EDGEWORTH: Efficient and Scalable Authoring of Visual Thinking Activities

Wode Ni
nimo@cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Sam Estep
estep@cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Hwei-Shin Harriman
hweishih@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Kenneth R. Koedinger
koedinger@cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Joshua Sunshine
sunshine@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

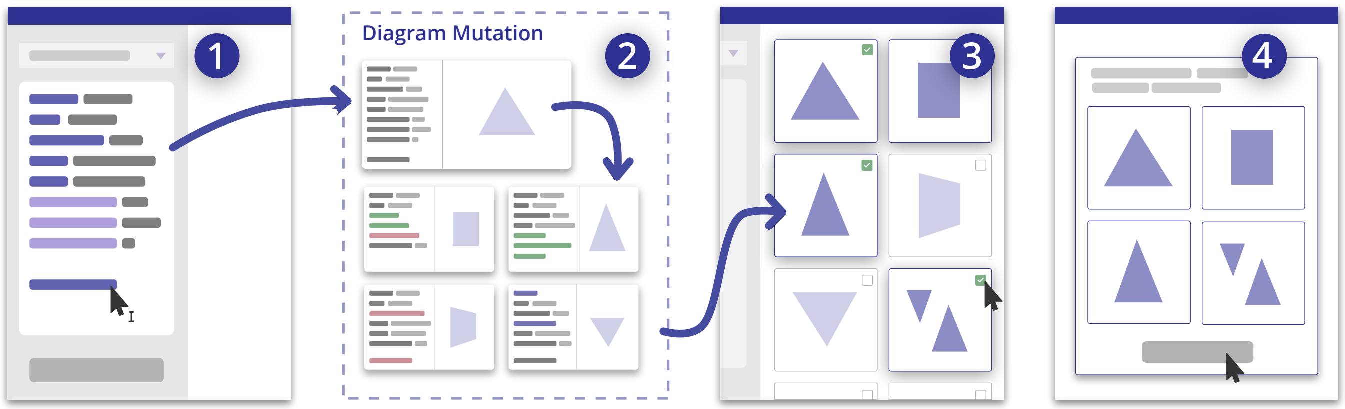


Figure 1: EDGEWORTH is a diagrammatic problem authoring tool that automatically generates diagram variations from a single diagram: the author creates an example diagram (1), then EDGEWORTH generates a myriad of diagram variations (2), from which the author selects diagrams (3) to form a diagrammatic multiple choice problem (4).

ABSTRACT

Visual thinking with diagrams is a crucial skill for learning and problem-solving in STEM subjects. To improve in this area, students need a variety of visual problems for deliberate practice. However, in our interviews, educators shared that they struggle to create these practice exercises because of limitations of existing tools. We introduce EDGEWORTH, a tool designed to help educators easily create visual problems. EDGEWORTH works in two main ways: firstly, it takes a single diagram from the user and systematically alters it to produce many variations, which the educator can then choose from to create multiple problems. Secondly, it automates the layout of diagrams, ensuring consistent high quality without the need for manual adjustments. To assess EDGEWORTH, we carried out case studies, a technical evaluation, and expert walkthrough demonstrations. We show that EDGEWORTH can create problems in three domains: geometry, chemistry, and discrete math. These problems were authored in just 15 lines of EDGEWORTH code on average. EDGEWORTH generated usable answer options within the first 10 diagram variations in 87% of authored problems. Finally, educators gave positive feedback on EDGEWORTH's utility and the real-world applicability of its outputs.

CCS CONCEPTS

• **Applied computing** → **E-learning**; • **Human-centered computing** → **Interactive systems and tools**.

