User-defined Difficulty Levels for Automated Question Generation

Rahul Singhal
School of computing
National Institute of Singapore
Email: nitk.rahul@gmail.com

Martin Henz
School of computing
National Institute of Singapore
Email: henz@comp.nus.edu.sg

Shubham Goyal
Holmusk Pvt. Ltd.
Email: shubhamjigoyal@gmail.com

***Abstract*—We propose a *difficulty model* for generating questions across formal domains according to the difficulty level provided by the user. Our model is interactive and adaptive to user input. The model uses predefined factors for measuring the difficulty and a user defines the difficulty level by ordering these factors. We use lexicographical ordering to compare the difficulty of questions based on a user-defined ordering of factors and a concomitant algorithm for handling these factors. Further, we provide a feature called *scenario guidance*, which allows users to change the scenario at run time. We develop a software using the proposed model, which generates new questions according to a user-defined difficulty level. In order to evaluate the proposed framework, we conducted a pilot test of the software, in which teachers generate questions according to their chosen desired input including the difficulty level. The results show that the system is effective, helpful and robust. Overall, the framework shows promising benefits for teachers and organizations involved in setting questions for standardized tests.**

I. Introduction

Questions of varying difficulty level are required to aid and assess one's mastery of a concept. However, defining the difficulty level of a question is subjective and problematic, since multiple skills such as visualization, analysis, abstraction, deduction, diagrammatic interpretation etc. can be required to solve a question. Mastery of these skills varies from person to person. Given the task of setting a test, different teachers will select a different set of questions of differing difficulty levels because they do not share the same notion of difficulty.

Similarly, a question which is tough for one student may not be tough for another student in the same class, since in a classroom, every student has a different level of understanding a topic. With the limited class-time, a teacher cannot focus on every student's weakness. This gap can be filled by a software that can generate questions according to the difficulty level defined by the student. The software can then act as a personalized instructor and meet the required level of student proficiency. Additionally, such software can generate multiple questions of the same difficulty level based on user given inputs for practice and to prevent cheating in large-scale tests.

Currently, little published work deals with the difficulty of a question, and none with the goal of automatic question generation based on a user-defined difficulty levels, no such software is available. The literature review shows that even the best available systems, such as Andes [1], Chemistry Studio [2], ActiveMath [3], JGEX [4], Geogebra, Cinderella and Sketchpad, are not able to automatically handle difficulty level of a question.

This paper describes a framework to generate questions based on user-defined difficulty levels. We use predefined factors for measuring the difficulty levels and allow users to order these factors. The order will decide the factors to be added/subtracted to modify the difficulty of the next generated question. The new framework is integrated with our previously developed framework [5] for automated questions generation. Overall, the combined framework can generate large number of questions to test concepts in various domain objects, based on the user-defined difficulty levels.

The main contributions of this paper are as follows:

1) We propose a difficulty model for question generation according to user-defined difficulty levels.
2) We integrate the difficulty model with our previous framework to generate geometry questions
3) We evaluate the framework and demonstrate the effectiveness of our question generator. It can generate questions covered in high school textbooks and that can be asked in various competitive tests such as SAT and GMAT, based on user-defined difficulty levels.

II. Related Work

In this section, we first describe the various theories proposed for student's growth in geometry. These theories describe various factors responsible in handling geometry at different levels, which can be used to define the difficulty level of a question. We then deal with the literature regarding automated measurement of the difficulty level of questions.

A. Theories for describing student's growth in the geometry domain

Piaget/Inhelder: Piaget and Inhelder's theory [6] describes the development of the ability to represent space. Representations of space are constructed through the progressive organization of the child's motor and internalized actions, resulting in operational systems [7]. The order of development is seen to be: topological (connectedness, enclosure, and continuity); projective (rectiliniarity); and Euclidean (angularity, parallelism, and distance). They describe a sequence of stages in the development of children's ability to distinguish between shapes when drawing them and can be used to assess

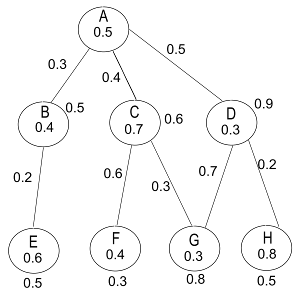


Fig. 1: Difficulty measuring technique in which the difficulty is measured by adding the weights from root to leaf nodes

student development. Hence, geometry questions of increasing difficulty level can contain an increasing number of these elements.

However, this theory has not been widely accepted, since young children may be able to operate with some Euclidean concepts. It seems more likely that topological, projective and Euclidean notions all develop over time and their usage becomes increasingly integrated.

The Van Hiele levels: The Van Hieles proposed a model of geometric thinking which comprises five levels of reasoning -recognition (perception is visual only); analysis (a figure is identified by properties); classification or informal deduction (the significance of the properties is realized); formal deduction (geometric proofs are constructed); and rigour (an aspect later shown to be difficult to operationalize and differentiate from the previous level).

