to grades

Grade	4	5	6	7	8	9	10	11	12	13	Sum
Karel	2474	5542	7639	5908	4348	3753	2958	1961	610	710	35903
Movie	118	290	1021	1275	1569	1256	1072	714	135	40	7490
Turtle	1495	3671	8007	8912	7638	6228	3860	2694	872	683	44060

For each task, we worked with the data from all the pupils who had successfully solved it. For each age group and each task, we calculated the variance of the number of times the program was built by individual respondents. The grade identified as the most appropriate for work with the particular task was the one in which the value of the variance was maximum. In the case of an insignificant maximum (less than 5% compared to ambient values), we also included these ambient values.

To answer RQ2, we compared our VPB method with the reference method based on correct answer ratio (van der Wegh, 2013). For the reference method, we focused on the school grade in which the success rate exceeded the predetermined reference value for the first time. We considered this grade as the most suitable grade for the given task.

This value was set as the proportion of contestants in the Bebras Challenge who had achieved a better result in the contest than if they had chosen the answers randomly in the multiple-choice test tasks. In 2022, this proportion of such contestants was 75%. In addition, this value coincides with the 75% success rate determined by Klinkeberg *et al.* (2011), according to which this boundary makes the tasks challenging, yet not too difficult.

Because these series of tasks had a time limit, there were situations in which some pupils did not have time to solve some tasks due to time constraints. In such cases, we did not take unsolved tasks into consideration.

5. Results

To answer RQ1, we worked with the variance of the number of program builds in different age groups of pupils for all solved tasks. The results of the analysis of these variances for each of the three microworlds are shown in tables (see Fig. 3 to Fig. 5). The rows of the tables show individual tasks from the series, the columns the grade. For each task, the highest variance value (or the values that are at most 5% smaller), are highlighted. In the tables it can be seen in which grades the variance was the greatest.

We can see a specific pattern in all three tables (Fig. 3, Fig. 4, Fig. 5). Reading the table row by row and considering each task separately, the variance of the number of program builds was low at first, gradually increased, culminated and began to decrease. For easy tasks, the culmination was already reached in the 4th grade and the variance only decreased in higher grades.

	4	5	6	7	8	9	10	11	12	13
T1	36.0	37.5	37.4	34.4	30.0	24.3	20.5	19.4	19.5	15.7
T2	14.9	14.3	13.6	12.7	11.1	10.2	7.4	7.3	5.2	8.6
T3	21.9	17.9	19.7	16.3	14.7	12.7	9.7	7.7	7.4	7.5
T4	25.9	28.9	25.2	19.4	19.5	18.6	15.2	13.0	8.5	10.5
T5	10.7	11.9	11.4	12.3	10.5	8.7	10.8	8.2	8.6	9.3
T6	45.8	49.0	49.0	48.2	36.9	33.6	20.8	17.9	15.1	20.2
T7	24.2	27.1	27.0	24.6	22.7	21.3	17.2	15.2	17.2	16.2
T8	30.2	38.3	49.7	52.6	53.9	57.6	48.9	32.2	42.0	53.9
T9	11.9	13.2	18.8	17.8	17.9	14.6	13.4	11.4	9.2	13.5
T10	20.4	23.4	26.7	25.0	23.7	26.2	21.0	22.5	18.2	17.9
T11	16.3	14.7	15.9	17.8	13.7	14.6	12.5	10.7	9.5	11.0

Fig. 3. Variance of the number of program builds in individual tasks of the Karel the Robot microworld with respect to the grade.

