Research of Methods for Practical Educational Tasks Generation Based on Various Difficulty Levels

Dmytro Malikin [1] and Iryna Kyrychenko [2]

[1] *Kharkiv National University of Radioelectronics, Nauky Ave. 14, Kharkiv, 61166, Ukraine*
[2] *Kharkiv National University of Radioelectronics, Nauky Ave. 14, Kharkiv, 61166, Ukraine*

Abstract

E-learning is a relevant area for research, especially in the context of evaluating the completion of tasks by users. A lot of systems use an approach when a constant set of practical tasks is used to check the user's knowledge. This paper proposes an algorithm to create dynamic tasks based on various difficulty levels. Moreover, this algorithm is explained in such a way that it can be applied in several areas of learning with several modifications. Using clustering methodologies can be helpful in implementing such an algorithm to make practical tasks solving more valuable and useful.

Keywords

E-Learning, MOOC, Clustering, Generation Process, Difficulty Level

1. Introduction

Nowadays, distance learning and e-learning especially is an actual area for research. Unlike traditional education with a teacher in the classroom, online learning offers the benefits of being able to learn at a student's own pace and having constant access to knowledge resources. The pandemic has also significantly influenced the transition to such a case, with a large part of pupils and students studying at home.

In addition to primary education, there is also a niche for training on various MOOC platforms, where people can receive knowledge that they do not need for professional activities but may be of personal interest to them. Even though in today's world, people have sufficient learning opportunities, both educational institutions and online platforms have a shortcoming, which is expressed in a particular specific curriculum. For people who can show different knowledge acquisition levels for one reason or another, practical tasks of the same level are provided to consolidate the theory. It happens mainly because the result of education should be the student's compliance with a certain level, which is tested by a set of exams, tests and more. However, this does not solve the problem of understanding particular principles that lead to the solution to the exam's tasks. Allowing understanding exactly how tasks should be solved, and not just knowing which answer is appropriate for this task, increases the benefits of acquiring knowledge.

People are different, so the tasks given to them should also be different, especially when the education process goal is to teach a person, not just to give information without any influence.

The paper proposes to consider the option of user training using a set of practical tasks generated by different difficulty levels depending on the success of the user's tasks at previous levels. In such a way, users that show good results in tasks solving will receive more complex tasks to improve their skills. Moreover, the users that have some difficulties in understanding would have simpler tasks with a higher chance of understanding the given theory better.

COLINS-2022: 6th International Conference on Computational Linguistics and Intelligent Systems, May 12–13, 2022, Gliwice, Poland
EMAIL: dmytro.malikin@nure.ua (D. Malikin); iryna.kyrychenko@nure.ua (I. Kyrychenko)
ORCID: 0000-0002-7554-6185 (D. Malikin); 0000-0002-7686-6439 (I. Kyrychenko)

This approach can be useful mainly for MOOC systems. Online courses are a great place to test such a solution since the process of introduction of such an approach into the education system can take a long time. However, it can be done if the results of the testing are satisfying.

2. Related works

One of the main problems why performing practical online course tasks in their usual sense can be ineffective is that the set of tasks is static. It does not change depending on who performs it. Therefore, all course participants solve the same set of tasks. For the most part, the courses themselves may contain only one assignment per section. Therefore, some have 5 minutes to read the theoretical material and complete the practice, while others will have to go back to the beginning several times to solve the problem finally. Such a condition applies primarily to massive open online courses (MOOC) where the students' diversity is high enough since anyone can access the course they want. Therefore, the individuality of a student and one's specific learning progress should be considered when producing a set of practical tasks.

Research on the impact of dynamic tasks on students' performance is presented in work by Dagiene et al. (2017) [1]. As stated by the authors, if learners are challenged with non-routine tasks at an appropriate level, their cognitive skills can be significantly improved. Having a difficulty level of the task corresponding to students' success in solving this task type can help with their motivation. A student who knows how to solve a task on a low level is more likely to solve a similar task on a higher level and more likely to be interested in such solving. On the contrary, if a student is faced with a complex problem, it can affect one's motivation and, therefore, decrease the effectiveness of the learning process.

Kormos and Wilby, in their work (2019), state that an essential aspect of tasks that can influence students' motivation to engage with them is their content. Personally relevant task content can arouse situational interest, which in turn can lead to increased attention, sustained effort and enjoyment and ultimately result in more learning [2]. The same applies to a specific difficulty level of the task. Problem difficulty is considered by the student as personal and, therefore, relevant.

Students' motivation is an essential element of high-quality education, according to Yilmaz, Sahin and Turgut (2017) [3]. It can increase the chance that students would solve the task with a high level of enthusiasm. In order to raise students' motivation on tasks solving different task difficulty levels should be considered. Students perceive their tasks differently. Researchers have shown that task difficulty can influence students' self-regulation. Difficulty can also affect students' problem-solving strategies and tactics. Nawaz et al. (2021) state that too easy or vice versa, too complex problems would have much fewer benefits from solving them [4]. If the task has a proper difficulty level and it is a prior knowledge problem, it may lead to better learning outcomes. Students perceiving such practical tasks may have higher confidence in knowledge.