KEYWORDS

Diagram Authoring, Diagrammatic Problems, Educational Content Authoring

L@S '24, July 18–20, 2024, Atlanta, GA, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0633-2/24/07.
https://doi.org/10.1145/3657604.3662034

ACM Reference Format:
Wode Ni, Sam Estep, Hwei-Shin Harriman, Kenneth R. Koedinger, and Joshua Sunshine. 2024. EDGEWORTH: Efficient and Scalable Authoring of Visual Thinking Activities. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale (L@S '24), July 18–20, 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3657604.3662034

1 INTRODUCTION

Mastery of visual thinking using diagrams enables more robust learning [39] and flexible problem solving [2, 7, 32, 33, 35, 45]. Deliberate practice with visual representations leads students to mastery [16, 41]. Traditionally, educators author visual practice by drawing diagrams by hand. In formative interviews (Section 3), educators reported the vital role of visual practice in their instruction, but noted the tedium of authoring due to tool limitations, leading to fewer diagrams used than desired. Manual authoring can hardly keep up with the growth of STEM learners and demand for more visual practice.

As a first step towards scaling up visual practice authoring, we built Edgeworth, a diagrammatic problem generator. Edgeworth generates *translation problems*, an effective type of visual practice [31] that ask students to determine diagrammatic *examples* and *counterexamples* of a textual/symbolic description (Figure 2). To help authors get the most out of one diagram, Edgeworth contributes a "build once, generate many" authoring paradigm: Instead of manually editing diagrams to get variations, the author creates a single diagram and Edgeworth automatically generates diagram variations (Figure 1 ① ②). The interaction design of Edgeworth allows the author to visually select diagram variations to rapidly form translation problems (Figure 1 ③ ④). Given the diversity of instructional contexts in STEM, we designed Edgeworth to be domain-agnostic: it uses a generic program mutation technique (Section 4.3) to change the author-provided diagram to produce variations.

To effectively scale up visual practice authoring, Edgeworth must support a diverse set of instructional domains, generate high-quality diagrams consistently, and allow educators to author real-world problems. In Sections 5 to 7, we evaluate Edgeworth by answering the following research questions on these qualities:

RQ1 **Expressiveness**: Can Edgeworth generate meaningful diagram mutants in multiple domains of instruction?
RQ2 **Reliability**: Can Edgeworth reliably generate translation problems with relatively few variations required?
RQ3 **Ecological validity**: Do real-world instructors consider Edgeworth-generated translation problems to be useful?

First, to show the expressiveness of Edgeworth, we performed case studies by recasting 31 problems from Euclidean geometry, discrete math, and general chemistry textbooks and courses (Section 5). Second, we evaluated the reliability of Edgeworth by labeling 310 diagram variations from the recasted problems by hand. With high inter-rater reliability, the result shows that Edgeworth can reliably generate diagrams that constitute valid four-choice translation problems, when constrained to 10 variations per problem.

Finally, we conducted walkthrough demonstrations with 9 educators that have experience creating problems. The goal of the demonstrations was to obtain feedback on the ecological validity of Edgeworth-generated problems and the usefulness of Edgeworth in general. Overall, these experts found Edgeworth-generated problems to contain pedagogically useful variations and high visual quality. They provided detailed feedback on individual diagram variations and suggested how Edgeworth might fit into their instructional contexts.

2 BACKGROUND AND RELATED WORK

In this section, we provide background on why students need to practice for better fluency in visual representations, how diagram variations help students practice, and how Edgeworth relates to existing problem generation tools.

2.1 Usefulness of Diagram Variations for Representational Fluency

Representational fluency refers to the ability to quickly understand a visual representation and to use it to solve domain-specific tasks [16, 41, 45]. To become representationally fluent, an important first step is to identify meaningful aspects of a particular representation. Mapping between symbolic and visual representations, which can be practiced with symbol-diagram translation problems, leads to intuitions about the way equivalent structures relate to each other [31, 40]. The learning that results from constructing connections between symbols and diagrams can be more flexible [22]. Students are better at transferring their learning from the problems they have explicitly practiced to novel problems [48], and their conceptual understanding is improved [26].

There is a lot of evidence in the learning science literature that using multiple examples and repeated practice are effective at teaching STEM skills. There are substantial STEM learning benefits for using multiple worked examples per topic [43]. Research also indicates the importance of active learning [12, 15] and repeated practice [18, 51] that occurs within varied contexts [42, 49]. In the context of acquiring representational fluency, diagram variations help students discern crucial parts of a particular representation [38]. The level of variation depends on where students are in the learning process: early on, students benefit from discerning examples and counterexamples that differ in only one dimension of variation. As students become more fluent, students may benefit from a *fusion* of multiple varying dimensions [13].