	4	5	6	7	8	9	10	11	12	13
T1	113,9	67,9	45,9	40,9	29,2	31,2	30,3	30,5	26,8	24,2
T2	136,7	86,2	80,7	50,3	55,9	45,7	56,5	45,6	47,2	44,4
T3	147,4	108,2	100,8	86,8	77,9	71,2	73,5	57,4	66,7	58,5
T4	153,9	123,9	112,3	98,6	87,5	86,2	70,6	64,1	70,8	64
T5	146,3	168,5	195,9	171	161,3	172,7	154,6	130,2	133,9	94,8
T6	85,2	73,8	90,6	75,7	69,3	65,7	56,7	66,9	74,2	75,1
T7	190	115,7	115,8	107,6	88,2	86,3	57,3	59,2	71,2	46,8
T8	36,9	50,6	51,7	43	34,9	36,1	31,2	26,5	43,3	33,7
T9	57,5	112,5	195,2	84,6	76,4	70,2	63,7	60	67,9	97
T10		110,7	244,5	150,9	147,6	106,3	106,2	126,2	53,4	47,6
T11		49,8	92,2	69,9	78,6	78	69,6	69,1	136,4	64,9

Fig. 4. Variance of the number of program builds in individual tasks of the Turtle microworld with respect to the grade.

	4	5	6	7	8	9	10	11	12	13
T1	5,9	11,5	11,2	3,8	4,3	3,1	3,9	2,7	3,1	3,3
T2	58	38,1	34,6	19,6	20,7	19	17,6	17	12,8	10,9
T3	77	80,3	30	28,5	28,3	16,8	23,3	21,6	12	3,5
T4	147	178,8	127	154,1	122,8	159,7	104,6	105,9	98,6	18,4
T5	198	248,2	239,4	322,4	295,6	288,7	271,8	348,2	295,2	202
T6	31,9	169,5	103,2	126	160	137,5	90,8	112,1	140	26,4
T7		93,4	129,9	166,6	180,6	189,9	112,8	200,9	89,3	67
T8		21,3	21,3	35,4	44,6	40,2	37,4	41,6	22,6	9,3
T9		34,1	76,1	65	58,8	55,9	51,1	60	58,7	12,4
T10		63	74	103,9	109,6	76,1	86,9	84,7	72,3	55
T11			94,6	108,2	132	105,9	154,2	83,7	63,7	49,2
T12			79,7	176,2	366,4	146,4	141,5	111,6	72,2	
T13			73,3	170,4	159,3	53,2	103,8	126,1	27	
T14			245,9	361,3	470	425,7	474,1	289,3	181,8	

Fig. 5. Variance of the number of program builds in individual tasks of the Movie microworld with respect to the grade.

We assume that if a task was easy, all pupils needed few iterations to solve the task and their variance was low. If the task was difficult, then the majority of pupils needed to build the program many times, and thus the variance was also low. A group of mixed ability pupils had a higher variance of the number of program builds. It was precisely for such a group that the task was reasonably demanding. We assume that the greatest variance indicated the optimal age of the pupil for this task.

An issue arising from RQ2 is the extent to which the areas we identified as areas with the greatest variance correlated with pupils' success rate. To answer it, we used the reference method. Here, we considered the most appropriate grade for the task to be the grade where the success rate exceeds 75% for the first time (see the argumentation for this limit in section 4.2). The results obtained by both methods are shown in Table 2 for the Karel the Robot microworld.

Correlation analysis allowed us to compare the relationship between the values found by the VPB method and the reference method. The Spearman correlation coefficient between the values shown in Table 2 was 0.7. We proceeded in the same way for the other series of tasks. For the Movie the correlation was 0.62, for the Turtle tasks it was 0.86. All the above correlations were statistically significant at the $\alpha = 0.05$ level.

Table 2

Results gained using the VPB method and the reference method. Each of the methods identifies the grade for which the task is optimally demanding in the series of tasks in the Karel the Robot microworld

Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
VPB method	4,5,6	5,6	4	5	5,7	5,6,7	5,7	9	6,8	5,9	7
Reference method	4	4	4	4	4	6	7	10	7	7	4

This allowed us to answer RQ2. There were significant correlations between the VPB method and the reference method. In the case of the Karel the Robot and Turtle, there were strong correlations, and in the case of the Movie there was a moderate correlation according to Dancey and Reidy (2017). Thus, we can answer RQ1 that the optimal age of pupils can be identified with respect to the difficulty of the programming task using the VPB method.