There are also works that talk about actual methods that allow to improve decisions made on which tasks would be given to students. As stated in work by Porayska-Pomsta (2016), artificial intelligence (AI) methodologies provide a powerful tool for supporting educational practitioners' understanding and innovative practices [5]. However, sharing practices between teachers is still one of the main conditions for the successful improvement of learning processes. That is why combining AI tools and practitioners' sharing would raise the current level of learning effectiveness in MOOC specifically.

In work by Zou et al. (2019) knowledge graph algorithm is proposed to create practical tasks for students. The algorithm splits knowledge into several groups: ready to learn, unready, and already mastered. Such an approach generates tasks based on the performance or prerequisite skills of a user that solves them [6].

To sum up mentioned statements, high students' motivation is one of the main conditions of successful learning. Motivation is highly influenced by content and level of personalization in problems they need to solve while educating. Also, the more tasks are dynamic in changing their difficulty, the more enthusiasm is shown by the student while solving them.

Different AI methods are already used to improve the process of education that shows a good level of applicability in this specific area. There are existing methods in tasks generation based on previously solved tasks and students' experience and knowledge at all.

Modern AI and machine learning (ML) technologies greatly influence everyone's life. While education is one of the primary needs for people, introducing AI and ML methods would result in significant improvements in the education process.

So, the purpose of this research is to attempt to find an algorithm for practical tasks generation based on different levels of difficulty and current students' progress.

3. Proposed algorithm

In general, the algorithm should be able to generate sets of tasks with a specific level of difficulty given this level as an input.

Suppose there is some problem T with parameters $A = \{a_1, ..., a_n\}$. The parameters can take different forms. Considering such problems in the context of mathematics, we can assume that T is correlated with a certain example or problem, for example, an equation that needs to be solved. Accordingly, there is a certain set of $x = \{x_1, ..., x_n\}$, which is the answer to the problem. At this stage, we omit the possible cases when the solution is special, for example, the lack of roots in the equation.

In equation

$$ax^2 + bx + c = 0, \quad (1)$$

the set A would contain (a, b, c). Then, we can say that depending on the values of A changes the difficulty of T. In the terminology of information technology, in fact, the complexity of the algorithm as such will not change. Considering the time complexity in the big O notation, one uses only the asymptotic value, not paying attention to specific data. If the algorithm executes n operations at n input values, its time complexity will remain linear, regardless of which values were submitted to the input.

However, it is relatively easier for a person to perform calculations on numbers with two digits than on numbers with five digits. This specific difficulty is used to determine the level of tasks in this work.

Having T and the parameters A, it is also necessary to provide the algorithm F such that

$$F(T(A)) = x \quad (2)$$

So, F is the algorithm for solving the problem and x in (2) is the solution for T.

There are also some types of tasks when it is necessary to find answers to several questions from one problem. Then the set of the input data for T extends – it needs a set of algorithms $F = \{F_1, ..., F_n\}$, which lead to corresponding solutions $X = \{x_1, ..., x_n\}$.

Let l be a level of difficulty given as an input to the proposed algorithm G. Then, this algorithm can be formulated as

$$G(l, T, F) = (A_l, X) \quad (3)$$

Of course, algorithms for solving most modern mathematical problems already exist, so it is unnecessary to impose on the course creator the need to describe an algorithm for solving, for example, a quadratic equation. A system implementing this algorithm should be able to solve known fundamental problems (not only in math) without their explicit definition.

Thus, it is proposed to design an algorithm that takes T, F and l at the input and generates a certain set A on call, which belongs to the l level of difficulty of the problem T.

It should be noted that for the proposed algorithm, it is necessary to determine the upper and lower limits for each value of A_n, in order to be able to make a precise distribution between the levels of difficulty of the problem. Another way is to determine constant values for a_n. An option with an algorithm extension to generate an infinite number of difficulty levels is possible but is not considered in this paper. Sets containing the upper and lower bounds values would be A_{max} and A_{min} respectively.

Also, specific limitations can be added for x. An example of such a case is when the task with an equation should have a solution with real numbers.

After obtaining A_{max} and A_{min}, sets of A within these limits should be generated. These sets can be defined as sets containing the maximum and minimum values of A_n, respectively, or they can be found dynamically by scoring each set.

Assume that the number of levels of difficulty is equal to L. In a simpler version, it is sufficient to find the step of arithmetic progression for each member of the set $A = \{a_{nmin}, \ldots, a_{nmax}\}$ with the number of members L. This method has certain disadvantages:

- different ways of cheating when solving problems since everybody would have the same task

- when setting the problem, it can be established that X can acquire specific values, and therefore not any arbitrary set A is suitable for the task.

Let us introduce the notation of the difficulty scoring function S. The function should be applied to the A_{max} and A_{min} sets:

$$S_{min} = S(A_{min});\ S_{max} = S(A_{max}) \tag{4}$$

Thus, all possible sets A will have values of difficulty are in the range:

$$S(A) \in [S_{min}, S_{max}] \tag{5}$$

The question arises, how exactly should the function S calculate the difficulty of the set A, since the big O is not suitable in this case.

3.1. Calculating the task difficulty

Since the computing power of the hardware on which the algorithm is executed can be quite different, and the execution time of $F(T(A_{max}))$ could possibly be almost the same as of $F(T(A_{min}))$, a slightly more complex approach to computing difficulty should be considered.