In the van Hiele model the levels were considered to be hierarchical and discontinuous, with each level dependent on mastery of the previous level [8], [7], [9]. However, several papers oppose this theory. Burger [10] proposed that the levels were dynamic and continuous, rather than static and discrete. Clements and Battista (1991) considered that the levels appear to describe students' geometric development. However, they found that students may be difficult to classify, especially in the transition from Level 2 to Level 3 and they also found some evidence for a more basic level than the van Hiele's Level 1 (visual thinking).

B. Automated generation of difficulty levels

An algorithm proposed by Khan [11], [12] used course ontology for representing knowledge. Course ontology is a hierarchical structure where the child of a node is required to understand the knowledge of the parent node. Each node has two values: prerequisite weight and the self weight and the link between the parent-child nodes is assigned a value defined as the link weight (see Figure 1). Prerequisite weight defines the amount of knowledge required from its parent node and self weight is the amount of knowledge intrinsic of the node. Link weight is assigned according to the semantic importance of the child concept to the parent concept. The difficulty of a solution is obtained by adding prerequisite weight, self weight and link weight of all the traversed nodes for finding the solution. However this approach is not applicable in domains such as geometry where a question can have multiple solutions of various difficulty levels depending on the theorems used to solve it. In such cases, assigning the weights for each node and link varies for each solution. Another issue with this mode is that the individual weights may differ from student to student.

Korhan and Rifat [13] developed an intelligent tutor system called MathITS for mathematics education which used concept maps for representing knowledge and a differential equation model to measure difficulty. The system ia based on a dynamic model which depends on whether the question has been solved by the student in the past or not. After each attempt, the difficulty rate of the question changes. The model used by them follows the equation:

$$y'(t) = \left(\frac{b-a}{a+b}\right)y(t)\left[\frac{p-y(t)}{p}\right],\ t > t_o$$

where a and b specifies how many times the question Q_i was answered correctly and incorrectly respectively, t represents the number of attempts made by a student and p is a variable used to stabilize the equation. The question Q_i has been asked to students $a + b$ times, and y represents the new difficulty rate. The approach used in MathITS is suitable for teaching purpose where the difficulty level of a question decreases if a student is unable to solve it. However, the question-bank is manually added in the system. Additionally, there is no provision of configuring the questions according to user-desired difficulty.

In the engineering domain, Rui[14] has defined a framework called Standards, Analysis, Synthesis and Expression (SASE) for question generation and assessment. In this framework, concepts are nodes, which are linked to each other as in concept maps. A table for each node is stored with information on the attributes of the node and its relationship with neighboring nodes. The table stores all the possible logical conditions that can be applied on the node, along with all the possible actions and rules that can be applied to the output of logical conditions. The table decides the next node to follow while generating a question and difficulty is calculated by counting the number of nodes traversed for obtaining a solution. However, this approach depends on the rules in the tables, which are static and may not be suitable for measuring the difficulty of the questions with respect to each user.

Li and Sambasivam [15] developed an intelligent assessment system consisting of three modules: question difficulty assessment, automatic question generator and guided problem solving. The system starts with pre-stored questions and a directed acyclic graph to measure the proficiency level of learners. The degree of difficulty D can be defined as follows:

$$D=w_1\ N+\ w_2\ P+\ w_3\ M$$

where D is the degree of difficulty of a question. N is the number of conditions given in the question. P is the number of downward edges in the path traversed during question generation. M is the number of upward edges in the path traversed during question generation. w_1,w_2,w_3 are the weight factors used to balance between the path length and the number of conditions. The approach is quite generic but they have predefined set of questions and there is no provision of configuring the difficulty level of questions by the user..

C. Summary/Gap

The theories described for student's growth in geometry may not be applied to other domains such as algebra. Further, all the methods described so far for automated measurement of difficulty levels either used predefined static weights assigned to each concept node or decided difficulty on the basis of total number of nodes traveled. However, these methods are not applicable in domains such as geometry where multiple solutions can exist for the same question and each solution uses different concepts. In such cases, the weight assigned to the concept should differ in the situation when user intentionally wants that concept to be tested as compared to that of situation where the concept is not required by the user but it is a prerequisite for the solution. Additionally, a user cannot configure the generated question according to his desired difficulty levels.

We adopt the model by Li and Sambasivam [15] and propose a framework which is dynamic and therefore adapts to the user-desired difficulty level while generating the questions. Additionally, the users can configure the factors responsible for the difficulty of the generated question until the desired question is generated. The next section describes the difficulty model.