Rau [46] reports that, unfortunately, providing support for representational fluency is time-consuming with current tools. Our formative data (Section 3) confirmed this claim and revealed barriers resulting from the limitations of diagram authoring tools. To address these limitations, Edgeworth aims to simplify the workflow for creating diagram variations for repeated practice.

2.2 Tools for Problem Generation

Kurdi et al. [34] conduct a systematic review of automatic problem generation tools and show that the majority of tools address language learning. In this section, we focus on problem generation tools in STEM learning and discuss how they relate to diagrammatic problem generation and Edgeworth.

Intelligent Tutoring Systems (ITS) are automated curricula that include practice problems with personalized feedback (*inner loop*) and customize problem selection to students' performance (*outer loop*) [53]. Problem banks are an important component of ITS tools, so many systems have built-in authoring support to generate a large number of problems via templating. For instance, Cognitive Tutor Authoring Tools (CTAT) is an ITS authoring platform [3]. CTAT has a "Mass Production" feature that lets the user create a problem template and insert problem-specific values via a spreadsheet [4].

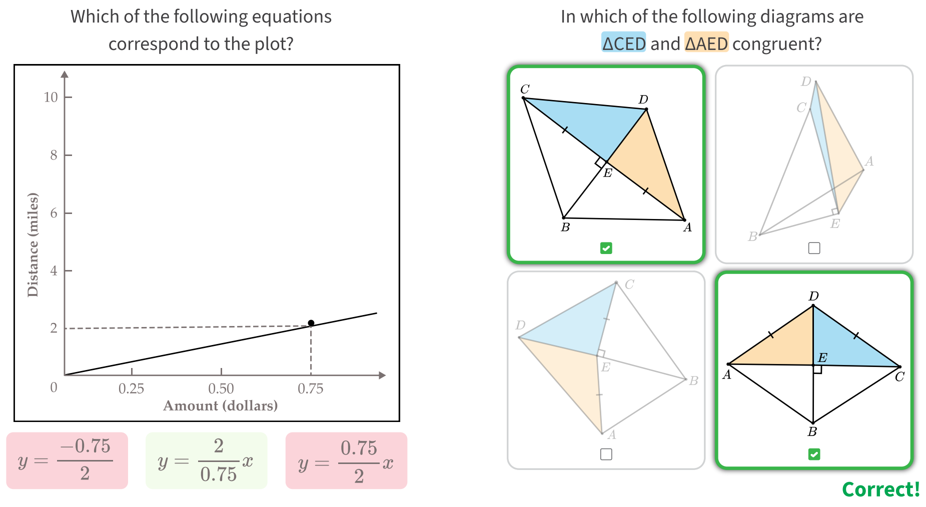


Figure 2: Left: A translation problem that helps students discern the structure of linear equations (adapted from Kellman et al. [31]). **Right:** An Edgeworth generated problem that trains student on triangle congruence theorems using diagram variations.

Similarly, the ASSISTment builder allows authors to "variabilize" numerical values in problem templates for automatic generation [47].

In the context of testing, researchers proposed systems that generate test problems (*items*) automatically for adaptive testing and cost-effectiveness [20]. Due to the need for numerous test items, automatic item generation systems also rely on templating (*item models*) to generate items [21, 27, 44]. For instance, IGOR [20, Chapter 13] has a similar approach to templating as CTAT and ASSISTment. While the templating approach is suitable for symbolic problems, they do not automate diagram generation. Authors still need to provide individual diagrams in templates in CTAT, ASSISTment, or IGOR.

