Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Synthesis of Geometry Proof Problems

Chris Alvin
Louisiana State University
calvin1@lsu.edu

Sumit Gulwani
Microsoft Research, Redmond
sumitg@microsoft.com

Rupak Majumdar
MPI-SWS
rupak@mpi-sws.org

Supratik Mukhopadhyay
Louisiana State University
supratik@csc.lsu.edu

Abstract

This paper presents a semi-automated methodology for generating geometric proof problems of the kind found in a high-school curriculum. We formalize the notion of a geometry proof problem and describe an algorithm for generating such problems over a user-provided figure. Our experimental results indicate that our problem generation algorithm can effectively generate proof problems in elementary geometry. On a corpus of 110 figures taken from popular geometry textbooks, our system generated an average of about 443 problems per figure in an average time of 4.7 seconds per figure.

1 Introduction

Learning in mathematics is more deeply rooted when a student is able to view a problem from multiple perspectives: graphically, numerically, and algebraically. High school geometry is particularly interesting in this regard because it combines the implicit visual perspective and deductive logic skills. This paper presents a technology to enhance geometry education. In particular, we present a technique for automatically generating fresh geometry proof problems from the figures of given problems.

Generating fresh problems that have specific solution characteristics (such as difficulty level, use of a certain set of concepts) is a difficult task for the teacher. Automating this has several benefits. First, it can help avoid copyright issues. It is illegal to make photocopies of a textbook and may not be legal to publish an original problem from a textbook on a course website. A problem generation tool can provide instructors with fresh problems (that have characteristics similar to that of the original problem) for use in their assignments, exams, or lecture notes. Second, it can help prevent cheating in classrooms or online education platforms (with unsynchronized instruction) since each student can be provided with a different problem but with the same characteristics. Third, it can be used to generate personalized workflows for students. If a student solves a problem correctly, then the student may be presented with a problem that is more difficult than the last problem, or exercises a richer set of concepts. If a student fails to solve a problem, then the student may be presented with simpler problems to identify, reinforce, and master core concepts.

We formalize the notion of a geometry proof problem, which consists of a figure, some assumptions about the figure, goals that need to be established about the figure, and the set of axioms that need to be used. We propose a semi-automated methodology for generating such problems. Given a figure and a set of axioms, our problem generation technique produces a set of problems over that figure in the form of pairs of assumptions and goals. Such problems, generated across a large set of figures provided by the user, can be stored in a database along with their characteristics. This empowers users to query the database with specific characteristics to obtain custom problems.

Our problem generation technique operates in three steps. First, it produces a hypergraph (Definition 2) that represents all possible proofs for all possible problems over a given pair of user-provided figures and axioms. The hypergraph construction requires enumerating all facts that are true of the figure as nodes in the hypergraph. Furthermore, a set of source facts is connected to a target fact using a directed hyperedge labeled with a user-provided axiom if the axiom can be used to deduce the target fact from the source facts. Then, the tool systematically enumerates all minimal sets of assumptions (Algorithm 1). An assumption is a fact about the figure, and informally, a set of assumptions is minimal if every assumption is non-redundant. Finally, for any minimal set of assumptions I, the tool systematically enumerates all possible goal sets G such that (I, G) is an interesting problem (Algorithm 2).

We evaluated the effectiveness of our problem generation algorithm on 110 figures taken from various geometry textbooks. Our algorithm generated an average of 443 problems in an average time of 4.7 seconds per figure. We also observed that there were several problems with same characteristics across various figures.

This paper makes the following contributions:

- We formalize the notion of a geometry problem (§2) and motivate some problem generation interfaces (§3).
- We present a technique for generating proof problems over a given geometric figure (§4).
- We describe experimental results illustrating the efficacy of our problem generation interfaces and our problem generation algorithm (§5).

Axiom Name	Premises	Conclusion(s)
Midpoint Def.	Midpoint(M, AB)	$AM = MB$
Angle Addition	$\angle ABC, \angle CBD$ Exterior(D, $\angle ABC$)	$\angle ABC + \angle CBD$ $= \angle ABD$
Vertical Angles	Intersect(X, AB, CD)	$\angle AXD \cong \angle CXB$, $\angle AXC \cong \angle BXD$
Side-Side-Side	$\Delta ABC, \Delta DEF$, $AB \cong DE, BC \cong EF$ $CA \cong FD$	$\Delta ABC \cong \Delta DEF$
Alt. Int. Angles	$CD \parallel EF$, Intersect(M, AB, CD), Intersect(N, AB, EF)	$\angle ENM \cong \angle NMD$, $\angle FNM \cong \angle NMC$

Figure 1: Example of Axioms

2 Preliminaries

Informally, a geometric figure is a pictorial representation of a collection of geometric objects (points, lines, circles) in a specific orientation with each other. Internally, we represent geometric figures using first-order logic constraints which can be derived by analyzing a pictorial representation. We work in a first order language with arithmetic, with constants ranging over points. We omit a full description of the logical language and illustrate it through examples. Our logic consists of relations such as betweenness $\mathsf{Between}(A, B, C)$, congruence, and equality relations on line segments or angles. For ease of readability, in the following examples, we also use derived predicates such as $\mathsf{Triangle}(A, B, C)$ (the three points form a triangle, denoted ΔABC), $\mathsf{Collinear}(A, B, C)$ (points are collinear), $\mathsf{RightAngle}(A, B, C)$, etc.

