Automatic Exercise Generation in Euclidean Geometry

Andreas Papasalouros

Automatic Exercise Generation in Euclidean Geometry

Andreas Papasalouros

Department of Mathematics, University of the Aegean, 83200 Karlovassi, Greece

Abstract. Automatic assessment has recently drawn the efforts of researchers in a number of fields. While most available approaches deal with the construction of question items that assess factual and conceptual knowledge, this paper presents a method and a tool for generating questions assessing procedural knowledge, in the form of simple proof problems in the domain of the Euclidean Geometry. The method is based on rules defined as Horn clauses. The method enumerates candidate problems and certain techniques are proposed for selecting interesting problems. With certain adaptations, the method is possible to be applied in other knowledge domains as well.

Keywords: exercise generation, knowledge representation.

1 Introduction

Automatic assessment has recently drawn the efforts of researchers in a number of fields including Education Research, Cognitive Psychology and Artificial Intelligence. Automated assessment has a number of potential benefits:

- It facilitates the process of creating question repositories.
- It may support the personalization of questions generated based on student profile and personal learning goals.
- It may become a component of Intelligent Tutoring Systems, where already stored domain knowledge and pedagogic strategies may become the basis for automatic problem construction.
- It can become a crucial component in the process of automation of instructional design.

In this work we envisage a knowledge-based framework where subject domain knowledge will be combined with pedagogic strategies and a knowledge-based description of learning goals, which eventually will drive the whole instructional design process in the sense of generating content knowledge descriptions, examples, and (self-)assessment.

Automatic assessment mainly focuses on generating questions that assess factual or conceptual knowledge. Nevertheless, not much work has been conducted in the direction towards automatic assessment of problem solving skills. The present work aims at generating assessment items for problem solving in the

form of *proof* exercises in Euclidean Geometry. The domain knowledge that is used is considered to be stored into a knowledge base in the form of *rules* [1] that are used by the learner during the solving process. Problems are stated as a set of *assumptions* (givens) and a *statement* (goal) to be derived from the givens. In the case of Geometry proof problems, which we are dealing with in this paper, assumptions and statement to be proven are related through geometric relationships among geometric elements, i.e. segment lengths and angle sizes, possibly with an associated diagram of the geometric setting under consideration [2].

A student solver is anticipated to successively apply selected rules which are stored in the knowledge base in order to prove the goal statement. These rules correspond to certain kinds of mathematical knowledge: axioms, theorems and definitions [3]. Student knowledge representation with rules is pedagogically sound. For example, Scandura [1] suggests that both lower and higher order knowledge is described in terms of Post-like production rules.

Research in problem solving, either in general or in the field of Mathematics [4, 3], has shown that problem solving skills involve understanding and the application of meta-cognitive and strategic knowledge. This work aims at generating meaningful problems that assess basic understanding and rule application, while the assessment of strategic knowledge is left as a future work.

Furthermore, generated problems are intended to assess *understanding* rather than engage students in a mechanical trial and error procedure for reaching the intended goals. Thus, the presented method has a *cognitive* [5] perspective for the assessment of students. Although exercises generated with the proposed method are simple enough, they are anticipated to *assimilate* exercises that experienced teachers should either generate themselves, or select for application in real educational settings. Although the domain of generated problems is restricted to Euclidean Geometry, the presented method may potentially be applied in other areas of Mathematics, as well in non-mathematical fields.

The structure of this paper is as follows: The relation of the proposed method with other methods in the literature follows in Section 2. Section 3 presents the theoretical background of the proposed method. Section 4 describes the algorithm for problem generation and its implementation, followed by an evaluation in Section 5. The paper ends with some conclusions, focusing on the generalization of the proposed method in other domains.

2 Related work

Heeren et al. [6] have developed a methodology for describing problem solutions and feedback strategies based on functional programming. The methodology applies in educational problems that engage algorithmic solutions, e.g. matrix manipulation, finding roots of equations, algebraic manipulation of expressions. This methodology can be used in a reverse manner,that is, certain descriptions of problems can be instantiated with particular values, thus, yielding new problems that assess the application of algorithms under consideration. However, this approach is not appropriate for non-routine problems [7], such as the proof ex-

ercises in Euclidean Geometry, where no specific algorithm for problem solution is known by the learner.

Holohan et al. [8] describe a method for exercise generation in the domain of database programming. This method is based on an ontology that describes database schemata, which serves as input for generating exercises asking students to form queries based on textual query descriptions. The proposed method can generate exercises from domain specific ontologies, however it is not clear how declarative knowledge described in various ontologies can serve as input for problem generation in a uniform fashion across domains.