If these times are almost the same, the time on several executions of F can be calculated. The process can be automated depending on the retrieved difference between the time of F_{max} and F_{min} execution. If the difference is less than some ε value, then executions amount is raised. The actual number of executions can be retrieved experimentally.

Another metric for scoring that can be used is the memory amount taken by the process during the execution. Although when the problem T is a mathematical equation of some kind, the memory amount taken can be very low, for different tasks it can be very useful in the process of scoring.

So, to get a difficulty score of the set A, several metrics can be used:

- t – time taken by task execution. If several executions were made to retrieve a more accurate delta between sets A_{min} and A_{max}, then the same number of executions should be applied to the arbitrary generated set
- m – memory amount taken by task execution.

S_{min} can be calculated as:

$$S_{min} = t_{min} * w_t + m_{min} * w_m \tag{6}$$

And S_{max} can be calculated as:

$$S_{max} = t_{max} * w_t + m_{max} * w_m \tag{7}$$

where w_t and w_m are the weights of corresponding metrics that define which metric should influence the score more. Weights also should be found experimentally to achieve a better scoring function.

Of course, other metrics can be found depending on task T, so actually, the scoring function should be as:

$$S = \sum_{i=1}^{n} M_i * w_i \tag{8}$$

where M_i (i = 1, ..., n) is just a set of metrics, and w_i is a set of their weights.

3.2. Levelling parameters sets

The easiest solution for the discussed problem is to provide a course creator with the possibility of defining specific boundaries of each A_i. It would work for specific use cases when the problem T is simple enough for which one can easily define several difficulty levels. Since it is not clear for all problems, then another approach to split task sets should be used. One of them is clustering.

Clustering is the method of grouping a set of objects by their similarity. Objects in one cluster then should be more similar by some specific features [7].

First, sets A should be generated. They can be found as certain combinations of values of a_1, ... a_n, given that a_n can be defined as:

- value from the interval a_{nmin}, …, a_{nmax}
- a specific value from a constant set.

The resulting sets A can be considered as vectors of values. These vectors should be normalized. That means that A should be transformed into a unit vector – a vector with a length of A. With the same rules, the values of their difficulty scores also should be normalized. Normalized vector $\hat{A}$ of a set $\vec{A}$ can be found in the following way:

$$\hat{A} = \frac{\vec{A}}{\|\vec{A}\|} \tag{9}$$

where $\|\vec{A}\|$ is a magnitude of the vector $\vec{A}$.

After the normalization process, the vectors can be used as an input to a specific clustering algorithm. Let us define the number of clusters as L – number of difficulty levels. As a distance function $D(A_1, A_2)$ between the vectors A_1 and A_2 would be used a function that compares the corresponding $S(A_1)$ and $S(A_2)$. As a result of the clustering process, parameters sets would be divided between L clusters, and a specific clustering function would be trained to make a decision in which cluster any new set A can be assigned.

Thus, to implement the whole algorithm G the following steps should be performed:

1. Take function F, an expected result X and the number of difficulty levels L as an input
2. Define boundaries for a_n values – a_{nmin} and a_{nmax}
3. Generate sets A and A_{min} and A_{max} (or find them dynamically after scoring)
4. Filter sets A that should fit for specific solutions X
5. Find scores S_{min}, S_{max}
6. Calculate scores of all other sets
7. Normalize vectors of sets A
8. Pass normalized vectors to clustering model
9. On demand generate new set A and, using the clustering model, assign it to specific cluster l
10. On demand take a difficulty level l and find a new set A that can be assigned to this cluster.

4. Experiment

To test this algorithm, the prototype was created with a quadratic equation as an example of a problem that needs to be solved. At first, the algorithm that solves the problem should be implemented. A quadratic equation can be solved in different ways. For example, the roots of the equation can be found by this formula:

$$x = \frac{2c}{-b \pm \sqrt{b^2 - 4ac}} \tag{10}$$

So, parameters set A can be described as $\{a, b, c\}$.

The steps to perform an experiment are the following:

1. Implement an algorithm described in (10) to solve the problem
2. Generate sets A that meet defined conditions
3. Filter sets A, so they meet the requirements for the answer to the problem
4. Score each set A
5. Cluster them into l levels of difficulty

So, the next step after implementation is a generation of different sets of A. It can be done by simply iterating through ranges for each parameter bounds defined before the experiment. For the experiment for each parameter, a range of [-10, 10] was used with a step equal to 1. So, each parameter would be an integer greater or equal to -10 and less or equal to 10. One exceptional validation for a is that it cannot be equal to 0 since in this case, the equation would not be quadratic. For such conditions, 7600 sets of A were generated. Also, specific validation for the result of the solver was introduced – x_1 and x_2 should be integers, but the equation can have only one root, and this root can be equal to 0. After this filtering of obtained results, the number of sets was reduced to 298.

The next step is to find scores of each set of A. For the experiment, only two scores were used – memory and time of algorithm execution that was made several times. Scores are calculated by the formula (8) with weights of 0.3 for memory and 0.7 for time scores. Minimum and maximum score values of each set A can be found in Table 1.

Table 1
Minimum and maximum scores

Set	Score
(-9, 0, 0)	4.765
(1, 5, 4)	5.321

As can be seen from the result equation $-9x^2 = 0$ is scored as easier to solve in comparison with equation $x^2 + 5x + 4 = 0$.