III. Generic Difficulty Model

Difficulty is a subjective matter and therefore it is hard to define a notion of difficulty which is valid for all users. Hence, we propose a generic model of handling difficulty while generating question which is adaptive to the users. Questions are generated based on user's given input and desired-difficulty levels. The main properties of the model are as follows

- The model includes the factors affecting the difficulty of a question.
- The model allows the user to define their own difficulty levels by assigning the order of these factors. Additionally, the user can modify the order of the factors later until the question of desired difficulty levels is generated.
- The model adapts to the ordering of the factors given by the user.

The factors responsible for measuring the difficulty of a question can be classified into three categories–*Affecting Semantic Relationships (SR), Affecting Quantitative Relationships(QR) and Affecting relationships derived from meta-information (DR)*. The next section describes these factors in detail. We explain the model using the domain of mechanics (high school physics) as example.

IV. Factors affecting the difficulty of a question

Factors affecting the semantic relationships (SR)

SR deals with the ways in which a domain objects interact with each other. It refers to the non-numeric and logical relationship between the domain-objects developed due to their interaction. For example, in mechanics, two blocks connected by a massless string will have the same acceleration if the string is taut. The major factors in this category are as follows

1) Number and type of domain-objects involved
2) Number and type of domain-rules involved
3) User given scenarios such as Figure 4a.

Factors affecting the quantitative relationships (QR)

QR refers to assignment of numeric values to the properties of the domain-objects, for example, the acceleration of blocks and pulleys, masses of blocks and pulleys in a question.

Factors derived from the solution (DR)

DR refers to the factors which affect the difficulty of a question using the information in the generated solution. For instance, the number of steps in a generated solution can be an important factor for difficulty in a domain. Some of the factors which falls in this category are:

1) Length of the solution
2) Direct/Indirect use of rules involved

Example of a direct use of rule is direct application of an equation where all variables except one is available. One instance of indirect use of rules in mechanics is the use of solving linear equations for finding values of the variables.

The model analyzes and adapts to all SR, QR and DR factors. The framework uses these factors in an ordered way to generate domain questions. The ordering plays a significant role in the difficulty of the generated question and the next section describes the ordering in detail.

V. Ordering theory

We compare two questions and used lexicographical ordering of the factors used for generating them. For example, if the factors "F1" and "F2" are used for the generation of two questions, then the lexicographical ordering between the two questions are as follows

$$\begin{gathered}(F1_1, F2_1) \prec (F1_2, F2_2) \\ \textit{iff} \\ (F1_1 < F1_2) \qquad \textit{or} \\ (F1_1 = F1_2 \qquad \textit{and} \qquad F2_1 \leq F2_2)\end{gathered} \tag{1}$$

Equation 1 implies that a question having a bigger factor "F1" is considered harder as compared to other question. However, if the factors "F1" are same then the question having a bigger factor "F2" is considered harder.

$$\begin{gathered}(F1_1, F2_1) \equiv (F1_2, F2_2) \\ \textit{iff} \\ (F1_1 = F1_2 \qquad \textit{and} \qquad F2_1 = F2_2)\end{gathered} \tag{2}$$

Equation 2 shows the lexicographical ordering for the similar questions where the two questions are similar if both the factors "F1" and "F2" are equal to their respective factors in other priority list. In order to use the lexicographical ordering for quantitative comparison, we need to show that this ordering can be represented as ordinal representation, which is discussed in the next section.

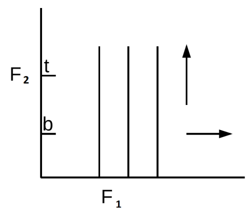


Fig. 2: Showing the ordering representation between two factors F_1 and F_2

Proof of Ordinal representation

Theorem 1. *If the domain is countable, a lexicographical ordering between the domain-elements can be represented in the form of ordinal representation.*

Proof of a domain countability: In a formal domain, objects can be considered as countable and the scenarios are generated by the intersection of the domain objects. Intersection of countable objects results in the generation of countable sets, hence scenarios are also countable. In addition, meta information of the objects is generated through algorithm's application on the generated scenarios and hence they also become countable.

Proof of ordinal representation: If Domain = $F_1 \times F_2$, Equation 1 defines a lexicographical relation between F_1 and F_2 and its utility representation is as shown in Figure 2, then we can say that the right side has better points, but if two points are equally far to the right, then the top point is better. If we choose two distinct points on each vertical line and suppose there is a utility representation U, then the top point t_x on the line with first coordinate x must map to a higher number than the bottom point b_x on that line. Now consider the collection of intervals $\{[U(b_x), U(t_x)] \ : x \geq 0\}$. These intervals are all disjoint. and non-degenerate, hence each contains a rational number. These rational numbers are all distinct, and we have one for each vertical line, so if a utility function exists, there must exist an countable collection of rational numbers. A scenario is a collection of countable points; hence, the rationals are countable. So U must exist for scenarios.

In addition, ordinal (numerical) representation maps the ordering of the factors with the real numbers and hence can be used in measuring the difficulty level.