Williams [9] proposes the use of domain ontologies for generating mathematical word problems for overcoming the so-called question bottleneck in Intelligent Tutoring Systems (ITS). The semantics of OWL Semantic Web language are utilized. The proposed approach is based on Natural Language Generation (NLG) techniques and aims at exploiting existing Semantic Web knowledge bases in the form of ontologies and linked data. The difficulty of the questions can be specified, based on specific factors such as question length, the existence of distracting information, etc. The approach aims at generating meaningful questions.

The work in [10] describes the generation of multiple choice assessment items for assessing analogical reasoning. Again, domain ontologies are used as input for question generation. Analogies in questions are extracted by identifying certain structural relationships between concepts in knowledge bases in the form of OWL ontologies. Different levels of analogy are defined for extracting correct (key) and false items (distractors).

In [11] a question generation component of an ITS is described. Multiple choice questions are generated by utilizing OWL semantic relationships, subsumption, object /datatype properties and class/individual relationships. The presented method is based on a set of templates in two levels: At the semantic level, implemented as SPARQL queries, and at the syntactic /sentence realization level, implemented as XSL Transformations.

Papasalouros et al. [12] propose a number of strategies for multiple choice question generation, based on OWL semantic relationships. Besides text questions, the authors provide strategies for generating media questions, demonstrating their approach with image hot spot questions. Simple NLG techniques are utilized, so that questions are not always grammatically correct.

The above-mentioned approaches use ontologies for generating questions for mostly assessing declarative knowledge. Declarative (or conceptual) knowledge assessment deals with checking for relationships and meaning,thus ontologies, as formal expressions of semantic networks, are appropriate for expressing knowledge to be assessed. From the above approaches, only the work in [9] deals with problem generation, albeit relatively simple word problems. According to the typology of problem solving proposed in [7], word problems (named story problems) are considered as a different category than the so called 'rule-using problems', which are actually tackled in current work. Current work aims at non trivial problems [13] that assess procedural knowledge in the form of proofs that employ specific execution steps.

3 Theoretical background

Problem solving has been a field of intensive study in both AI and Cognitive Psychology. Thus, a set of well established principles has been proposed. According to [4]

- problem solving is considered as searching into a problem space;
- a problem space refers to the solver's representation of a task. It consists of
  - set of knowledge states (initial, goal, intermediate),
  - a set of operators for moving from one state to another and
  - local information about the path one is taking through the state

Procedural knowledge, such as problem solving skills, can be modelled by a *production system* with three kinds of productions (knowledge)[14, 1]:

- propositions,e.g. knowledge of rules, etc. that can be applied to a particular situation towards problem solution;
- pattern recognition (matching), in which a particular rule is correctly applied to a given situation yielding a new situation, that is, a new state in the problem space;
- strategic knowledge, which guides the process of rule application through certain heuristics, or engages higher order procedures, such as scripts, that is, proper encodings of already known solutions to intermediate sub-problems.

Thus, given the operators for moving from one state to another, that is, the domain-specific rules, a problem can be defined by identifying the initial and goal states in the above sense. Then, the role of problem solvers is to construct their own path into the problem space, towards the solution of the problem.

Proof exercises in Euclidean Geometry, as well in any other domain, is a kind of procedural knowledge. An exercise can be described through a proof tree, such as the one depicted in Fig. 1 for proving the congruency of two segments. The role of the learner is to correctly apply specific rules, in the form of axioms and theorems, in order to construct the proof. Thus, in the case of proofs such as the above, each state in the problem space is a tentative form of the proof tree, the goal state being the proof tree under consideration, or any tree that proves the statement under consideration under the given sentences.

As an example, we consider a knowledge base on Euclidean Geometry containing the following simple rules, properly encoded:

$$\left.\begin{array}{r} \angle ABC \cong \angle DEF \\ \overline{AB} = \overline{DE} \\ \overline{BC} = \overline{EF} \end{array}\right\} \Rightarrow \triangle ABC \cong \triangle DEF \quad \text{(Side-Angle-Side rule)}$$

$$\triangle ABC \cong \triangle DEF \Rightarrow \overline{AB} = \overline{DE} \qquad \text{(Triangle congruency)}$$

$$\angle ABC \cong \angle ACB \Rightarrow \overline{AB} = \overline{AC} \qquad \text{(Equilateral triangles)}$$

Then, a proof tree, concerning the above rules is presented in Fig. 1 together with the corresponding diagram. In the following section, a method for generating exercise of this kind will be presented.