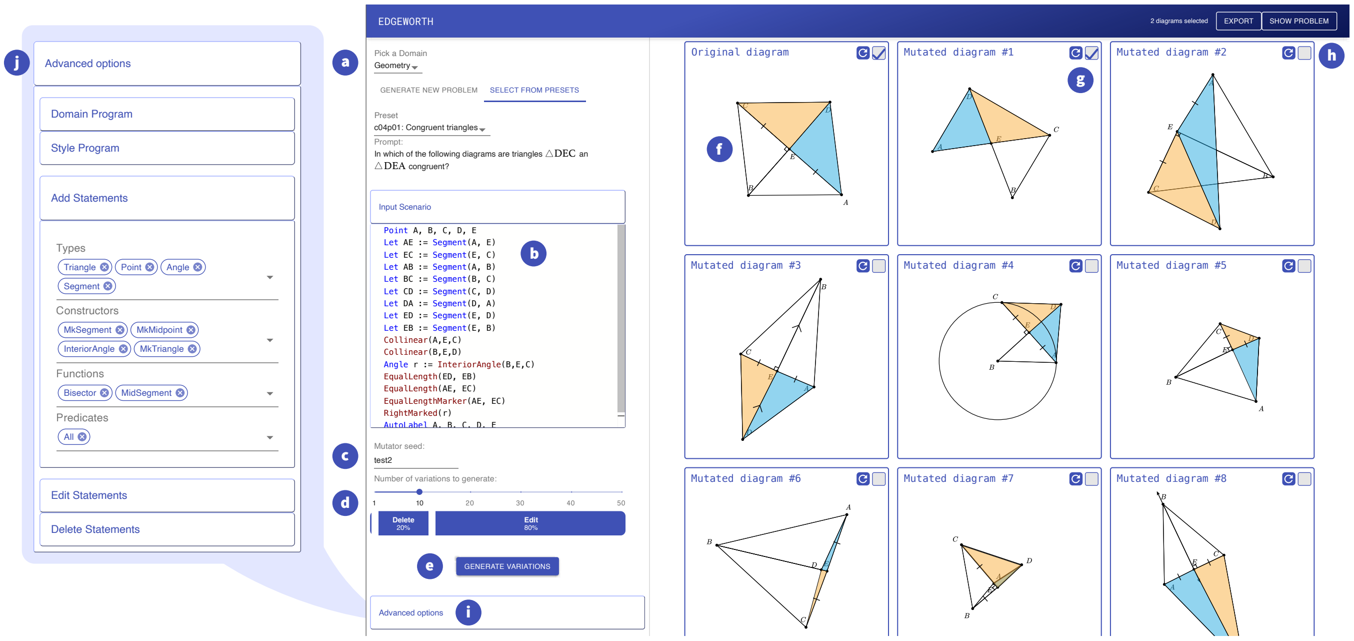
Edgeworth complements these tools by enabling authors to automate diagram variation production. Diagrammatic problems generated by Edgeworth can be integrated into problem banks and managed by the outer loop of ITS for an adaptive learning experience. Edgeworth does not currently support template variables in the textual prompt or diagram labels. However, it is possible to parameterize the example diagram as a problem template and use existing template-based systems to generate problem variations.

Other problem generation systems employ different methods from templating. A number of systems use *program synthesis* to synthesize a program that produces many problem instances [23]. Singh et al. [52] generate algebraic equality proof problems from example problems. Weitekamp et al. [55] speed up ITS authoring in CTAT by synthesizing ITS problems from user demonstration of problem solutions. Andersen et al. [6] model procedures to solve algebra problems as imperative programs and use execution traces of these programs to generate a series of problems. Notably, Gulwani et al. [24] generate solutions to geometry drawing problems by synthesizing programs of ruler-and-compass geometry constructions from a program specification. Though not strictly a problem generation tool, the generated solutions can be illustrated diagrammatically. However, the approach in [24] is specific to the domain of geometry, whereas Edgeworth's approach is domain-agnostic. Synthesis-based systems often have an advantage of a simpler user experience, since the author can provide examples and the tool automates problem generation itself. The approach of Edgeworth takes inspiration from these tools in that Edgeworth only requires the author to provide one example diagram. However, Edgeworth does not need to generate programs from a specification. It merely performs mutations on an example diagram.