6. Discussion

We have shown that there were significant correlations between the variance of the number of program builds (VPB) method and the reference method of determining the age appropriateness of a programming task. These correlations exist for different kinds of programming tasks. This shows the general applicability of this method.

One of the possible reasons why the Movie microworld has lower correlation than the other two traditional microworlds may be the way the task was set. In this microworld, pupils must at least once run an "empty" program in order to see how the programmed sprite should behave during animation. Without it, they cannot even decide whether to solve the task. In the VPB method, this does not affect the variance of the number of program builds. In the case of the reference method, pupils who decided not to solve the task and only looked at the assignment are counted among the unsuccessful solvers. This affects the success rate. Both methods can therefore differ more in this microworld.

We also considered other reference methods. We could not use the methods Classical Test Theory (Kalelioğlu *et al.*, 2022) or Item Response Theory (Belletini *et al.*, 2015; Lonati *et al.*, 2017; Wiebe *et al.*, 2019) because they work with multiple-choice tasks. We also could not use the method of subsequent interviewing of the respondents (Vaníček and Šimandl, 2020) as used series of tasks were not part of the Bebras contest itself and we had no tool that would motivate respondents to give us the feedback.

Effenberger *et al.* (2019) determined the difficulty of tasks as the average of four factors: the failure rate, the median solving time, the median number of edits, and the median number of executions. In contrast, our VPB method uses only one variable: number of program builds. At the same time, it corresponds well to the reference method.

The VPB method can only be used with high number of participants. In small groups, if there is a learner who likes to experiment and generates a lot of program builds, this will significantly increase the variance of the number of program builds of the whole age group. To minimize such problems, we recommend a minimum number of about 15 pupils in a group. According to Hozo *et al.* (2005), 15 is the borderline between a very small sample and a moderately sized sample. Smaller group sizes could have distorted the results. Larger minimum group sizes increase the requirements for the total research sample size. With 10 age groups in the research, this would mean at least 150 participants. The exclusion of extreme cases from the data (Dancey and Reidy, 2017) will allow further reduction of the number of participants, but carries with it an ethical problem, as these cases may not always be well identified in the data.

The VPB method is influenced by how the series of tasks is constructed by the authors. Because of this, the VPB method and the reference method may give different results. As a hypothetical example, we present the case of two consecutive tasks in such a series: the first task requires a large knowledge leap compared to the previous one; the second task is a variation of the first task and therefore does not have this leap. A more difficult task is indicated by a lower success rate in the reference method and a higher number of program builds in the VPB method. In our example, the success rate of the second task for the reference method will be similar to the success rate of the first task. However, with the VPB method, the number of program builds will probably be significantly lower than in the first task because the solvers have already overcome the knowledge leap. Then the reference method will indicate two similarly difficult tasks, the VPB method will show a significant decrease in difficulty. The author's experience in creating series of gradating tasks may have an impact on the quality of the VPB method.

The reliability of the VPB method is affected by other factors such as a pupil's motivation, the implementation of the test by the teacher and the related possibility of respondents' cooperation or cheating. Our analytical tools showed that this occurred at a rate of over 5% in some age groups. Cooperation in the group will be reflected in the results in such a way that it tends to suppress extreme opinions. As a result, this reduces the variance of the number of program builds compared to the situation where the solvers were working individually. It is then possible that the task will appear more suitable for a younger group of pupils. The way testing is implemented can affect the quality of the VPB method.

7. Conclusion

The contribution of our research lies in the description of a new method based on variance of the number of program builds. It can be used to determine the appropriateness of the difficulty of a convergent programming task. We compared the results of this VPB method with the reference method based on correct answer ratio. For traditional series of tasks from the Karel the Robot and Turtle microworlds, we could observe a strong correlation between these two criteria. A moderate correlation was seen in the new Movie microworld. All correlations reported are statistically significant.