We compute internal representations from pictorial representations of a figure. We assume that input figures are drawn to scale but the problem instances we generate will not assume that figures are drawn to scale. Thus, in the internal representation for a figure Fig, we distinguish between *implicit* and *explicit* facts. Implicit predicates only provide orientation (or "betweenness") information but not relationships on measurements. Explicit predicates provide relations based on measurement and may not hold when the figure is distorted. For example, implicit predicates would state that ABC is a triangle or that line segments AB and CD intersect at M, and explicit predicates would state $AB = CD$ or $\angle ABC$ is a right angle. Technically, implicit predicates are those facts about the figure provable in *ordered geometry* (Coxeter 1969), and explicit predicates are those facts provable in Euclidean geometry minus the implicit ones. For a figure Fig, we write $\mathcal{I}(\mathsf{Fig})$ for the set of implicit facts and $\mathcal{E}(\mathsf{Fig})$ the set of explicit facts.

A *geometry axiom* is a Horn clause whose ground instances are implicit or explicit predicates. That is, an axiom consists of a set of *premises* and a *conclusion*. The free variables in an axiom are (implicitly) universally quantified. Given an axiom A, we say that A *derives* a predicate p from a set P of predicates if there is an instantiation of the premises of A with P and the conclusion with p. Fig. 1 gives some examples of geometry axioms.

Definition 1 (Geometry Problem). *Let* Fig *be a figure, let* $\mathcal{I}(\mathsf{Fig})$ *be the set of implicit facts, and let* Axm *be a set of geometry axioms. A* geometry (proof) problem *over* $(\mathsf{Fig}, \mathsf{Axm})$ *is a pair* (I, G)*, where the assumptions* $I \subseteq \mathcal{E}(\mathsf{Fig})$ *and goals* $G \subseteq \mathcal{E}(\mathsf{Fig})$ *are sets of explicit facts such that* $I \cap G = \emptyset$ *and* $\mathcal{I}(\mathsf{Fig}) \cup I \cup \mathsf{Axm}$ *imply each* $g \in G$ *using first-order reasoning.*

In the above definition, we require the disjointness condition between I and G to ensure problems are non-trivial, and the derivation condition to ensure problems have solutions. A geometry problem (I, G) over $(\mathsf{Fig}, \mathsf{Axm})$ is *interesting* if no strict subset of I together with $\mathcal{I}(\mathsf{Fig})$ can establish every goal in G using Axm. An interesting problem is *strict* if G is minimal, i.e., (I, G') is not interesting for any strict subset $G' \subsetneq G$. Observe that an interesting problem where G is a singleton is strict. An interesting geometry problem (I, G) over $(\mathsf{Fig}, \mathsf{Axm})$ is *complete* if for any predicate $p \in \mathcal{E}(\mathsf{Fig})$, $\mathcal{I}(\mathsf{Fig}) \cup I \cup \mathsf{Axm}$ derives p. A complete problem is strict if it is not complete for any strict subset G' of G. Fig. 2 gives some examples of interesting and complete geometry problems.

Let (I, G) be a problem over $(\mathsf{Fig}, \mathsf{Axm})$. A proof that $\mathcal{I}(\mathsf{Fig}) \cup \mathsf{Axm} \cup I$ derives G consists of first-order derivations, one for each $g \in G$, whose root is labeled g, whose leaves are elements of $\mathcal{I}(\mathsf{Fig}) \cup \mathcal{E}(\mathsf{Fig})$ and whose internal nodes are obtained by instantiating an axiom from Axm. Our problem generation algorithm will search through many proofs. Hence, we use a hypergraph representation for all possible derivations. Since the set $\mathcal{I}(\mathsf{Fig})$ is fixed, we do not represent nodes for them.

Definition 2. *A* saturated hypergraph $H(\mathsf{Fig}, \mathsf{Axm})$ *for a pair* $(\mathsf{Fig}, \mathsf{Axm})$ *is a hypergraph whose nodes consist of all predicates in* $\mathcal{E}(\mathsf{Fig})$ *and whose edges are of the form* (P, p, A)*, where* $P \subseteq \mathcal{E}(\mathsf{Fig})$ *is a set of explicit predicates,* $p \in \mathcal{E}(\mathsf{Fig})$ *is an explicit predicate, and* $A \in \mathsf{Axm}$*, such that there exists a set* $Q \subseteq \mathcal{I}(\mathsf{Fig})$ *such that* A *derives* p *from* $P \cup Q$*.*

For a set $T \subseteq \mathcal{E}(\mathsf{Fig})$, we define $\mathit{Derive}(T) = \{g \in \mathcal{E}(\mathsf{Fig}) \mid T \cup \mathcal{I}(\mathsf{Fig}) \cup \mathsf{Axm} \models g\}$. The set $\mathit{Derive}(T)$ coincides with the set of nodes reachable in the hypergraph $H(\mathsf{Fig}, \mathsf{Axm})$ starting from the set T of nodes. Thus, $\mathit{Derive}(T)$ can be computed for every set $T \subseteq \mathcal{E}(\mathsf{Fig})$ in time polynomial in the size of the hypergraph.