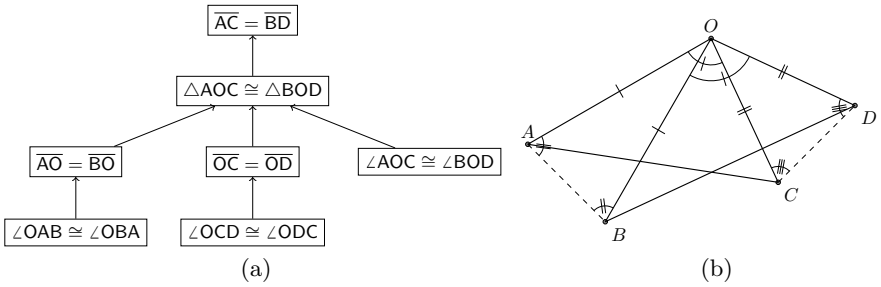

Fig. 1. An example of a generated proof tree (a) and corresponding diagram (b).

4 Problem generation and ranking

4.1 Representation of domain knowledge

We consider a knowledge base that contains a set C of Horn clauses. Each clause has the form

$$b_1, \ldots, b_n \Rightarrow h,$$

where literal h is the head of the expression,

All literals $b_1, \ldots, b_n, h$ denote relationships between geometry elements, such as congruence, in first order logic.

The following geometry elements are considered: Segments, angles and triangles. A literal example is the following expression:

$$cong(triangle(A, B, C), triangle(D, E, F))$$

In the above, *cong* is a predicate denoting congruence, $triangle(A, B, C)$ is a term referring to a triangle while $A, B, \ldots, F$ are variables denoting points. The predicate denotes the relationship $\triangle ABC \cong \triangle DEF$.

The root of the proof tree, as depicted in Fig. 1, is the head of some clause $c \in C$. Each node in the tree represents a literal. Every node in the tree represents a head of some clause in the knowledge base. More specifically, for every internal node, t there exists a clause, c in the knowledge base such that the head, h, unifies with t and each child of t unifies with a corresponding clause in the body of c.

4.2 Problem generation

We present an algorithm that generates trees representing problems. The aim of the algorithm is, given a set of clauses, the enumeration of all *proof* trees of a given maximum depth. The algorithm is presented below.

Algorithm 1 generates all proof trees that can be created by the set C of clauses up to a given depth, namely Max, that satisfy the given constraints. Each literal, L, constitutes a node in the generated proof tree. The above recursive

Algorithm 1 Problem generation algorithm

```
procedure GENERATE(L,Depth,Path, Tree)
    if Depth ≤ Max then
        for all clauses c ∈ C in the form b_1, ..., b_n ⇒ H such that
                L' is a unifier of H and L do
            Let Path' the unification of Path
            if L' ∉ Path' and every node in Path' is valid then
                Add node L' to Tree
                Let Path' = Path' ∪ {L'}
                Let b'_1 ..., b'_n be the unifications of b_1 ..., b_n
                for all b ∈ {b'_1 ..., b'_n} do GENERATE(b, Depth + 1, Path', t)
                    Add t as a child of L' in the Tree
                end for
            end if
        end for
    end if
end procedure
```

procedure generates the tree in a backward chaining fashion. That is, the root of the tree is a sentence to be proved, while the leaves of the tree are either known literals, such as tautologies and other self-evident predicates, or predicates considered as *assumptions* (givens). The first clause selected by the procedure defines the sentence to be proved. However, the exact form of the predicate to be proved, in terms of the actual points involved, is specified during the execution of the algorithm, since variable substitution may take place during expression unification. During tree construction the algorithm checks whether all nodes in the generated path are *valid*. A sentence is valid if participating terms, angles, triangles, segments, are well formed, e.g. in a segment AB, point A is different from point B.

In Algorithm 1 variables denote points, thus the unification of terms engages only variable substitutions. Unification is executed in the usual sense: if there exists a substitution θ that unifies L and H, i.e. $L\theta = H\theta$, then $b'_1 = b_1\theta, \ldots, b'_n = b_n\theta$ [15].

Term variables are never grounded during problem generation. It is assumed that variables with different names do always refer to different points.

4.3 Problem ranking

This work aims towards the generation of interesting problems, appropriate for usage in real educational practice. The method described above systematically enumerates proof trees, each defining a corresponding problem. There is a need for characterizing problems on the basis of their pedagogic quality [10], as well as for selecting problems according to specific characteristics.