We represent the ordering between the factors given by the user using a data structure called priority list described in detail in the next section.

User-priority list

User-priority list is a list of *key-value* pairs of factors that can affect a scenario and a question. The key refers to the pre-defined factors affecting the difficulty of a question and the value refers to the number given to it by the user. The factors belong to either SR, QR or DR. For example, a user-priority list in mechanics may be

Number of Blocks–10, Number of Pulley–8, Acceleration of at-least one block–7.

The numbers assigned to each object describe its priority. The next section explains the processing of user-priority list in detail.

Processing User-priority list in handling difficulty factor

Ordering refers to priority assignment to the items specified in the user-priority list. The user needs to assign numbers to the factors, which he wants to see in the generated question. The assigned number to each factor is directly proportional to the priority of the factor in generating a question. To generate a question of harder and easier difficulty level, higher value of a factor implies higher and lower occurrence of that factor, respectively. Further, demand of higher and lower complexity in a question involves addition and reduction of factors having higher priorities, respectively. The priority of the factors in the user-priority list mentioned in the last section is as follows

Acceleration of a block $\prec$ Number of Pulley $\prec$ Number of Blocks

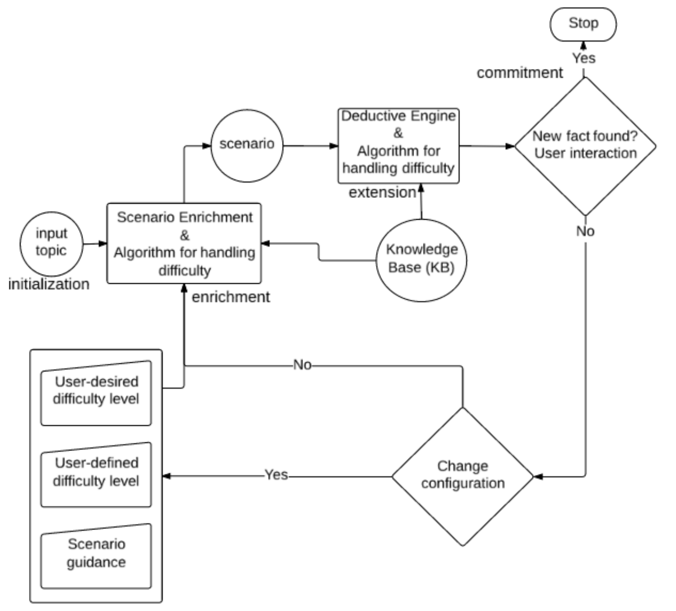
where $\prec$ denotes the priority symbol. The above expression implies that the priority of blocks is higher than the pulleys, which has higher priority than the constraint of acceleration of a block.

Summing up these numbers will give the difficulty level of a generated question. A larger sum implies higher difficulty level of the generated question. For instance, if the sum of assigned numbers of the factors in the generated question is 25 then a harder question would have sum higher than 25. The cause can be addition of any item from blocks, pulleys or block's acceleration. Similar logic applies for generating *easier and similar* questions. However, generating similar question would have an extra constraint of each factor having the same difficulty points. For example, the two questions are similar if they have equal number of blocks, pulleys and both scenarios have accelerating blocks. Generally these figures are isomorphic.

VI. User interaction with the model

The user plays a crucial role in deciding the difficulty of a generated question and this section describes the ways in which a user can interact with the framework.

1) **Specification of the input required for question generation**: Mathematically, an input for a geometry question generated by the framework can be represented by a triple (Object $\mathcal{V}$, Rule $\mathcal{R}$, Query type Qt). The input plays a major role in deciding the difficulty as priority of rules changes with respect to the objects used. For instance, in mechanics, testing accelerating block concept is different than testing with accelerating pulleys and wedges.
2) **User-defined difficulty level**: The process of allowing the user to generate the ordering between the factors defined in user-priority list is termed as *user-defined difficulty level.*



Stop
Yes
commitment
Deductive Engine & Algorithm for handling difficulty
extension
scenario
New fact found? User interaction
No
input topic
initialization
Scenario Enrichment & Algorithm for handling difficulty
enrichment
Knowledge Base (KB)
No
User-desired difficulty level
User-defined difficulty level
Scenario guidance
Yes
Change configuration

Fig. 3: Flowchart for handling adaptability to difficulty factors across formal domains