Commonly used in human intelligence tests and as computer vision benchmarks, Figural Analogy Problems (FAPs) give a series of diagrams and ask the respondent to infer or select the next diagram given some patterns in the given diagrams [56]. Early automatic FAP generators were based on human-crafted shape composition rules [28] and cognitive models [17]. Newer systems [8, 54] encode variation rules [11] as first-order logic constraints. While FAPs are by definition highly diagrammatic, FAPs focus on pure visual reasoning, while in STEM problems often focus on mapping symbolic notations to visuals. Moreover, diagrams in STEM are much more diverse due to the multitude of disciplines, and are not limited by a few variation rules. That said, Edgeworth takes inspiration from FAP generators' rule-based approach. However, Edgeworth's mutations are domain-agnostic and operate on logical objects, not fragments of the diagram itself.

3 FORMATIVE INTERVIEW

We conducted semi-structured interviews with 6 educators to understand how they author, use, and maintain diagrammatic problems. We recruited participants based on their background in education and usage of diagrams in their work. Selected participants work as secondary school teachers, university professors, teaching assistants, and competitive math coaches. All participants (P1–6) indicated that they have experience creating instructional material, authoring problems, and/or developing online courses that include visual content. Example interview questions include what roles diagrams play in the participant's educational materials, how students interact with diagrams, and how diagrams are authored and



Pick a Domain: Geometry

Preset: c04p01: Congruent triangles

Prompt: In which of the following diagrams are triangles △DEC an △DEA congruent?

```
Point A, B, C, D, E
Let AE := Segment(A, E)
Let EC := Segment(E, C)
Let AB := Segment(A, B)
Let BC := Segment(B, C)
Let CD := Segment(C, D)
Let DA := Segment(D, A)
Let ED := Segment(E, D)
Let EB := Segment(E, B)
Collinear(A,E,C)
Collinear(B,E,D)
Angle r := InteriorAngle(B,E,C)
EqualLength(ED, EB)
EqualLength(AE, EC)
EqualLengthMarker(AE, EC)
RightMarked(r)
AutoLabel A, B, C, D, E
```

Mutator seed: test2

Number of variations to generate: 10

Figure 3: The user interface of Edgeworth. The author first provides a textual prompt (a) as an input scenario in Substance notation (b). Then, clicking "Generate Variations" (e) generates the specified number of diagram variations (d) at random based on a string seed and weights on Add, Delete or Edit mutations (c). In the diagram panel, the top-left diagram (f) corresponds to the input scenario and the rest are diagram variations generated by Edgeworth. The author can visually select diagrams (g) to assemble a diagrammatic multiple-choice problem (h). If needed, the author can fine-tune the mutator using "Advanced options" (i j).

maintained. The full interview protocol is included in supporting files.

Participants reported the usage of diagrams to build conceptual understanding and emphasized the need for deliberate practice to acquire representational fluency. Traditional educational materials, especially in higher education, tend to emphasize "procedures, memorization, and symbolic manipulation" (P6). Similarly, teachers such as P1 suffer from "the curse of knowledge" of teaching visual fluency: teachers tend to "under-train" students and they struggle to use visuals for problem-solving. As a result, students often become "symbolically good" and do not develop "good conceptual understanding" (P3). Visuals like diagrams and graphs provide alternative representations that help students "develop intuition" (P3) and "become better problem-solvers" (P4). To improve their instruction, all of our participants (P1–6) attempt to incorporate more diagrams in their instructional materials. Some also ask students to draw, annotate, and explain diagrams (P1, P2, P6). P2 encourages students to learn "multiple representations" and makes diagrams central to their math and programming curricula. When students practice with diagrams, teachers also gain richer feedback on students' level of understanding, and "learned more from this [student-drawn diagram] than 10 similar problems without the pictures" (P6).