The next step is to run a clustering process on these values to find sets divided into different difficulty levels. While performing the experiment, three clustering algorithms were chosen [8]. It should be noted that implementation of clustering algorithms is not a goal for this research, that is why already existing libraries were used, particularly scikit-learn for Python. An important requirement for the algorithm is that number of clusters should be predefined. All tested methods were called with default parameters, only clusters number was passed as an input.

After clusters are retrieved, they should be sorted by the average difficulty score in each of them.

The first clustering algorithm is K-Means – the most popular nowadays. Its goal is to divide inputs into clusters in which each input belongs to the cluster with the nearest mean. The amount of sets per cluster after K-Means is applied is shown in Figure 1.

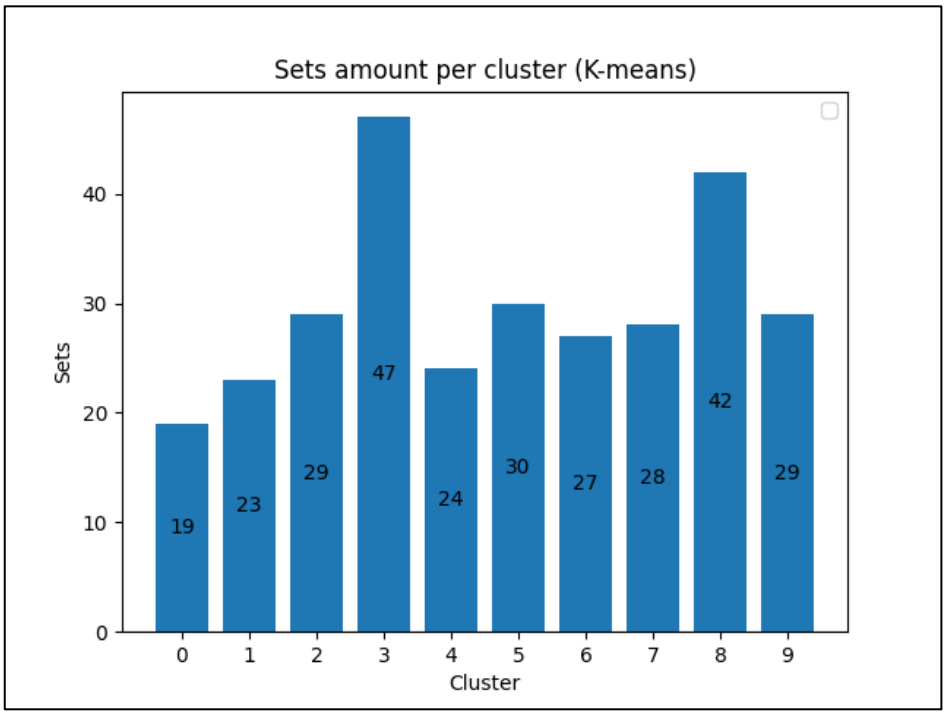

Figure 1: Clustering with K-means

As can be seen from the figure, clusters have a similar number of sets. However, some of them have more sets. Examples of sets from levels 0 and 9 are shown in Table 2. The average score for level 0 is 4.782, and for level 9 is 4.986.

Table 2
K-Means clusters examples

Level	Set	Score
0	(-9, 0, 0)	4.765
0	(-5, 0, 5)	4.798
0	(-7, 7, 0)	4.782
9	(-1, 3, 4)	4.842

9	(-3, 6, 9)	4.796
9	(1, 5, 4)	5.321

These examples contain a case when a set from level 9 has a score less than a set from level 0. It happens because set values themselves were also given as an input to clustering algorithms to find groups not only by score itself but also between specific parameters in sets. Level 0 shows that sets with zeroes in parameters had a lower score which is why in comparison, such tasks can be perceived as easier as it is for the real person.

The next clustering algorithm used in the experiment is agglomerative clustering, that is a subclass of hierarchical clustering. These algorithms are trying to build a hierarchy of clusters. Agglomerative clustering has each input starting in its own cluster, and then pairs of clusters are merged, moving up by hierarchy.

The results of the algorithm applied are shown in Figure 2. There is also a similar number of sets in each cluster.