3 Problem Generation Interface

Before we present our problem synthesis algorithm, we provide a user's view to interacting with our system.

The user provides a geometry figure drawn to scale and a set of axioms as inputs to the system, and can specify parameters to generate a desired set of problems with specific features.

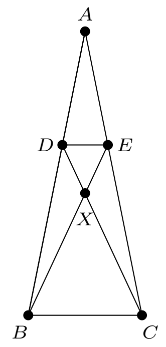
3.1 Features of a Geometry Problem

A geometry problem $P = (I, G)$ over a pair $(\mathsf{Fig}, \mathsf{Axm})$ has several features such as:

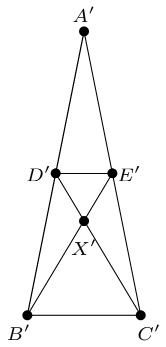
- The objects of the figure Fig and their properties $\mathcal{I}(\mathsf{Fig})$, $\mathcal{E}(\mathsf{Fig})$, e.g., the number of points, triangles, etc.
- The size of the goal set $|G|$.

Consider figures $(\mathsf{Fig}_1, \mathsf{Axm})$ and $(\mathsf{Fig}_2, \mathsf{Axm})$ shown in (a) and (b) below, respectively and Axm is the the common set of axioms. The original textbook problem is (I, G) where $I = \{\triangle ABE \cong \triangle ACD$ and $G = \{\triangle ADE \sim \triangle ABC\}$.
Fig_1 is indistinguishable from Fig_2 save points D, D', E, E', and consequently X, X'. Specifically, in Fig_2 D' is the midpoint of segment $A'B'$; similarly E' is the midpoint of $A'C'$. That is, $\mathcal{I}(\mathsf{Fig}_1) = \mathcal{I}(\mathsf{Fig}_2)$ while $\mathcal{E}(\mathsf{Fig}_1) \neq \mathcal{E}(\mathsf{Fig}_2)$ since $\{\text{Midpoint}(D', A'B'), \text{Midpoint}(E', A'C')\} \subset \mathcal{E}(\mathsf{Fig}_2)$.

(a)
$(\mathsf{Fig}_1, \mathsf{Axm})$



(b)
$(\mathsf{Fig}_2, \mathsf{Axm})$



For $(\mathsf{Fig}_1, \mathsf{Axm})$ and $(\mathsf{Fig}_2, \mathsf{Axm})$ we will generate the exact same set of problems $(I, \{g_1\})$ where $I = \{\Delta ABE \cong \Delta ACD\}$ and g_1 may be any of the following example propositions.

- $\triangle ADE \sim \triangle ABC$
- $\angle BCD \cong \angle CBE$
- $\triangle DBC \cong \triangle ECB$
- $\triangle BCX$ is Isosceles
- $\triangle DEX$ is Isosceles
- $DE \parallel BC$
- $\angle DEA \cong \angle CBA$
- $\triangle BDX \cong \triangle CEX$

Since I completely defines Fig_1, all problems $(I, \{g_1\})$ are *strictly complete problems*. Whereas I does not define Fig_2 since it is not possible to prove $\text{Midpoint}(D', A'B')$ nor $\text{Midpoint}(E', A'C')$. Therefore, for Fig_2 all problems in $(I, \{g_1\})$ are simply *interesting problems*.

Figure 2: An Example of Strictly Interesting and Strictly Complete Problems

- The type of the goal, e.g., congruent triangles, equal segments, etc.
- Quantitative features of a proof, such as depth of a proof (i.e., the longest path from the assumptions to the goal in the proof), the width of a proof (maximal number of nodes in a level in the proof), the number of deduction steps (i.e., the number of hyperedges in the proof), and the number of axioms used. These features can be computed from the representation of proofs in the saturated hypergraph.
- A subset of Axm that occurs in every proof of the problem.
- Whether the problem is complete or not.

Our system allows defining arbitrary features as long as they are efficiently computable from the syntactic description of the problem or from the hypergraph representation.

3.2 Query Interface to Problem Generation

We propose an interface where the teacher can specify a relational query over the set of problem features and obtain a corresponding set of problems. We describe a semi-automated methodology to support this interface. Our methodology requires manual input of $(\mathsf{Fig}, \mathsf{Axm})$ pairs. For each such pair, we generate the set of all interesting problems using the problem generation technique described in (§4). This set of problems, along with their features, populate a relational database. We may then query the database using a standard relational query (§5 gives examples of such queries with results in Fig. 10).