We consider as interesting exercises those that incorporate specific pedagogic qualities. More specifically, we assume that interesting problems engage learner understanding. Greeno [14] asserts that problems that promote understanding

help learners develop internal representations that comprise three characteristics: *coherence* of the internal structure of a problem representation, for example, the coherence of a graph that represents a geometric diagram; *correspondence* between the representation of the problem and a real life representation meaningful to the learner, such as a visual depiction of a geometric diagram; and *connectedness* of general concepts in the learner's knowledge (connections).

Based on the above, a simple measure for identifying interesting exercises is proposed, by identifying certain characteristics of exercise definitions that *may* promote understanding. The degree of connectedness can be measured by the number of different rules involved in the solution of a particular problem. Furthermore, coherence of learner internal structures is promoted by problem definitions which are, by themselves, of high density, that is, engage a large number of connections, i.e., congruency relations, and a small number of related elements. In our problem definition, *points* are the basic elements that are related in the above sense. Thus, we assume that coherence increases with the number of different rules involved in the solution of a particular problem and decreases with the number of points in a particular problem. Regarding correspondence of problem representations, every proposition is a symbolic element that corresponds to an element in a diagram. Although corresponding diagrams are not yet created automatically in our approach, we are currently dealing with issue.

Considering the tree structure of the problem space, it is obvious that the size of the problem space grows exponentially with its depth [16]. Given that deep proof trees correspond to deep problem spaces, we assume that how interesting a problem is increases with the depth of the problem proof tree.

Regarding the estimation of the difficulty of a problem based on internal problem characteristics, we anticipate that difficulty increases with the depth of the problem space, as well as with the number of rules involved.

Given the above, we have identified certain metrics for evaluating generated problems according to their pedagogic *interest*, and *difficulty*, i.e. the maximum depth, M, of the generated problem space, the number, R, of different *rules* that were used for problem generation and the number, P, of points in a particular diagram. We anticipate that generated exercises are ranked for their *interest* according to $I = \frac{M \times R}{P}$, while they are ranked for difficulty according to $D = M \times R$.

4.4 Initial prototype

A prototype has been implemented in PROLOG. This language seemed a natural choice due to its inherent support for the manipulation of logical expressions, unification and backtracking. The prototype generates the tree (initial, goal, intermediate states) by implementing Algorithm 1. Implementation follows certain conventions for geometric rules representation adopted from [17].

The prototype is able to generate exercises in symbolic form and we are currently working towards the semi-automatic construction of diagrams from exercise descriptions.

5 Evaluation

A pilot evaluation was conducted in order to estimate the feasibility of the whole approach. We entered 7 rules in the system, which correspond to specific geometric theorems: Side-Angle-Side rule of triangle congruency, equal angles in corresponding sides in equilateral triangles and equality of corresponding sides in congruent triangles. The prototype generated 270 problems, some of which were isomorphic, that is, identical from a pedagogical point of view. Note that the problem illustrated in Fig. 1 was among the ones generated. All problems were correct, as checked by the author of this paper. From these, 10 problems were selected for evaluation, covering almost evenly the range of all calculated values in interest and difficulty. The small number of problems was due to evaluators' limited time availability.

Selected problems were given to two experienced High School Mathematics teachers in order to be ranked according to two criteria: interest (*Would you assign the particular problem to your students as an understanding/application exercise?*) and difficulty (*How difficult do you consider the problem in question, related to the other problems?*). For each problem, the assumptions, the sentence to be proved as well as a diagram generated by hand were given to the participants. Teachers were allowed to assign the same rank to specific problems. The concordance of teacher rankings was measured by using Kendall's Tau-b correlation coefficient [18]. Concordance was computed for the rankings of the two participants, as well as between each participant and our proposed measures. Corresponding values are depicted in Table 1.

Table 1. Selected problems evaluation data

Difficulty	
T1 and T2	0.51
T1 and $M \times R$	0.41
T2 and $M \times R$	0.73
Interest	
T1 and T2	0.58
T1 and $\frac{M \times R}{P}$	0.47
T2 and $\frac{M \times R}{P}$	0.62

Although the number of participants is very limited and the report of the above data is anecdotal, we see that for both difficulty and interest, the concordances between each teacher and our metrics is of the same level as the corresponding concordances between teachers. Although these results by no means can be generalized, they are hopeful initial indicators of the potential validity of the proposed measures for exercise selection, proving a useful basis for further justification and /or adjustment.