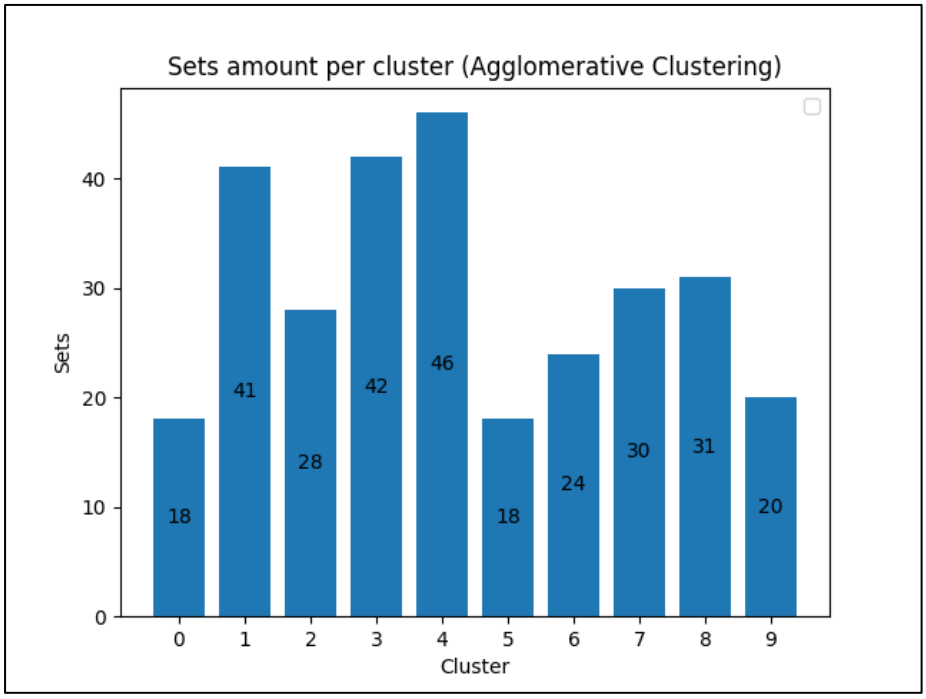

Figure 2: Clustering with Agglomerative algorithm

In Table 3, examples of sets from levels 0 and 9 are shown. The average score for level 0 is 4.802 and for level 9 is 4.837.

Table 3
Agglomerative clusters examples

Level	Set	Score
0	(-3, -9, 0)	4.795
0	(-2, -6, 0)	4.813
0	(-3, -6, 0)	4.799
9	(2, 8, 0)	4.842
9	(1, 8, 0)	4.835
9	(3, 9, 9)	4.835

Results show that different levels contain mostly similar sets of parameters which is why such a clustering algorithm with this specific scoring function cannot be applied to solve the problem.

However, with specific clustering parameters in addition to modified scoring function (for example, with added scoring parameters or changed weights), it can also be tested to give a good solution.

The third clustering algorithm that was used in the experiment is spectral clustering. It uses a spectrum of similarity matrix of the data to perform the reduction in dimensions. Then clustering is made in fewer dimensions.

The results of this method application are shown in Figure 3. As can be seen from this figure, differences in amounts of sets per cluster are more visible in comparison with previous clustering algorithms applied. Level 3, for example, has the most sets, and this amount is almost twice as big as the amount of level 4, which is the second cluster by a number of parameters sets.

Such a case can be very useful in actual tasks creation for educational courses. The figure shows that medium levels of difficulty have most of tasks generated. That would be the main case when using them in real conditions because most people would probably solve tasks exactly from these levels.

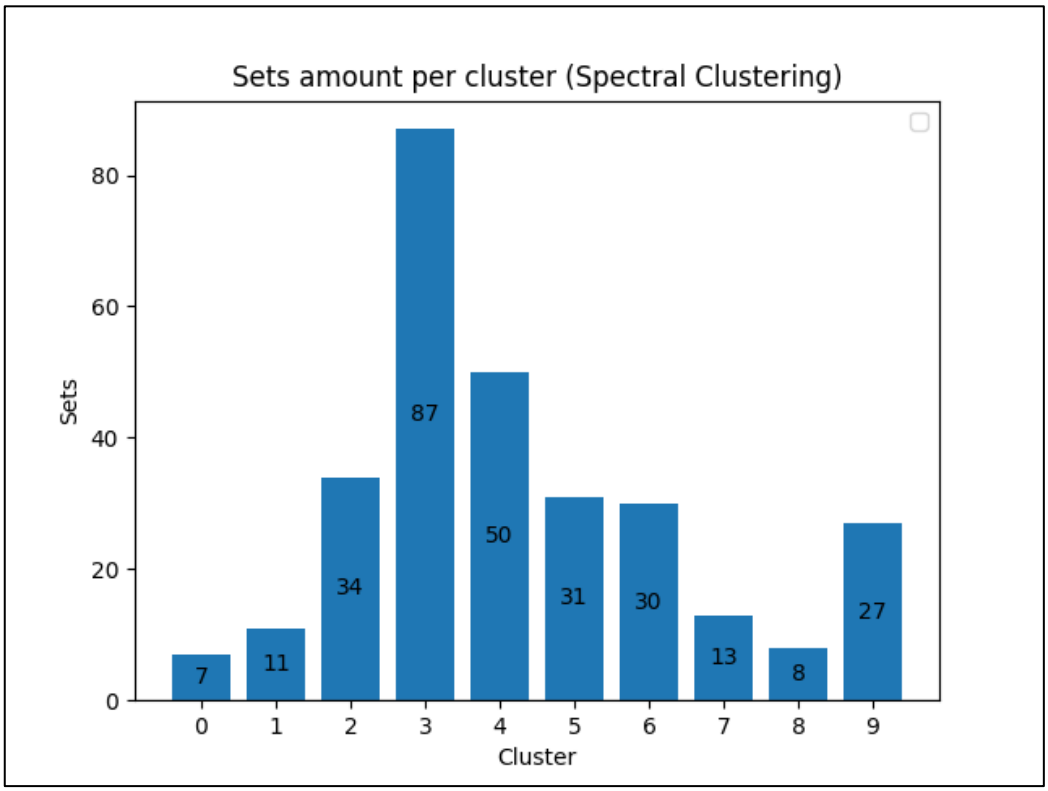

Figure 3: Clustering with Spectral algorithm

Actual examples of parameters perceived from spectral clustering are shown in Table 4. This table also contains sets from level 3 that contained the greatest number of tasks parameters. They would be an example of tasks from medium level of difficulty. Average score for level 0 is 4.776 and for level 9 is 4.840.