As part of our research, we examined the variance of the number of program builds among successful solvers of the given task. Pupils who had not built the correct program for the given task were not included in the research. The reason was that some of the unsuccessful solvers might have been able to solve the problem, but there were other reasons for their failure. In our opinion, these reasons include the approaching time limit for the end of the test or the pupil's fatigue towards the end of the series. Another factor may be the low motivation of pupils in schools where they were asked to sit the test.

The limit of our research is the fact that in the Czech Republic teaching programming was not compulsory at the time of the research, yet it was implemented by many

schools. Some pupils of a certain grade included in the research had completed some form of programming education while others had not. This fact may have affected the variance of the number of program builds. The technical limit of the research is the uncertainty as to whether the pupils really attended the grade they declared.

Our VPB method can also be used to determine the most suitable task for a certain age group of pupils. For this purpose, we need a series of tasks with gradating difficulty for pupils of one age group. The first tasks in the series should be easy for all pupils, i.e. with a lower variance. The following tasks should be progressively more demanding, so that more pupils need more iterations to find the solution and variance will increase. As the difficulty of the tasks continues to grow, the variance decreases from a certain point. It can therefore be assumed that in a given series of gradating tasks there will be a task whose difficulty is the most suitable for the given group of pupils. It will be the one that has the highest builds number variance. The location of the task with the highest variance in that series could give an indication of how difficult the entire series is.

The VPB method expands the range of tasks for which it is possible to determine their appropriateness for the age of pupils. It could be used even for convergent programming tasks, where it is not possible to automatically evaluate the correctness of the solution and the time to solve the task is not limited. It would thus be possible to use it to assess the difficulty of convergent tasks within almost any programming environment, if a plug-in enabling monitoring of the number of program builds is inserted.

References

Al-Batah, M.S., Alhindawi, N., Malkawi, R., Al Zuraiqi, A. (2019). Hybrid Technique for Complexity Analysis for Java Code. *International Journal of Software Innovation*, 7(3), 118–133. DOI: 10.4018/IJSI.2019070107

Bavera, F., Quintero, T., Daniele, M., Buffarini, F. (2020). Computational Thinking Skills in Primary Teachers: Evaluation Using Bebras. In: Pesado, P., Arroyo, M. (Eds.), *Computer Science – CACIC 2019. CACIC 2019. Communications in Computer and Information Science*, vol. 1184. Springer, Cham, 405–415. DOI: 10.1007/978-3-030-48325-8_26

Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A., Torelli, M. (2015). How Challenging are Bebras Tasks? An IRT Analysis Based on the Performance of Italian Students. In: *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE ,15)*. Association for Computing Machinery, New York, 27–32. DOI: 10.1145/2729094.2742603

Bellettini, C., Lonati, V., Monga, M., Morpurgo, A. (2020). Behind the Shoulders of Bebras Teams: Analyzing How They Interact with the Platform to Solve Tasks. In: Lane, H.C., Zvacek, S., Uhomoibhi, J. (Eds.), *Computer Supported Education. CSEDU 2019. Communications in Computer and Information Science,* vol 1220. Springer, Cham, 191–210. DOI: 10.1007/978-3-030-58459-7_10

Budinská, L., Mayerová, K., Winczer, M. (2018). Gender Differences in Graph Tasks – Do They Exist in High School Bebras Categories Too? In: Pozdniakov, S., Dagienė, V. (Eds.), *Informatics in Schools. Fundamentals of Computer Science and Software Engineering. ISSEP 2018. Lecture Notes in Computer Science*, vol 11169. Springer, Cham, 295–306. DOI: 10.1007/978-3-030-02750-6_23

CSTA (2017). *K-12 Computer Science Standards*. https://csteachers.org/k12standards/

Dagienė, V. (2008). The BEBRAS contest on informatics and computer literacy – students' drive to science education. In: *Joint Open and Working IFIP Conference. ICT and Learning for The Next Generation*. 214–223. `https://www.bebras.org/sites/default/files/documents/publications/DagieneV-2008.pdf`

Dancey, C.P., Reidy, J. (2017). *Statistics without Maths for Psychology* (7th ed.). Pearson Education, London.