While the benefits of and need for diagrammatic practice are clear, participants reported that tool limitations led to manual and repetitive authoring experience. Because participants typically create many problems and iterate on their content often, they face a trade-off when authoring visual content: more visuals are beneficial for learning but are time-consuming to create and modify. When authoring practice problems, P1 struggled to "create simple shapes by myself" and always ended up "copy-pasting and searching online" repeatedly. Similarly, P6 reported that they "get online images for pre-made resources, but whenever I want something a little custom, it'll take a lot of time." To streamline the visual authoring process, P2 and P5 developed custom pipelines for authoring problem sets and quizzes using existing programming tools. Like the problems described by prior research on diagramming tool usability [37], these tools often lack support for "high-level tweaking of my diagrams" (P2) and "are a pain to use because the language is not semantic and hard to use for non-programmers" (P5). Participants showed us many examples of tedious changes necessary to create diagram variations.

From the results, we derived the following design requirements for tool design to address participants' needs:

- **D1** Address the need for practicing representational fluency
- **D2** Simplify the workflow for generating diagram variations
- **D3** Obviate the need to attend to low-level diagramming details

4 SYSTEM DESIGN OF EDGEWORTH

Edgeworth realizes the design goals from Section 3 by: 1) providing a domain-agnostic workflow for rapidly authoring diagrammatic practice problems (**D1**), 2) automatically suggesting numerous diagram variations of a single example diagram and allowing the author to visually select from the variations (**D2**), and 3) fully automating the layout for all diagram variations (**D3**). Figure 3 walks through the user interface of EDGEWORTH, a simple and clean design that encapsulates the ideas above.

In Section 4.1, we demonstrate the workflow of EDGEWORTH by showing how to author an example diagrammatic problem in Euclidean geometry. We then describe EDGEWORTH's approach to diagram layout in Section 4.2 and how it generates diagram variation in Section 4.3.

4.1 Author Workflow

In this section, we use an example from high school geometry to demonstrate the process of creating a problem in EDGEWORTH.

4.1.1 Create an example diagram. The author wants to write a problem about triangle congruence to assess students' understanding of the *Side-Angle-Side* (SAS) rule. They want to create a translation problem including one diagram where the SAS rule is satisfied and three others where it is not. The author first describes an example diagram (Figure 3ⓑ) where this rule is satisfied. They construct a scenario involving two triangles: $\triangle DEC$ and $\triangle DEA$ share one side DE and have two equal sides EC and EA. $\angle CEB$ indicates that AC and BD are perpendicular and therefore $\angle DEC = \angle DEA$. Therefore, $\triangle DEC$ and $\triangle DEA$ are congruent by the SAS rule. Given this description, EDGEWORTH lays out the diagram automatically (Figure 3ⓕ).

4.1.2 Select from EDGEWORTH-generated diagrams. Now the author can use EDGEWORTH to mutate the example diagram by clicking "Generate Variations" (Figure 3ⓔ). EDGEWORTH performs mutations on the example scenario and generates a grid of diagram variations. The grid is designed to give the author an overview of the mutation results, and diagrams are prominent in each cell to facilitate faster visual selection. The top-left cell in the grid will always display the original example diagram (Figure 3ⓑⓕ), and the rest correspond to mutation results.

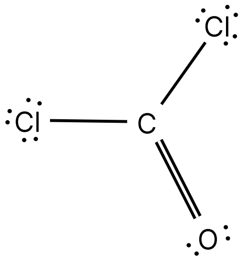
By inspecting each diagram in the grid, the author can determine if it is a good fit for their translation problem. If so, they click the top-right checkbox (Figure 3ⓖ) to include the diagram in the problem.

4.1.3 Preview and export the problem. After the author picks a sufficient number of diagrams (4 in this case), they can preview the translation problem by clicking "Show Problem" (Figure 3ⓗ), which displays an interactive multiple-choice widget. If the author is satisfied, they can click "Export" to download the diagrams and metadata to use the problem in their context. EDGEWORTH exports to Scalable Vector Graphics (SVG) images for static media, source programs for interactive use, and detailed mutation trace metadata for comprehensive analysis and reference purposes.