Table 4

Spectral clusters examples

Level	Set	Score
0	(-4, 0, 0)	4.778
0	(-9, 0, 0)	4.765
0	(-10, 0, 0)	4.767
3	(1, -8, 0)	4.837
3	(-6, -6, 0)	4.78
3	(-1, -6, 0)	4.81
9	(-1, 6, 7)	4.802

9	(1, 6, 9)	4.864
9	(1, 6, 5)	4.854

Table 4 also shows visual differences between generated difficulty levels. Level 0 contains parameter sets with only one non-zero value, level 3 sets contain one zero parameter, and parameters in level 9 are all non-zero. It shows similarities to how human can see which task is more or less difficult by a number of significant values.

5. Result

The experiment involved testing the proposed algorithm on quadratic equation solving using primitive scoring function based on time and memory consuming on solver execution. Three clustering algorithms were used to divide generated parameter sets into different difficulty levels:

- K-Means method
- Agglomerative clustering
- Spectral clustering

To help understand which algorithm suits better in the experiment conditions, two main characteristics are used:

- How many sets are in each cluster
- Is assigning to one or another cluster reasonable

In addition, each algorithm's disadvantages were taken into account to find out better choice for different use cases.

A bar chart with these amounts is shown in Figure 4:

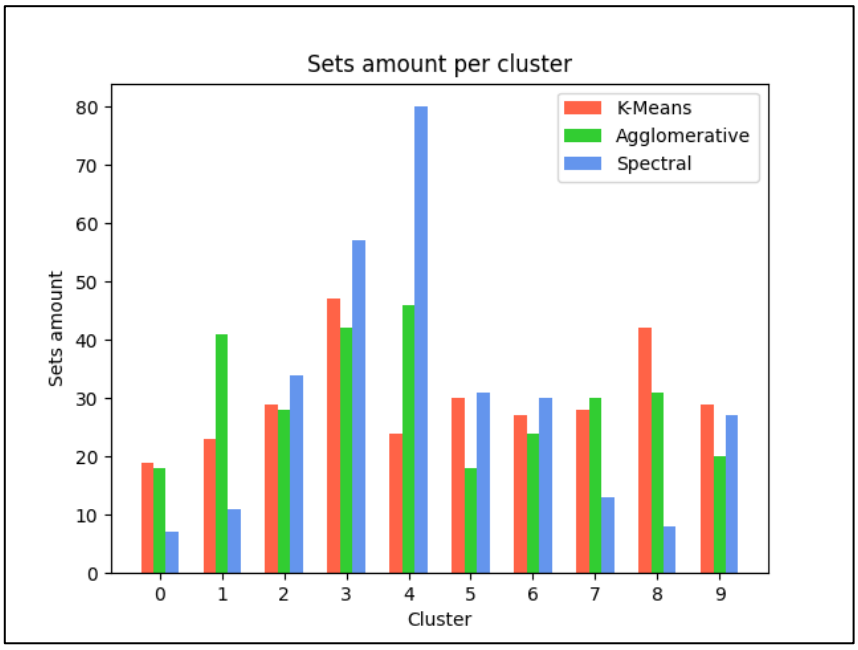

Figure 4: Bar chart of amounts per cluster

Actual amounts of sets per difficulty level for each clustering algorithm are shown in Table 5.

Table 5
Amounts of sets per level

Level	K-Means	Agglomerative	Spectral
0	19	18	7
1	23	41	11
2	29	28	34

3	47	42	57
4	24	46	80
5	30	18	31
6	27	24	30
7	28	30	13
8	42	31	8
9	29	20	27

K-Means and agglomerative clustering has shown similar results in the amount of sets per cluster. But agglomerative algorithm provided clusters that would not fit for real usage of it since parameters in different levels were chaotic.

Spectral clustering shows the best results in both the number of sets per cluster and their actual content in each cluster.

A comparison between these algorithms is presented in Table 6.

Table 6
Comparison of clustering algorithms

Comparison	K-Means	Agglomerative	Spectral
Number of sets per cluster	Similar amounts	Similar amounts	Medium levels contain most of the parameter sets, low and high contain less
Reasonable assigning of a parameter set to a cluster	Average results	Assignment was chaotic	Shown the best results
Disadvantages	Cannot handle outliers and noisy data	Time complex and close pairs can be merged too soon	Computationally expensive in most cases

A disadvantage of the K-Means algorithm in problems with the handling of outliers and noisy data was not experienced in the experiment since all parameters do not differ enough for this. Having the spectral algorithm as the main for dividing into different difficulty levels also cannot always be possible since on a large amount of data it would be expensive. However, in most cases, it is assumed that tasks generation would not have such amounts of data, that is why it can be used as a first priority to choose.

Testing different weights for memory and time scores for each set has not shown any significant differences. Examples of minimum and maximum scores with different weights are shown in Table 7.