A student or teacher may define their own pair $(\mathsf{Fig}, \mathsf{Axm})$ either using their own creativity or from textbooks, to generate fresh problems corresponding to that pair. In that respect, our methodology has a multiplicative effect: starting from the figure of a problem, our algorithms generate many more problems over the same figure.

4 Algorithm for Problem Generation

Our algorithm for problem generation has three steps. In the following exposition, we focus on clarity rather than performance. The enumeration of problems is exponential in the worst case; we show in (§5) that nevertheless, the enumeration can be performed successfully in practice.

4.1 Step 1: Hypergraph Construction

The input to the algorithm is a geometry figure Fig drawn to scale and a set of axioms (Horn clauses). The algorithm internally computes the sets $\mathcal{I}(\mathsf{Fig})$ and $\mathcal{E}(\mathsf{Fig})$ and constructs the saturated hypergraph for $(\mathsf{Fig}, \mathsf{Axm})$. The hypergraph is used to compute $\mathit{Derive}(T)$ queries for sets $T \subseteq \mathcal{E}(\mathsf{Fig})$ in the subsequent steps of the algorithm.

4.2 Step 2: Minimal Assumption Generation

A set $T \subseteq \mathcal{E}(\mathsf{Fig})$ is *minimal* if either $T = \emptyset$ or for each $t \in T$, we have that $T \setminus \{t\}$ is minimal and $\mathit{Derive}(T) \neq \mathit{Derive}(T \setminus \{t\})$. Minimality is a necessary condition for an interesting problem.

In the second step, the problem synthesis algorithm systematically enumerates all minimal sets of assumptions. Algorithm 1 shows a simple fixed-point procedure to compute the set of all minimal sets.

4.3 Step 3: Strictly Interesting Problem Synthesis

The final step enumerates, for each minimal set of assumptions I, all possible goal sets G such that (I, G) is a strictly interesting problem.

```
Algorithm AllMinimalSets(Fig, Axm)
Input: Figure Fig, axioms Axm
Output: Set of all minimal sets of $\mathcal{E}(\mathsf{Fig})$
 1: $AllSets = \{\emptyset\}$
 2: $Old = \emptyset$
 3: while $AllSets \neq Old$ do
 4:     $Old = AllSets$
 5:     for all $I \in AllSets$ do
 6:         for all $f \in \mathcal{E}(\mathsf{Fig})$ s.t.
            $Derive(I) \neq Derive(I \cup \{f\})$ do
 7:             $AllSets = AllSets \cup \{I \cup \{f\}\}$
 8:         end for
 9:     end for
10: end while
11: return $AllSets$
```
Algorithm 1: Algorithm *AllMinimalSets*

We present the third step as the non-deterministic procedure Algorithm 2. It takes as input a figure Fig and axioms Axm, as well as a minimal set I of explicit predicates. It computes a strictly interesting problem by "growing" a set G of goals and returns (I, G) as the generated problem. Initially, the set G is empty (line 1). While the current set of goals is not strong enough to ensure the problem is interesting (line 2), the algorithm generates a new goal. To generate a new goal, the algorithm finds (non-deterministically, line 3) an assumption f that is not used to prove the current set of goals and finds (non-deterministically, line 5) a goal that is derivable using I but not without this assumption. Notice that since I is minimal, the set T on line 4 is non-empty. However, to ensure the condition $I \cap G = \emptyset$, we choose g from the set $T \setminus I$ on line 5, which may be empty.

By construction, Algorithm 2 ensures that returned problems are strictly interesting. For the returned pair (I, G), since the **while** loop exits, we know that every $f \in I$ is necessary to prove some goal in G; hence (I, G) is interesting. Further, the problem is strictly interesting since the algorithm returns a minimal set of goals G.

The non-deterministic choices of the algorithm are denoted by the **choose** operator, which selects an element of a set (if non-empty), and fails otherwise. By iterating over possible non-deterministic choice, the algorithm can generate every possible strictly interesting problem with assumption I.

Finally, in order to generate a complete problem, we can check that the input I to procedure *GenProblem* can derive all explicit facts, i.e., $Derive(I) = \mathcal{E}(\mathsf{Fig})$.

Theorem 1. *(1) Algorithm AllMinimalSets*(Fig, Axm) *returns the set of all minimal sets for* (Fig, Axm). *(2)* ***[Soundness]*** *Let I be a minimal set. If Algorithm GenProblem*(Fig, Axm, I) *returns* (I, G) *then* (I, G) *is a strictly interesting problem over* (Fig, Axm). *(3)* ***[Completeness]*** *Let* (I, G) *be a strictly interesting problem for* (Fig, Axm). *Then there is a possible run of Algorithm GenProblem*(Fig, Axm, I) *that returns* (I, G).