Delal, H., Oner, D. (2020). Developing middle school students' computational thinking skills using unplugged computing activities. *Informatics in Education*, 19(1), 1–13. DOI: 10.15388/infedu.2020.01

Effenberger, T., Čechák, J., Pelánek, R. (2019). Measuring difficulty of introductory programming tasks. In: *Proceedings of the Sixth (2019) ACM Conference on Learning @ Scale*. Association for Computing Machinery, New York, 1–4. DOI: 10.1145/3330430.3333641

Foster, C. (2015). The Convergent-Divergent Model. *Educational Designer*, 2(8). http://www.educationaldesigner.org/ed/volume2/issue8/article28/

Gupta, M., Rajnish, K., Bhattacharjee, V. (2022). Cognitive Complexity and Graph Convolutional Approach Over Control Flow Graph for Software Defect Prediction. *IEEE Access*, 10, 108870–108894. DOI: 10.1109/ACCESS.2022.3213844

Hozo, S.P., Djulbegovic, B., Hozo, I. (2005). Estimating the mean and variance from the median, range, and the size of a sample. *BMC Medical Research Methodology,* 13(5), 1–10. DOI: 10.1186/1471-2288-5-13

Izu, C., Mirolo, C., Settle, A., Manilla, L., Stupurienė, G. (2017). Exploring Bebras tasks content and performance: A multinational study. *Informatics in Education*, 16(1), 39–59. DOI: 10.15388/infedu.2017.03

Jašková, L., Kostová, N. (2020). Difficulty of Bebras tasks for lower secondary blind students. In: Kori, K., Laanpere, M. (Eds.), *Informatics in Schools. Engaging Learners in Computational Thinking. ISSEP 2020. Lecture Notes in Computer Science*, vol 12518. Springer, Cham, 3–14. DOI: 10.1007/978-3-030-63212-0_1

K-12 Computer Science Framework Steering Committee (2016). *K-12 Computer Science Framework*. Association for Computing Machinery, New York. https://dl.acm.org/doi/book/10.1145/3079760

Kalaš, I., Horváthová. K. (2022). Programming concepts in lower primary years and their cognitive demands. In: Passey, D., Leahy, D., Williams, L., Holvikivi, J., Ruohonen, M. (Eds.), *Digital Transformation of Education and Learning – Past, Present and Future. OCCE 2021. IFIP Advances in Information and Communication Technology*, vol 642. Springer, Cham, 28–40. DOI: 10.1007/978-3-030-97986-7_3

Kalelioğlu, F., Doğan, D., Gülbahar, Y. (2022). Snapshot of Computational Thinking in Turkey: A Critique of 2019 Bebras Challenge. *Informatics in Education*, 21(3), 501–522. DOI: 10.15388/infedu.2022.19

Klinkenberg, S., Straatemeier, M., van der Maas, H.L.J. (2011). Computer adaptive practice of Maths ability using a new item response model for on the fly ability and difficulty estimation. *Computers & Education*, 57(2), 1813–1824. DOI: 10.1016/j.compedu.2011.02.003

Lokar, M. (2020). Pišek – Programming with Blocks Competition: A new Slovenian Programming Competition. In: Kori, K., Laanpere, M. (Eds.), *Proceedings of the International Conference on Informatics in School: Situation, Evaluation and Perspectives. ISSEP 2020. CEUR Workshop Proceedings*, vol. 2755. 1–12. https://ceur-ws.org/Vol-2755/paper1.pdf

Lonati, V., Monga, M., Malchiodi, D., Morpurgo, A. (2017). How presentation affects the difficulty of computational thinking tasks: an IRT analysis. In: *Proceedings of the 17th Koli Calling International Conference on Computing Education Research (Koli Calling ,17)*. Association for Computing Machinery, New York, 60–69. DOI: 10.1145/3141880.3141900