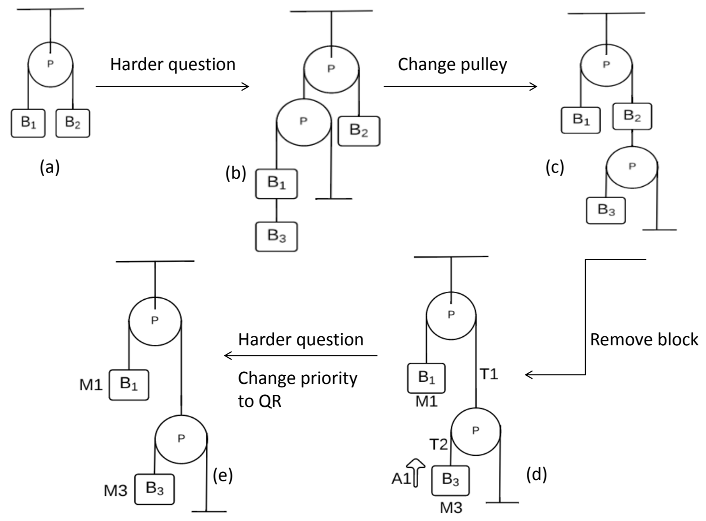


Fig. 4: Working of the difficulty model with an example (a) generation of a mechanics scenario from the user-given input; (b) addition of pulley and block to make the question harder; (c) change the position of a pulley *P* via scenario guidance; (d) removal of a block to make the question easy; (e) changing the priority with QR as highest priority and hence removing the value of forces in the rope and acceleration of a block to make the question harder

3) **User-desired difficulty**: Additionally, the user selects the difficulty direction of the next generated question from the following options
   a) Easier than the given question
   b) Similar to the given question
   c) Harder than the given question
4) **Scenario guidance**: The process involves modifying the scenario at run-time. If the users are not satisfied with the interaction and properties of the objects in the current question then they can change them according to their choice. Figure 4b and c explains this feature with an example. The user changes the position of the pulley *P*. Additionally, the user can add the objects from the user-priority list which are not present in the currently generated question.
5) **Algorithm termination**: The users can stop the working of the framework if they are satisfied with the generated question.

The users can change these options multiple times based on the generated question until they are satisfied. The next section describes the framework used to adapt to the difficulty factors.

VII. Framework

The proposed framework is an extension of the framework mentioned in [5]. Figure 3 shows a flow diagram of the proposed difficulty model for adaptability to the difficulty factors. The previous framework generates a question from the user-given input. If the user is not satisfied with the difficulty level of the generated question, the previous framework randomly enriches the scenario and generates a new question, which may not be desirable to the user. However, the new framework provides the user with various options, which can lead to next generated question according to user's choice. The user can change the user-desired difficulty level such as higher, similar or harder for the next question. In addition, the user can change the priority-list of the factors and the framework uses the new priority list for the next question. Furthermore, the user can change the scenario in the current question (see scenario-guidance) and the framework generates the next question based on the new current scenario. This process of user interaction and new question generation continues until a satisfactory question is generated. In this way the difficulty model is user-adaptable. The details of user-interaction are given in Section VI.

Figure 4 explains the framework with the help of an example. The input for generating a question in mechanics is as follows

- $\mathcal{V}$: a pulley and two blocks
- $\mathcal{R}$: At least one block is given an acceleration

The framework generates a question from the input given by the user (see Figure 4a). The user is not satisfied with the scenario in the newly generated question and orders the system to generate a harder question. The framework picks the highest priority factor from the user-priority list and adds a pulley and a block (see Figure 4b). In the newly generated scenario, the user changes the position of a pulley *P* through scenario guidance feature (see Figure 4c). Next, the user requests the generation of an easier question, compared to the previously generated question. As a result, the framework removes a block (*B2*) (highest priority factor in user-priority list) from the existing scenario to make the scenario easy, and generates a new question (see Figure 4d). Again, the user is not satisfied and changes the ordering of the predefined factors in the user-

Difficulty level	Affecting SR	Affecting both QR and DR
Easier than a given question	Reduce the number of occurrences of the selected item by 1	Give more data to result in smaller number of steps
Similar to a given question	Change the location of occurrences of the selected item	Interchange the data with the help of pattern matching
Harder than a given question	Increase the number of occurrences of selected item by 1	Reduce the given data or indirectly give the data requiring a larger number of steps

TABLE I: Pre-defined algorithms for changing the difficulty of the generated question

priority list, this time, giving priority to the QR relationship to generate a harder question. Hence, the generated question has the same scenario but less number of quantitative relationships by removing the value of forces acting on the ropes and acceleration of a block making the question more difficult(see Figure 4e). Finally, the user is satisfied with the generated question and the whole process stops.

A. *Integration of Difficulty model with the existing framework*

The proposed framework comprises two major components—scenario enrichment and deductive engine. Each component depends on various user-desired features for their functions. For instance, scenario enrichment will enrich the scenario depending on the user's choice with a higher number of objects or a lesser number of objects with more number of known relationships between them. Similarly, the deductive engine requires to know that what rules a user wants to test. The details of the framework without the logic of handling difficulty level are given in [5].