Fig. 2 shows some problems that were automatically generated by our algorithm.

```
Algorithm GenProblem(Fig, Axm, I)
Input: Figure Fig, axioms Axm, minimal set $I \subseteq \mathcal{E}(\mathsf{Fig})$
Output: Strictly interesting problem $(I, G)$
 1: $G = \emptyset$
 2: while $\exists f \in I$ s.t. $G \subseteq Derive(I \setminus \{f\})$ do
 3:     $f = \mathbf{choose}(\{f \in I \mid G \subseteq Derive(I \setminus \{f\})\})$
 4:     $T = Derive(I) \setminus Derive(I \setminus \{f\})$
 5:     $g = \mathbf{choose}(T \setminus I)$
 6:     $G = G \cup \{g\}$
 7: end while
 8: return $(I, G)$
```
Algorithm 2: Algorithm *GenProblem*

5 Experimental Results

Benchmark We ran our problem generation algorithm on a set of 110 figures taken from standard mathematics textbooks in India (Sinclair and Dikshit 2006b; 2006a) as well as textbooks and workbooks popular in the United States (Boyd 2006; Larson et al. 2007; McDougal 2007; Jurgensen, Brown, and Jurgensen 1988). We used a uniform set of axioms for all of our experiments; this set included axioms related to parallel lines, congruent triangles, similar triangles, etc.

The distribution of these 110 figures described by the size of the implicit facts per figure, $|\mathcal{I}(\mathsf{Fig})|$, is a bimodal distribution with modes around 40 and 75 and mean 46.5. The bimodal distribution indicates our attempt to balance our experiments with simple as well as more complex figures.

The distribution of these 110 figures described by the number of deduced facts per figure, $|\mathcal{E}(\mathsf{Fig})|$, is a positively-skewed distribution (mean 108, median 82, and standard deviation 96.7) that indicated few figures result in a large hypergraph making our problem generation algorithm often run efficiently in practice.

Effectiveness of our problem generation algorithm *GenProblem* We now present evaluation of our problem generation algorithm *GenProblem* with respect to the number of problems that it generates as well as the time taken to generate those problems. We ran our experiments on a laptop with Intel Core i5-2520M CPU at 2.5GHz with 8 GB RAM on 64-bit Windows 7 operating system.

We modified *GenProblem* to only generate problems where $|G| \leq 2$. This is because our preliminary prototype encountered memory issues with $|G| > 2$ since the problem generation procedure is exponential in $|G|$. (We hope to resolve this issue with a more optimized implementation in the future.) For each (Fig, Axm) pair, we fixed I to be the minimal set of assumptions that corresponded to the original textbook problem description corresponding to the figure F. For the 110 figures we observed a mean of 2.3 assumptions per figure with standard deviation 1.1; Fig. 3 presents statistics on the size of this fixed minimal set per figure.

Given a set of assumptions I over a pair (Fig, Axm), we determine the Boolean classification whether I completely defines Fig. We may informally describe a complete problem as a problem that is not open to interpretation. That

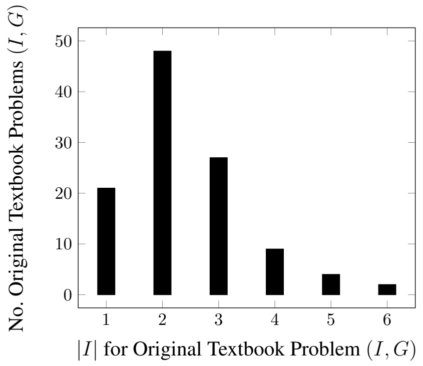


Figure 3: Number of assumptions $|I|$ per original textbook problem (I, G)

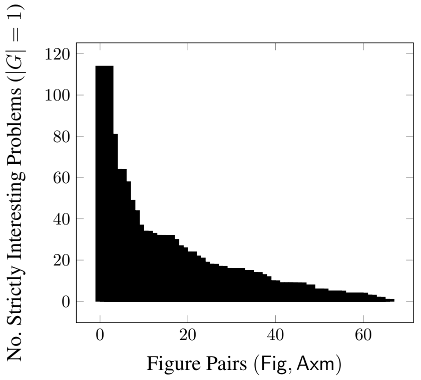


Figure 4: Number of strictly interesting problems where $|G| = 1$ generated per pair $(\mathsf{Fig}, \mathsf{Axm})$

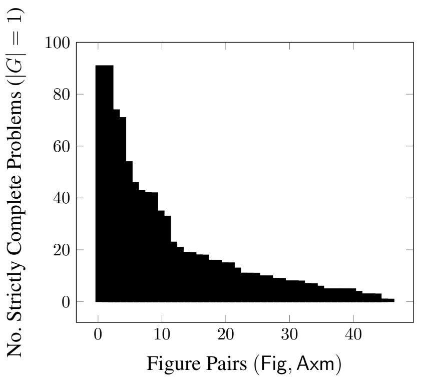


Figure 5: Number of strictly complete problems where $|G| = 1$ generated per pair $(\mathsf{Fig}, \mathsf{Axm})$