4.2 Diagram Notation and Layout

EDGEWORTH is built on PENROSE, an open-source diagram format and layout engine [57]. Compared with alternatives, PENROSE offers two advantages: (1) a high-level diagram notation that's easy for authoring and (2) an automatic layout engine. A diagram in PENROSE consists of a textual description of the diagram content (SUBSTANCE) and a reusable layout stylesheet.

SUBSTANCE is a simple declarative notation for describing objects and relations in a diagram. As shown in Figure 4, SUBSTANCE has three kinds of statements: type statements (e.g., `Carbon c`) declare new objects; constructors (`Bond b1 := MakeSingleBond(c, cl1)`) create new objects from existing objects; and predicates (`ZeroValenceElectrons(c)`) indicate relations among objects. A stylesheet translates the SUBSTANCE notation to shapes and layout constraints, and then PENROSE solves for diagram layouts automatically. Since EDGEWORTH authors only interact with SUBSTANCE, we omit the description of the stylesheet language in this paper; more details about stylesheets can be found in the official documentation[1] and Ye et al. [57].



```
Carbon c
Oxygen o
Chlorine cl1, cl2
Bond b1 := MakeSingleBond(c, cl1)
Bond b2 := MakeSingleBond(c, cl2)
Bond b3 := MakeDoubleBond(c, o)
ZeroValenceElectrons(c)
FourValenceElectrons(o)
SixValenceElectrons(cl1)
SixValenceElectrons(cl2)
```

Figure 4: Diagram and SUBSTANCE notation for the Lewis structure of phosgene ($COCl_2$).

The PENROSE ecosystem offers a wide range of stylesheets for STEM diagrams, and the current EDGEWORTH implementation builds on PENROSE's geometry, chemistry, and graph stylesheets for diagram layout. Since the existing PENROSE stylesheets are primarily used to generate a few human-written examples, they lack coverage for variations of SUBSTANCE descriptions required by EDGEWORTH. To this end, we have contributed our improved stylesheets, diagram examples, and new standard library functions to PENROSE.

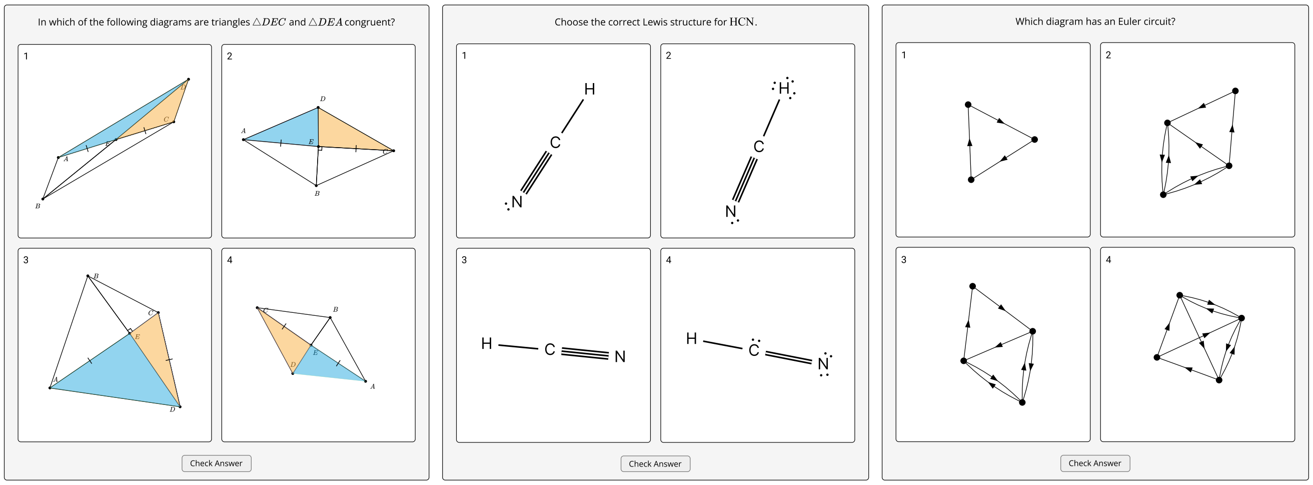
EDGEWORTH is the first application of PENROSE that concurrently optimizes and renders a grid of multiple diagrams. Therefore, we have made significant upstream contributions to PENROSE to support EDGEWORTH's use case. To make EDGEWORTH a performant client-side web application for interactive use, we have contributed to the migration from Haskell to TypeScript and various performance improvements to efficiently run tens of layout optimization jobs in a single session. Compared to the state of PENROSE at the publication of Ye et al. [57], our contributions have helped improve the performance of the system by 100×.