The key changes for integration of the difficulty model involves ordering of the factors required in decisions making in both the components. Previously, the framework used random selection of adding/removing domain objects and rules at various decision points. Currently, the framework uses the user-priority list. The list contains the ordering of the factors and helps in dealing with decision points. The next subsection describes the details of extension in each component of the framework to integrate with the difficulty model.

VIII. Scenario enrichment for handling difficulty

This component enriches the scenario. It operates at two levels–with or without an existing scenario. The details are as follows.

Generating scenarios from the user-input (Without-scenario): This component generates a scenario through a set of predefined ways to combine the domain objects. The process includes addition and removal of various factors such as objects and rules, to the existing scenario. The details of the component without the logic of handling the difficulty concept is given in [5].

The component deals with the ordering of the SR factors such as number and type of domain objects and rules to modify the current scenario. Previously the ordering was random every time the framework generated a question. However, now the framework uses user-defined difficulty levels for deciding the order of the factors and user-desired difficulty levels for application of these factors in an existing scenario. The second column in Table I shows the pre-defined algorithms for handling the semantic relationships. For each factor, making a question harder/easier implies adding/removing that factor in the existing scenario. Consider an instance where the order of the SR factors is as follows

Number of Wedges $\prec$ Number of Pulleys $\prec$ Number of Blocks

The above order shows that a user has given highest priority to blocks followed by pulleys and wedges are given least priority. For generating the harder question, the framework attempts to increase the blocks demanded by the user. If no more blocks can be added then the framework approaches pulleys followed by wedges. Similarly, for generating an easier question, firstly, the framework tries to reduce the number of blocks followed by pulleys and wedges. For generating a similar question, the framework tries to change the positions of blocks followed by pulleys and wedges.

Generating new scenarios from an existing scenario: This component is optional and is used when the user is not satisfied with the difficulty level of the generated question and wants to enrich it. The component enriches a scenario by assigning numeric values to the existing relationships in the generated scenario. This component along with the Deductive engine handle the quantitative relationships (QR) demanded by the user. Previously, the enrichment in QR was done randomly. However, currently, user-defined difficulty levels guide the enrichment process. The third column in Table I shows the pre-defined algorithms for handling quantitative relationships. The approach is similar to the scenario enrichment component described earlier, the difference being that this component is directed towards the properties of the existing domains objects in the generated scenario. The details of this component without the logic of handling the difficulty concept is given in [5].

A. *Deductive Engine for handling difficulty*

This component is responsible for finding the values of unknown variables of the generated scenario from the scenario enrichment component. The component acts as both a question generator and a solver. The unknown variables whose values have been found represent the generated questions. The steps that leads to finding the unknown variables represent the solution. The component handles the DR factors and assists the scenario enrichment component in handling the difficulty level. It acts as a filter showing only those questions which suits to the DR factors defined by the user-priority list. The details of this component without the difficulty handling logic part is given in [5].

B. *Implementation*

We employ SWI-Prolog (Version 7.1.2) [16] as deductive engine and represent axioms using Constraint Handling Rules

(CHR) [17] via this system's multi-head clause format. In our implementation, we use the CHR library provided by K. U. Leuven, on top of SWI-Prolog. A domain may require solving of linear equations, for which we employ the SWI-Prolog library CLP(R).

Evaluation

For evaluating the difficulty model, we developed a prototype of the model in Geometry, called "Euclid's Toolkit", and tested it with real users such as high school teachers, students and professionals who tutor for standardized tests such as CAT, GRE, GMAT, SAT etc. The knowledge database consists of triangles and concepts include perpendicular, median and parallel lines. Each concept has a corresponding theorem, which the user can select. Currently, the users can choose any one theorem from the options: Pythagoras theorem, Apollonius theorem and Similarity theorem.

Euclid's Toolkit provides a method for handling the difficulty level of a question. The factors which affect the difficulty of the question include the 3 concepts and 3 theorems. The user can assign priority to these factors by assigning them numeric values between 0 to 6.

User feedback for Euclid's Toolkit

Re-evaluation of Euclid's Toolkit is ongoing. So far it has been evaluated by 10 teachers with different educational backgrounds: 3 high school teachers, 4 private tutors for high school and 3 private tutors for standardized tests such as CAT, GRE and GMAT. The teachers used the Euclid's Toolkit for 1-2 hours and the filled an evaluation survey. The survey form consists of 40 questions covering all features of the software such as quality of the generated questions, difficulty factors, usefulness of the scenario guidance feature in handling the difficulty, etc.

The feedback was positive with all the teachers being either very satisfied or satisfied with the quality of of generated questions, the scenario guidance feature and the handling of difficulty factors. Feedback included wanting more control over the concepts addition process. Currently, the toolkit provides the options to select a triangle from the list of the generated triangles. It doesn't gives the exact point at which concept can be added. Also, one teacher comment that they spent time classifying the generated question and this could be automated.