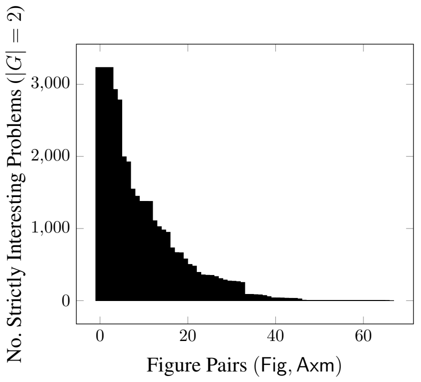


Figure 6: Number of strictly interesting problems where $|G| = 2$ generated per pair $(\mathsf{Fig}, \mathsf{Axm})$

is, complete problems are ideal for formal assessments. On the other hand, interesting problems are more malleable and therefore more applicable to homework or in-class investigations. Textbook problems are generally a mix of interesting and complete problems. We found for only 45 of 110 figures, the original textbook problem associated with it was complete. We expected a larger number of complete problems, but found that when drawing figures into our front-end slate, we were more likely to construct figures with unintended facts (e.g. points were likely to be midpoints, triangles likely to be isosceles or equilateral). This psychological factor lead to a greater number of original textbook problems being classified as interesting (but not complete).

Our methodology results in a large multiplicative effect: from a single pair $(\mathsf{Fig}, \mathsf{Axm})$ we are able to generate many problems. For the 65 of 110 original textbook figures that were classified as corresponding to interesting (but not complete) problems, we generated a total of 1309 and an average of 20.1 strictly interesting problems (I, G) where $|G| = 1$; the associated distribution is shown in Fig. 4. For the remaining 45 of 110 original textbook figures, which were classified as corresponding to complete problems, we generated a total of 877 and an average of 19.5 strictly complete problems (I, G) where $|G| = 1$ with distribution in Fig. 5. For $|G| = 2$, we generated 14760 strictly complete problems and 31801 strictly interesting (but not complete) problems. When $|G| = 2$ we have an empirical validation of the exponential growth in the number of generated problems. For a fixed set of assumptions I, the definition of a strict problem dictates $|I| \geq |G|$ for any G. Since many of our original textbook problems had $|I| = 1$, many figure pairs $(\mathsf{Fig}, \mathsf{Axm})$ cannot generate problems with more than a single goal. The corresponding distributions (shown in Fig. 6 and Fig. 7) are heavily skewed with mean 489 and median 84 for strictly interesting (but not complete) problems as well as mean 328 and median 49 for strictly complete problems.

GenProblem took an average time of 4.7 seconds (with standard deviation of 10.5 seconds) per $(\mathsf{Fig}, \mathsf{Axm})$ pair to generate the above mentioned problems with $|G| \leq 2$. For a given $(\mathsf{Fig}, \mathsf{Axm})$ pair, the majority of the processing time is in construction of the saturated hypergraph. Therefore, we expect a correlation between the number of explicit facts for F and the amount of time to process. As the worklist construction of $H(\mathsf{Fig}, \mathsf{Axm})$ requires that we compare each newly deduced node against all existent nodes in $H(\mathsf{Fig}, \mathsf{Axm})$, we expect hypergraph construction to be

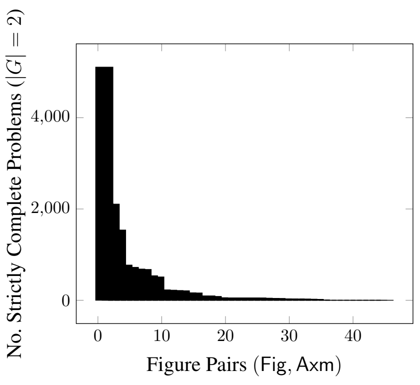


Figure 7: Number of strictly complete problems where $|G| = 2$ generated per pair (Fig, Axm)

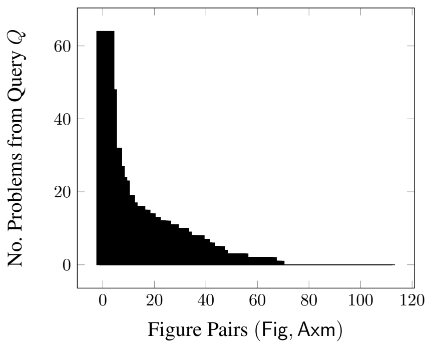


Figure 8: Problems Per Pair (Fig, Axm) for Query $Q = \{\text{steps} = 6 \text{ to } 10, \text{width} = 4 \text{ to } 8\}$

quadratic in the number of nodes in $H(\text{Fig}, \text{Axm})$; we have a strong quadratic correlation with coefficient $r^2 = 0.7785$.

Effectiveness of our methodology Once all problems are generated from all pairs (Fig, Axm), we can obtain problems with similar features across different figures. Let's consider two distinct use-cases from perspectives of a teacher and a student.