Table 7
Scores calculated with different weights

W_{mem}	W_{time}	Minimum score	Maximum score
0.1	0.9	2.078	2.722
0.2	0.8	3.425	4.021
0.3	0.7	4.765	5.320
0.4	0.6	6.106	6.620
0.5	0.5	7.440	7.919
0.6	0.4	8.772	9.218
0.7	0.3	10.104	10.517
0.8	0.2	11.437	11.816
0.9	0.1	12.769	13.116

As can be seen from the table, using only two metrics for scoring function results in randomness in clusters assignment. Delta between maximum and minimum values is always less than one; that is why

all parameter sets would be considered as similar when passed to the clustering function. Then, the most significant influence on the clustering result would be made by parameter values. More metrics should be applied to prevent this, and weights for these metrics should be thoroughly defined.

6. Discussions

Experiment results show that the algorithm needs improvements, especially for making it applicable in different areas of learning. Specific modifications should be made in metrics evaluation and scoring function. Additionally, particular clustering algorithms should be picked up to be used with different parameters sets.

Sets for the experiment were generated by iterating through all possible inputs for each parameter. If the possible number range step is less than 0.01, then the number of sets generated would rise significantly. It can be improved by defining a more accurate way. For example, only part of all sets can be generated from the whole possible range. Then after scoring these sets and clustering them, a new chunk of sets can be generated to fill medium level sets. This generation can be based on ML methods, and new sets would be created using features of medium levels clusters.

An important point that should be taken into account is that the time consumed by the execution of the solver depends on the computer hardware that is running it. That is, in case if time metric is still used, the solver itself should be running on a dedicated machine that would produce similar results, at least in the scope of one problem for which parameters sets are generated.

ML methods can also improve metrics definition and evaluation. Metrics can be calculated not only by a static algorithm but with a model that can predict metric values from the parameters set.

The scoring function itself can also be improved with ML techniques in the same way as metrics. Such a model would have to score the parameters set not by statically retrieved metrics but after learning how several previous sets were scored. However, this learning process should be supervised, that is why it would involve more people, and the purpose of the algorithm creation is to reduce efforts from teachers and course creators.

There are also a lot of possible improvements that can be made with a scoring mechanism like hierarchical scoring for the batch of methods that can solve the initial problem. It depends on the actual area of application and the variety of methods that allow solving the initial task.

The best clustering algorithm in the experiment was the spectral one. It has produced clusters for medium difficulty levels with the most parameters sets that are approximately close to normal distribution. It would fit for real cases the most since low and high levels of difficulty would probably be involved less. Moreover, specific sets in clusters were selected correctly from the student's point of view – the more significant parameters and the lower they are (which are not non-zero ones) in the equation, the more time it would take to solve it.

But, since this specific algorithm has its own disadvantages, like high computational cost on big data amounts, it can be a problem to use it for tasks with many parameters. Clustering methods also have different input parameters for the model. So, tuning the algorithm not only with different methods but also with different parameters could possibly help to find out the best clustering technique for the specific use case.

Furthermore, the proposed algorithm does not describe how exactly the difficulty level would change in the learning process. It can be done by simply solving n tasks at the current difficulty level. However, it can be integrated with knowledge graphs to produce a deeper learning structure. It can include not only the progress of the student in the current difficulty level but also in other related topics. So, the retrieval process of the new parameters sets for several levels can have more input parameters besides the only level value.

Another assumption that can be made for further modifications in the algorithm is to use tree structures instead of clustering to define the process of learning [9]. Starting from easier levels the next parameters of tasks can be marked as child branches of the root element. The difficulty level, in that case, would be the depth of parameters set in the tree structure. In this way, integration with knowledge graphs can also be done. Such a sophisticated structure of learning can be probably hard to implement, but after proper setup, it can be a powerful tool for the automatic and dynamic creation of learning courses for students.

With the results of the experiment, it can be said that the algorithm can already be used for simple tasks set creation. With different levels of difficulty, such sets would have a great influence on students' motivation and, therefore, would result in better outcomes from the learning process itself. On the other side, it would make teachers' work a lot easier when the course assumes a large quantity of practical tasks, especially when the possibility of perceiving the same tasks by different students should be minimized.

But still, teachers and course creators have a significant influence on the result of the generation process. They define a lot of factors that can improve or worsen the result of algorithm work, from the definition of parameters ranges to the determination of the methods that can possibly solve the task. That is why any improvements should be made not only with the algorithm itself but also with training of course creators to define as much data as needed to retrieve the best possible result. However, this is not the point of this paper since the description of how exactly the algorithm would be integrated into learning systems is not the goal of the research.

7. Conclusions

As a conclusion of the research, it can be stated that students' motivation is the key point in their self-organization and, therefore, the learning process. Motivation can be raised by introducing dynamic and hard enough practical tasks in education. Content of these tasks should generate depending on the student's progress: already learned subjects or already completed tasks on the easier difficulty levels.

As a result of the research, an algorithm of dynamic tasks generation grouped by different levels of difficulty is proposed. With specific modifications and improvements, it can be useful for various areas of learning. The basic version of the algorithm was tested in the experiment, which has shown promising results for the simple task of quadratic equation solving. However, at the current stage, it can already be used to create simple tasks like equations presented in the experiment or other mathematical tasks.