If made available to them, all the teachers would like to use Euclid's Toolkit for teaching and 80 % of them would like to use it for student assessment (see Figure 6). Two teachers criticized that complex questions (involving the use of multiple theorems) were generated only later on adding difficulty instead of at the first generated question. They wanted to have control over this process. Further, teachers were asked to assess Euclid's Toolkit in comparison to existing textbooks and online materials in terms of practice questions and new questions. 80% teachers rated Euclid's Toolkit higher than existing textbooks and online material for these parameters. (see Figures 7 and 8).

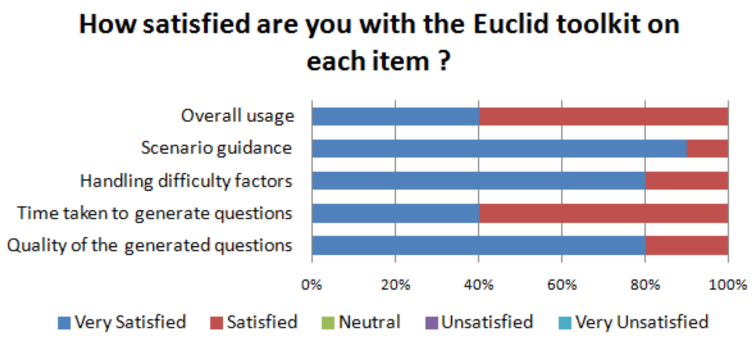


Fig. 5: Evaluation of Euclid's Toolkit: How satisfied are the users with the features of Euclid's Toolkit.

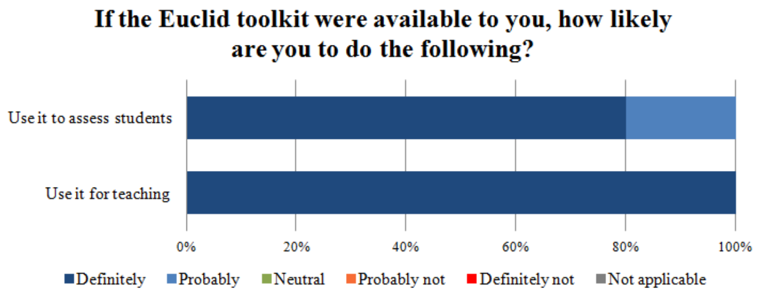


Fig. 6: Evaluation of Euclid's Toolkit: Its usability for teaching and testing the students.

The users liked the scenario guidance feature and being able to handle difficulty of geometry questions. They assessed that these features are useful for generating questions relevant for high school students. They also evaluated this toolkit as better than books and online material with respect to new questions and highly recommend it for assessing students.

Other suggestions from them included the addition of more objects such as circle, square, etc., concepts and geometry theorems. Also, generating questions with length and angle values in terms of variables such as "x_1" is uncommon for high school mathematics. They suggested the use of numeric values would be appropriate to teach and test students. Another suggestion was to generate multiple choice options so that this software can directly be used for standardized tests such as CAT, GRE and GMAT. However, these suggestions/criticism is out of the scope of the research.

IX. Conclusion

In this paper, we provide a framework for the automatic generation of questions and their solution across formal domains based on the user-defined difficulty level. Our system is able to quickly generate large numbers of questions on specific topics and of varying difficulty levels. Such a system will help teachers reduce the time and effort spent on the tedious and error-prone task of generating questions of various difficulty levels.

Future work needs to be done in testing the framework rigorously with teachers across different countries. Further, in the existing system, a user needs to feed his difficulty

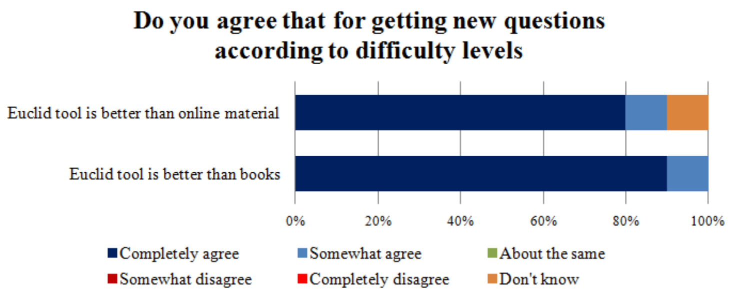


Fig. 7: Evaluation of Euclid's Toolkit: Usefulness as resource for practice questions in comparison to existing textbooks and online material.