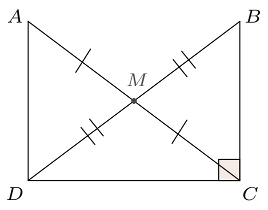
Consider the scenario where a teacher wants to generate a set of problems for students to review before the final exam. The teacher might construct a query Q to obtain problems that are (1) medium-to-hard (6 to 10 deductive steps) with (2) average width (4 to 8), and (3) contain a single goal. Q returns a total of 706 problems from our database with an average of 6.4 problems per pair (Fig, Axm); the graph in Fig. 8 details the number of problems per pair that satisfy Q. Fig. 9 shows a sample of those 706 problems.

Now let's consider a common scenario for a student preparing for an exam that will test on say proving triangles congruent using any technique. In this case, the student may specify a series of queries Q_i capturing problems of increasing difficulty as measured by the number or kind of deductive steps required. Each Q_i also specifies that the problem should have a single goal g that makes use of Congruent Triangles predicate. These queries Q_i are discussed below with the query results enumerated in Fig. 10.

Consider the pair (Fig, Axm) where Fig is the figure below and Axm is our common set of axioms. The original problem from the textbook over (Fig, Axm) is (I, G), where

$$I = \{\text{Midpoint}(M, BD), AM = MC, \text{RightAngle}(B, C, D)\}.$$

$$G = \{\triangle BMC \cong \triangle DMA, \text{RightAngle}(A, D, C), \triangle ADC \cong \triangle BCD, 2BM = AC\}.$$



The query Q generates several new problems of the form (I', g') over the pair (Fig, Axm), where $I' = I$ and g' takes on any of the following propositions.

- CD is an altitude of $\triangle ADC$
- RightTriangle(A, D, C)
- $\angle CDB$ and $\angle MAD$ are complementary
- $AD \perp CD$
- $AD \parallel BC$

Figure 9: Problems satisfying query $Q = \{|G| = 1, \text{steps} = 6 \text{ to } 10, \text{steps} = 4 \text{ to } 8\}$ over a given geometric figure.

i	Query: Q_i	Number Problems	Over (Fig, Axm)
1	$\{\text{s} = 1-2, G\}$	23	22
2	$\{\text{s} = 3-7, G\}$	73	50
3	$\{\text{s} = 6, \text{d} = 4, \text{w} = 5, G\}$	1	1
4	$\{\text{s} = 6, \text{d} = 4-5, G\}$	54	28
5	$\{\text{s} \geq 10, G\}$	26	14

Figure 10: Problems and Figures Satisfying Student Queries ($s =$ steps, $d =$ depth, $w =$ width, $G = \{\cong \triangle s\}$)

The student begins by specifying the query $Q_0 = \{$ steps = 1 to 2, $g\}$ and is provided one of the 23 problems. Assuming success with a few practice problems, the student seeks a series of more difficult problems and defines $Q_1 = \{$steps = 3 to 7, $g\}$. After completing some of the 73 possible interesting problems that match Q_1, the student encounters a problem that is intriguing in its structure. As a point of interest and practice, the student defines a query based on the parameters of the problem just completed: $Q_2 = \{$steps = 6, depth = 4, width = 5, $g\}$. The result of the query is that there is no other problem with the defined characteristics. Instead, the student relaxes the restrictions resulting in $Q_3 = \{\text{steps} = 6, \text{depth} = 4 \text{ to } 5, g\}$ and acquires 26 problems. Finally, the student may provide a query that requires the proof problem to have more than 10 deductions steps: $Q_4 = \{\text{steps} \geq 10, g\}$. After successfully completing one or more of these 26 problems, the student can be confident in their preparation for the exam.

6 Related Work

There are two categories of related work that we discuss below. (Gulwani 2014) describes some of this related work in more detail.

Technology for Geometry Education Automated geometry theorem proving (consisting of several techniques such as Wus method (Wen-Tsün 1986), Grobner basis method (Kapur 1986), and angle method (Chou, shan Gao, and Zhang 1994)) is one of the most successful areas of automated reasoning. Traditional automated geometry theorem proving systems tend to produce arbitrary proofs in the underlying logical domain that may not be readable and may be beyond the vocabulary taught in the class. Tutoring oriented systems such as Geometry Expert (Gao and Lin 2004) and Geometry Explorer (Wilson and Fleuriot 2005) allow students to create geometry constructions and use interactive provers to check and prove properties of those constructions. (Gulwani, Korthikanti, and Tiwari 2011; Itzhaky et al. 2013) even present techniques for automatically synthesizing geometry constructions given logical constraints that relate the various objects in the construction.

Our system can be used to solve those proof problems that do not require construction of any new object in the given geometric figure. It uses a relatively simple methodology of hypergraph reachability to check whether the goal can be reached from the assumptions. The novelty of our system lies in the hypergraph construction and associated algorithms over it that enables generation of various interesting problems.

Automatic Problem Generation (Singh, Gulwani, and Rajamani 2012) describes a problem generation technology for generating problems that are similar in structure to a given algebraic identity proof problem. The underlying technology leverages continuity of the underlying domain of algebraic expressions, and uses extension of polynomial identity testing to check the correctness of a generated problem candidate on a random input. In contrast, the domain of Boolean valued geometry predicates is non-continuous or discrete, and hence requires a different technology of checking whether a given problem is interesting or complete. Furthermore, our technology also enables solution generation.