Marbach, J., Maximova, A., Staub, J. (2022). A Tool to Create and Conduct Custom Assessments in Turtle Graphics. In: Bollin, A., Futschek, G. (Eds.), *Informatics in Schools. A Step Beyond Digital Education. ISSEP 2022. Lecture Notes in Computer Science*, vol 13488. Springer, Cham, 15–26. DOI: 10.1007/978-3-031-15851-3_2

Miková, K., Hrušecká, A. (2021). Identifikácia gradácie informatických konštruktov pri vyučovaní edukačnej robotiky na 1. stupni ZŠ (Identification construct gradation in robotics educational at primary school). In: Horváthová, D., Michalíková, A., Škrinárová, J., Voštinár, P. (Eds.), *Proceedings of conference DidInfo 2021*. Matej Bel University, Banská Bystrica, 118–122. https://www.didinfo.net/images/DidInfo/files/DIDINFO_2021_zbornik.pdf

Ministry of Education, Youth and Sports of the Czech Republic (2021). *Rámcový vzdělávací program pro základní vzdělávání* (Framework Education Programme for Elementary Education). Ministry of Education, Youth and Sports of the Czech Republic, Praha. https://www.edu.cz/wp-content/uploads/2021/07/RVP-ZV-2021.pdf

Moreno-Leon, J., Robles, G., Roman-Gonzalez, M. (2016). Comparing computational thinking development assessment scores with software complexity metrics. In: *2016 IEEE Global Engineering Education Conference (EDUCON)*. Institute of Electrical and Electronics Engineers, Abu Dhabi, 1040–1045. DOI: 10.1109/EDUCON.2016.7474681

Stupurienė, G., Vinikienė, L., Dagienė, V. (2016). Students' Success in the Bebras Challenge in Lithuania: Focus on a Long-Term Participation. In: Brodnik, A., Tort, F. (Eds.), *Informatics in Schools: Improvement of Informatics Knowledge and Perception. ISSEP 2016. Lecture Notes in Computer Science*, vol 9973. Springer, Cham, 78–89. DOI: 10.1007/978-3-319-46747-4_7

Tavares Coimbra, R., Resende, A., Terra, R. (2018). A Correlation Analysis between Halstead Complexity Measures and other Software Measures. In: *2018 XLIV Latin American Computer Conference (CLEI)*. Institute of Electrical and Electronics Engineers, Sao Paulo, 31–39. DOI: 10.1109/CLEI.2018.00014

Tomcsányi, P. (2009). Náročnosť úloh v súťaži Informatický bobor (Difficulty of tasks in the Bebras contest). In: *Proceedings of conference DidInfo 2009*. Matej Bel Univerzity, Banská Bystrica, 170–173.

Tomcsányiová, M., Tomcsányi, P. (2014). Analýza riešení úloh súťaže iBobor v školskom roku 2013/14 (Analysis of task solutions in iBobor contest in the school year 2012/13). In: *DidactIG 2014 conference*. Technical University of Liberec, Liberec.

van der Vegt, W. (2013). Predicting the Difficulty Level of a Bebras Task. *Olympiads in Informatics*, 7, 132–139. https://ioinformatics.org/journal/INFOL127.pdf

van der Vegt, W., Schrijvers, E. (2013). Analyzing Task Difficulty in a Bebras Contest Using Cuttle. *Olympiads in Informatics*, 13, 145–156. https://ioinformatics.org/journal/v13_2019_145_156.pdf

Vaníček, J. (2016). What Makes Situational Informatics Tasks Difficult? In: Brodnik, A., Tort, F. (Eds.), *Informatics in Schools: Improvement of Informatics Knowledge and Perception. ISSEP 2016. Lecture Notes in Computer Science*, vol 9973. Springer, Cham, 90–101. DOI: 10.1007/978-3-319-46747-4_8