After the experiment performed with the prototype implementation of the algorithm, it can be stated that this solution is not ready enough to be used in online courses. The algorithm does not take into account all possible areas of application. It would be comparatively easier to implement this algorithm for using it with known mathematical tasks or equations that are part of biology, physics or chemistry courses. However, it can be hard to find correct algorithm modifications and specify metrics and weights for scoring function when learning languages, for example. To make this algorithm applicable for such an area, methodologies from Natural Language Processing (NLP) can be used both for scoring parameters and implementing the task solver algorithm itself [10].

Metrics presented as an example in this research are predefined by the person who implements the algorithm. Two metrics used in the experiment were obviously insufficient to calculate precise scores of parameters sets. Nevertheless, it still can be a development direction toward using ML in metrics collection. For example, it can be considered that the course creator defines several sets of parameters and determines the values of the specific metric for each set. Then learning algorithms can be applied to teach a model how to evaluate this metric for each generated set.

The scoring function itself can be modified. Instead of the calculating sum of all weighted metrics, it can be any function that takes these metrics as an input and outputs the actual score. It cannot be determined precisely that a specific scoring function would fit for any possible case and it should be tested and modified for each use case.

Another direction for improvement is grouping different methods to solve a problem. For example, quadratic equations can be solved in various ways. Therefore, scoring parameters set on only one solution method cannot fulfil the algorithm's requirements. If solving by one method can be difficult and time-consuming, there still can be another method which would fit more for the specific parameters set. Having a score on a high level that involves scoring each of the known methods on a lower level would provide a more accurate result.

Furthermore, scoring can be made in a tree-like structure where the final score on a parameters set is the root of the tree, and it is calculated using scores in child elements of the root. Each of these child elements would have own scoring method either with single function or accumulation of different scoring functions.

Spectral clustering generated the best results in the experiment. On the other hand, the k-Means method was good enough and agglomerative clustering has generated chaotic sets in difficulty levels on the experimental input data. Further research on different methods should be performed in order to find out which clustering technique suits better for various sets of data, especially since the goal of the algorithm is to make it applicable to different areas of learning, whether it is mathematical problems or language exercises.

Another assumption for further research is that clustering can be substituted by another structure that can lead to more difficult implementation but a lot easier and more comfortable usage by teachers in future. Instead of a simple grouping of parameters sets into clusters, they can possibly be organized in a sophisticated hierarchical structure that would support having a sequence of tasks perceived by a student level by level, including other input parameters such as already learned related topics.

Also, the important point is that when integrating such an algorithm into a learning system, teachers and course creators should be properly educated [11]. It is the goal of the systems supporting the proposed algorithm to provide a complete set of instructions and tutorials.

8. References

[1] V. Dagiene, G. Stupuriene, L. Vinikiene, Implementation of Dynamic Tasks on Informatics and Computational Thinking, Baltic Journal of Modern Computing, Vol. 5, No. 3 (2017), 306-316. doi:10.22364/bjmc.2017.5.3.05.

[2] J. Kormos, J. Wilby, Task Motivation, in: The Palgrave handbook of motivation for language learning, Palgrave Macmillan, Cham, 2019, pp 267-286. doi:10.1007/978-3-030-28380-3.

[3] E. Yilmaz, M. Sahin, M. Turgut, Variables Affecting Student Motivation Based on Academic Publications, Journal of Education and Practice, Vol. 8, № 12 (2017), 112-120.

[4] S. Nawaz, N. Srivastava, J. Hyun Yu, How Difficult is the Task for you? Modelling and Analysis of Students' Task Difficulty Sequences in a Simulation-Based POE Environment, International Journal of Artificial Intelligence in Education, 2021. doi:10.1007/s40593-021-00242-6.

[5] K. Porayska-Pomsta, AI as a Methodology for Supporting Educational Praxis and Teacher Metacognition, International Journal of Artificial Intelligence in Education, Vol. 26, № 2 (2016). doi:10.1007/s40593-016-0101-4.

[6] X. Zou, W. Ma, Z. Ma, R. Baker, Towards Helping Teachers Select Optimal Content for Students, in: Proceedings of the 20th International Conference, AIED 2019, Chicago, IL, USA, 2019, pp. 413-417. doi:10.1007/978-3-030-23207-8_76.

[7] N. Sharonova, I. Kyrychenko, G. Tereshchenko, Application of big data methods in E-learning systems, in: Proceedings of the 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021), 2021, – CEUR-WS, 2021, ISSN 16130073. – Vol. 2870, pp. 1302-1311.

[8] Z. Zhou, S. Liu, Machine Learning, 1st ed., Springer Verlag, Singapore, 2021. doi:10.1007/978-981-15-1967-3.

[9] P. Brass, Advanced Data Structures, 1st ed., Cambridge University Press, New York, NY, 2008.

[10] G. Tereshchenko, I. Gruzdo, Overview and Analysis of Existing Decisions of Determining the Meaning of Text Documents, in: Proceedings of the International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 2019, pp. 645–653, 8632014. doi: 10.1109/INFOCOMMST.2018.8632014.

[11] Z. Dudar. I. Shubin, A. Kozyriev. Individual Training Technology in Distributed Virtual University, Current Trends in Communication and Information Technologies. IPF 2020. Lecture Notes in Networks and Systems, vol 212, pp. 379-399. Springer, Cham. doi:10.1007/978-3-030-76343-5_20.