(Andersen, Gulwani, and Popovic 2013) describes a problem generation technology for *procedural domain*, which includes problems commonly found in middle-school math curriculum such as subtraction and greatest common divisor computation. The underlying technology leverages test input generation techniques (Anand et al. 2013) to generate problems that explore various paths in the procedure that the student is expected to learn. In contrast, we address problem generation for a *conceptual domain*, where there is no step-by-step decision procedure that the student can use to solve a problem, but it requires creative skills such as pattern matching.

(Ahmed, Gulwani, and Karkare 2013) describes a problem generation technology for natural deduction problems. The underlying technology involves offline generation of a Universal proof graph (which is a hypergraph that represents all possible inference rule applications over propositions of bounded size) and then traversal of this graph to generate problems with certain features. Our hypergraph is similar in that it represents all possible applications of the various axioms, but its problem specific nature makes it much smaller and allows us to enumerate all interesting problems apriori and store them in a database to enable efficient identification of problems with specific features.

7 Conclusions and Future Work

This paper shows how to automatically generate geometry proof problems of the kind that are common in high-school curriculum. Automatic problem generation can be an important component of personalized workflow creation in intelligent tutoring.

There are several directions for future work. (a) Automatic figure generation: Our semi-automated methodology for problem generation requires the user to provide figures, with respect to which we generate problems. It would be useful to automatically generate such figures. (b) Natural language generation: Our problem generation tool generates problems at the level of logical predicates. It would be useful to translate the logical predicates into an equivalent, but succinct, natural language description in the form of a word problem. (c) Deployment and user studies: We would like to deploy our tool as part of personalized problem generation system and measure its effectiveness in improving student learning. (d) Application to other subject domains: We feel that our hypergraph-based problem generation approach can be adapted to work for other subject domains where the goal is to derive a new fact (or even compute some desired value) using a series of steps starting from some set of facts. This includes various non-inductive proof domains including those in geometry, algebra, and logic.

References

Ahmed, U. Z.; Gulwani, S.; and Karkare, A. 2013. Automatically generating problems and solutions for natural deduction. In *IJCAI*.

Anand, S.; Burke, E.; Chen, T. Y.; Clark, J.; Cohen, M. B.; Grieskamp, W.; Harman, M.; Harrold, M. J.; and McMinn, P. 2013. An orchestrated survey on automated software test case generation. *Journal of Systems and Software* 86(8).

Andersen, E.; Gulwani, S.; and Popovic, Z. 2013. A trace-based framework for analyzing and synthesizing educational progressions. In *CHI*.

Boyd. 2006. *Geometry (NJ Edition)*. New York, NY: Glencoe / McGraw-Hill.

Chou, S.-C.; shan Gao, X.; and Zhang, J.-Z. 1994. *Machine proofs in geometry—Automated production of readable proofs for geometry theorems.* World Scientific.

Coxeter, H. 1969. *Introduction to Geometry*. John Wiley.

Gao, X.-S., and Lin, Q. 2004. Mmp/geometer a software package for automated geometric reasoning. *Automated Deduction in Geometry*.

Gulwani, S.; Korthikanti, V. A.; and Tiwari, A. 2011. Synthesizing geometry constructions. In *PLDI*.

Gulwani, S. 2014. Example-based learning in computer-aided STEM education. *To appear in Commun. ACM*.

Itzhaky, S.; Gulwani, S.; Immerman, N.; and Sagiv, M. 2013. Solving geometry problems using a combination of symbolic and numerical reasoning. In *LPAR*.

Jurgensen, R.; Brown, R.; and Jurgensen, J. 1988. *Geometry*. Boston, MA: Houghton Mifflin Company.

Kapur, D. 1986. Using Gröbner bases to reason about geometry problems. *J. Symb. Comput.* 2(4).

Larson, R.; Boswell, L.; Kanold, T.; and Stiff, L. 2007. *Geometry*. Evanston, IL: McDougal Littel.

McDougal, H. 2007. *Larson Geometry: Practice Workbook*. Evanston, IL: McDougal Littel.

Sinclair, P., and Dikshit, G. 2006a. *Mathematics Textbook for Class IX*. http://www.ncert.nic.in/ncerts/textbook/textbook.htm?iemh1=0-15.

Sinclair, P., and Dikshit, G. 2006b. *Mathematics Textbook for Class X*. http://www.ncert.nic.in/ncerts/textbook/textbook.htm?jemh1=0-14.

Singh, R.; Gulwani, S.; and Rajamani, S. 2012. Automatically generating algebra problems. In *AAAI*.

Wen-Tsün, W. 1986. Basic principles of mechanical theorem proving in elementary geometries. *J. Autom. Reasoning* 2(3).

Wilson, S., and Fleuriot, J. D. 2005. Combining dynamic geometry, automated geometry theorem proving and diagrammatic proofs. In *Workshop on User Interfaces for Theorem Proving*.