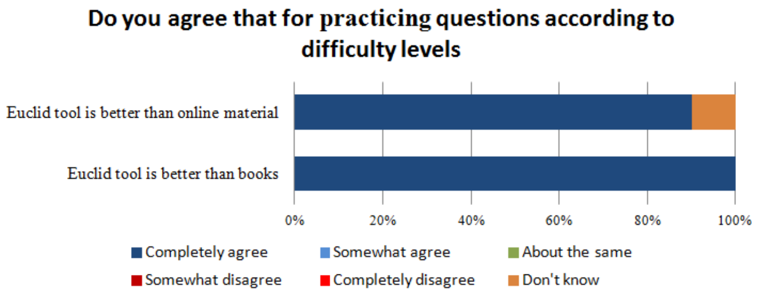


Fig. 8: Evaluation of Euclid's Toolkit: Usefulness as resource for new questions in comparison to existing textbooks and online material.

choices before generating a question and alters the difficulty options until his desired question gets generated. Future work involves automating this process through the use of machine learning. The machine would learn the difficulty level of each user during the process of question generation and apply this learning to generate new questions according to desired difficulty level automatically thus minimizing tedious user intervention.

REFERENCES

[1] K. Vanlehn, C. Lynch, K. Schulze, J. A. Shapiro, R. H. Shelby, L. Taylor, D. J. Treacy, A. Weinstein, and M. C. Wintersgill, "The Andes physics tutoring system—Five years of evaluations," in *Proceedings of the Artificial Intelligence in Education Conference*, ser. Frontiers in Artificial Intelligence and Applications, vol. 125, 2005, pp. 678–685.

[2] "Chemistry studio: An intelligent tutoring system," Jan. 2013. [Online]. Available: http://www.cs.stanford.edu/people/ashgup/Reports/BTP.pdf

[3] E. Melis and J. Siekmann, "ActiveMath—An Intelligent Tutoring System for Mathematics," in *Artificial Intelligence and Soft Computing—ICAISC 2004*, ser. Lecture Notes in Artificial Intelligence, L. R. et al., Ed. Springer, 2004, no. 3070, pp. 91–101.

[4] X. S. Gao and Q. Lin, "MMP/Geometer—a software package for automated geometric reasoning," in *Automated Deduction in Geometry*, ser. Lecture Notes in Artificial Intelligence, F. Winkler, Ed. Springer, 2004, no. 2930, pp. 44–66.

[5] R. Singhal, M. Henz, and S. Goyal, "A framework for automated generation of questions across formal domains," in *In Proceedings of 17th International Conference on Artificial Intelligence in Education*, F. E. Browder, Ed. American Mathematical Society, 2015, pp. 147–240.

[6] J. Piaget, B. Inhelder, F. J. Langdon, and J. L. Lunzer, "The child's conception of space," in *British Journal of Educational Studies*, vol. 5. Taylor & Francis, Ltd., 1957, pp. 187–189.

[7] D. Clements and M. Battista, "Geometry and spatial reasoning," in *Handbook of research on mathematics teaching and learning*, 1992, pp. 420–464.

[8] M. C. Afonso, M. Camacho, and M. M. Socas, "Teacher profile in the geometry curriculum based on the Van Hiele theory." in *Proceedings of the 23rd PME International Conference*, vol. 2, 1999.

[9] C. Lawrie, "An alternative assessment: The Gutiérrez, Jaime and Fortuny technique." in *Proceedings of the 22nd PME International Conference*, A. Olivier and K. Newstead, Eds., vol. 3, 1998, pp. 175–182.

[10] W. Burger and J. Shaughnessy, "Characterising the Van Hiele levels of development in geometry." in *Journal for Research in Mathematics Education*, vol. 17, 1986, pp. 31–48.

[11] J. Khan and M. Hardas, "A technique for representing course knowledge using ontologies and assessing test problems," in *Advances in Intelligent Web Mastering*, ser. Advances in Soft Computing, Wegrzyn-Wolska, M. Katarzyna, and P. Szczepaniak, Eds. Springer Berlin Heidelberg, 2007, vol. 43, pp. 174–179.

[12] J. I. Khan, M. Hardas, and Y. Ma, "A study of problem difficulty evaluation for semantic network ontology based intelligent courseware sharing," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, ser. WI '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 426–429.

[13] G. Korhan and A. Rifat, "Determining difficulty of questions in intelligent tutoring systems," *Turkish Online Journal of Educational Technology*, vol. 8(3), pp. 14–21, 2009.

[14] Y. Rui, "Problem generator system in engineering design tutor," Master's thesis, Rice University, 2002.

[15] T. Li and S. Sambasivam, "Automatically generating questions in multiple variables for intelligent tutoring," *Issues in Informing Science & Information Technology*, vol. 2, pp. 471–478, 2005.

[16] T. Schrijvers and B. Demoen, "The K. U. Leuven CHR System—implementation and application," in *First Workshop on Constraint Handling Rules—Selected Contributions*, 2004, pp. 1–5.

[17] T. Frühwirth and F. Raiser, Eds., *Constraint Handling Rules: Compilation, Execution, and Analysis*. Books On Demand, 2011.