Vaníček, J., Dobiáš, V., Šimandl, V. (2023). Understanding loops: What are the misconceptions of lower-secondary pupils? *Informatics in Education, 22*(3), 525–554. DOI: 10.15388/infedu.2023.20

Vaníček, J., Křížová, M. (2014). Kritéria obtížnosti testových otázek v informatické soutěži (Criteria of informatics contest tasks difficulty). In: Lovászová, G. (Ed.), *Proceedings of conference DidInfo 2014*. Matej Bel University, Banská Bystrica, 191–199. https://www.didinfo.net/images/DidInfo/files/didinfo_2014.pdf

Vaníček, J., Miková, K. (2023). Sady bobřích úloh se sestavováním programu z bloků (Sets of bebras tasks with assembling programming code from blocks). In: Dudáš, A., Michalíková, A., Voštinár, P., Škrinárová, J. (Eds.), *Proceedings of conference DidInfo 2023*. Matej Bel University, Banská Bystrica, 166–171. https://www.didinfo.net/images/Zbornik_DidInfo%202023_final%201.pdf

Vaníček, J., Šimandl, V. (2020). Participants' Perception of Tasks in an Informatics Contest. In: Kori, K., Laanpere, M. (Eds.), *Informatics in Schools. Engaging Learners in Computational Thinking. ISSEP 2020. Lecture Notes in Computer Science*, vol. 12518. Springer, Cham, 55–65. DOI: 10.1007/978-3-030-63212-0_5

Vaníček, J., Šimandl, V. (2023). Microworlds for Programming Bebras Tasks in Czechia. In: *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 2 (ITiCSE 2023)*. Association for Computing Machinery, New York, 653. DOI: 10.1145/3587103.3594223

Vaníček, J., Šimandl, V., Dobiáš, V. (2022). Bebras Tasks Based on Assembling Programming Code. In: Bollin, A., Futschek, G. (Eds.), *Informatics in Schools. A Step Beyond Digital Education. ISSEP 2022. Lecture Notes in Computer Science*, vol. 13488. Springer, Cham, 113–124. DOI: 10.1007/978-3-031-15851-3_10

Wiebe, E., London, J., Aksit, O., Mott, B.W., Boyer, K.E., Lester, J.C. (2019). Development of a Lean Computational Thinking Abilities Assessment for Middle Grades Students. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE ,19)*. Association for Computing Machinery, New York, 456–461. DOI: 10.1145/3287324.3287390

Yagunova, E., Podznyakov, S., Ryzhova, N., Razumovskaia, E., Korovkin, N. (2015). Tasks Classification and Age Differences in Task Perception. Case Study of International On-line Competition "Beaver". In: Jekovec, M. (Ed.), *The Proceedings of International Conference on Informatics in Schools: Situation, Evolution and Perspectives — ISSEP 2015*. University of Ljubljana, Ljubljana, 33–43. http://issep15.fri.uni-lj.si/files/issep2015-proceedings.pdf

V. Dobiáš is an assistant professor in the Department of Informatics in the Faculty of Education at the University of South Bohemia in České Budějovice in the Czech Republic. His research activities focus on the digital divide and computational thinking.

V. Šimandl is an assistant professor in the Department of Informatics in the Faculty of Education at the University of South Bohemia in České Budějovice in the Czech Republic. His research activities focus on the area of Informatics education in lower secondary schools and programming. He has been organizing the Czech Bebras Challenge for 10 years.

J. Vaníček is an associate professor and the head of the Department of Informatics in the Faculty of Education at the University of South Bohemia in České Budějovice in the Czech Republic. His area of interest is Informatics education in primary and lower secondary schools and early age programming. He is an author of 7 textbooks about information technology and programming. Between 2017 and 2020 he was a head of the strategic project PRIM developing new Czech national informatics curricula. He has been organizing Bebras Challenge for 16 years and he is a representative of the Czech Republic in the International Bebras Committee.