6 Conclusions

Preliminary evaluation has shown that the proposed method can generate useful exercises in Euclidean Geometry. However, the method can be generalized in other domains as well. The description of rules in the form of clauses is universally accepted in both cognitive psychology and knowledge representation literature as a domain independent formalism. The method generates problems that are evaluated according to search space depth, number of rules and problem element coherence /parsimony. From the above, only the measure of coherence / parsimony is defined in a domain-specific fashion. However, the latter characteristic can be easily adapted to specific problem domains. Nevertheless, the above should be demonstrated by applying this method in other domains as future research.

Solving problems in school by no means is limited to well structured problems such as the ones generated here. Generally, the solution of knowledge rich problems, inside and outside educational environments, involves the identification of patterns of solutions, the development of certain, usually domain specific, heuristics [16], which are not covered by our approach. Furthermore, the difficulty of problems in practice does not depend mainly on internal problem properties, but rather on the knowledge of similar problems by the learner.

Thus, more successful problem generation should involve the implication of heuristics, as well as rules; case based reasoning techniques may also be involved, in the sense that a problem generation agent should be based on a knowledge base containing not only a set of rules but also properly indexed problems together with their solutions.

Acknowledgement. The author would like to thank Mr Christos Tsaggaris for his valuable help in the evaluation presented in this paper.

References

[1] Scandura, J.M.: Knowledge representation in structural learning theory and relationships to adaptive learning and tutoring systems. Tech., Inst., Cognition and Learning **5** (2007) 169–271

[2] Anderson, J.R., Boyle, C.F., Farrel, R., Reiser, B.J.: Modelling cognition. In Morris, P., ed.: Cognitive principles in the design of computer tutors. John Wiley and Sons Ltd. (1987) 93–133

[3] Schoenfeld, A.H.: Handbook for research on mathematics teaching and learning. In Grows, D., ed.: Learning to think mathematically: Problem solving, metacognition, and sense-making in mathematics. MacMillan, New York (1992) 334–370

[4] Novick, L.R., Bassok, M.: Cambridge handbook of thinking and reasoning. In Holyoak, J., Morrison, R.G., eds.: Problem Solving. Cambridge University Press, New York, NY (2005) 321–349

[5] Greeno, J.G., Pearson, P.D., Schoenfeld, A.H.: Implications for NAEP of Research for Learning and Cognition. Institute for Research on Learning, Menlo Park, CA (1996)

[6] Heeren, B., Jeuring, J., Gerdes, A.: Specifying rewrite strategies for interactive exercises. Mathematics in Computer Science **3**(3) (2010) 349–370
[7] Jonassen, D.H.: Toward a design theory of problem solving. ETR&D **48**(4) (2000) 63–85
[8] Holohan, E., Melia, M., McMullen, D., Pahl, C.: The generation of e-learning exercise problems from subject ontologies. In: ICALT, IEEE Computer Society (2006) 967–969
[9] Williams, S.: Generating mathematical word problems. In: 2011 AAAI Fall Symposium Series. (2011)
[10] Alsubait, T., Parsia, B., Sattler, U.: Mining ontologies for analogy questions: A similarity-based approach. In Klinov, P., Horridge, M., eds.: OWLED. Volume 849 of CEUR Workshop Proceedings., CEUR-WS.org (2012)
[11] Žitko, B., Stankov, S., Rosić, M., Grubišić, A.: Dynamic test generation over ontology-based knowledge representation in authoring shell. Expert Systems with Applications **36** (2009) 8185–8196
[12] Papasalouros, A., Kotis, K., Kanaris, K.: Automatic generation of tests from domain and multimedia ontologies. Interactive Learning Environments **19**(1) (2011) 5–23
[13] Schoenfield, A.H.: On having and using geometric knowledge. In Hiebert, J., ed.: Conceptual and procedural knowledge: The case of mathematics. LEA Publishers, Hillsdale, NJ (1986) 225–264
[14] Greeno, J.G.: Understanding and procedural knowledge in mathematics instruction. Educational Psychologist **12**(3) (1987) 262–283
[15] Chang, C.L., Lee, R.C.T.: Symbolic Logic and Mechanical Theorem Proving. 1st edn. Academic Press, Inc., Orlando, FL, USA (1997)
[16] Chi, M.T., Glaser, R.: Problem solving ability. In Sternberg, R., ed.: Human Abilities: An Information-Processing Approach. W. H. Freeman & Co (1985) 227–257
[17] Coelho, H., Pereira, L.M.: Automated reasoning in geometry theorem proving with prolog. Journal of Automated Reasoning **2** (1986) 329–390
[18] Kendall, M.: Rank Correlation Methods. C. Griffin, London (1975)