4.3 Program Mutation

EDGEWORTH generates diagram variations by mutating the example diagram written in SUBSTANCE. We purposely designed the system to include a small set of simple and type-safe mutation operations. Similar to generic tree-editing algorithms [19], EDGEWORTH supports 3 kinds of mutation operators: **Add**, **Delete**, and **Edit**. **Add** appends a statement. **Delete** removes a statement and all other references to that statement.

Since compilation errors in SUBSTANCE will not produce diagrams, **Edit** involves one of the type-safe patterns listed below. Each **Edit** pattern contains a *guard* and an *action*. The guard checks if the operator is applicable to the given SUBSTANCE statement, and

[1] https://penrose.cs.cmu.edu/docs



In which of the following diagrams are triangles $\triangle DEC$ and $\triangle DEA$ congruent?

Choose the correct Lewis structure for HCN.

Which diagram has an Euler circuit?

Figure 5: We used Edgeworth to recast real-world problems as diagrammatic translation problems. Left: Determine if triangles are congruent. **Middle:** Identify the correct Lewis structure for hydrogen cyanide. **Right:** Identify graphs with Euler circuits.

the action performs the mutation. For instance, **Replace Arguments** is only applicable when the current context has existing variables of the desired type.

- **Swap Arguments** reorders the arguments passed into a statement; e.g., if `A` and `B` are `Triangles`:
  `Similar(A, B)` ⟶ `Similar(B, A)`
- **Replace Arguments** replaces the arguments passed into a statement with other arguments defined in scope; e.g., if `A, B, C, D` are `Points`:
  `s := MkSegment(A, B)` ⟶ `s := MkSegment(C, D)`
- **Replace Function** replaces a statement with a different statement that takes the same arguments; e.g., if `T` is a `Triangle` and `E` is an `Angle`:
  `Equilateral(T)` ⟶ `Scalene(T)`
  `Segment s := Bisector(E)` ⟶ `RightAngleMarked(E)`

Algorithm 1 The Edgeworth mutation algorithm.

```
 1: function GENERATE($p, \ell, h, a, d, e, A, D, E$)
 2:     $p' \leftarrow p$
 3:     $n \leftarrow$ uniform random integer between $\ell$ and $h$
 4:     for $i$ from 1 to $n$ do
 5:         $x \leftarrow$ uniform random real between 0 and $a + d + e$
 6:         if $x < a$ then
 7:             $m \leftarrow$ RANDOMADD($A, p'$)
 8:         else if $x < a + d$ then
 9:             $m \leftarrow$ RANDOMDELETE($D, p'$)
10:         else
11:             $s \leftarrow$ uniform random element of STATEMENTS($p'$)
12:             $m \leftarrow$ RANDOMEDIT($E, s$)
13:         end if
14:         $p' \leftarrow$ MUTATE($p', m$)
15:     end for
16:     return $p'$
17: end function
```

Algorithm 1 shows how the Edgeworth mutator works, at a high level. In addition to the input Substance description p, Edgeworth also takes a number of user-defined configuration parameters: (1) a number of variations to generate (the number of times Generate is called); (2) a range of mutation counts per variation (the input variables ℓ and h); (3) weights for **Add**, **Delete**, and **Edit** operations (the input variables a, d, and e respectively); and (4) filter sets A, D, and E which limit the set of mutations that the **Add**, **Delete**, and **Edit** operations can produce.

Given an example diagram, Edgeworth performs several rounds of mutation generation. Each round results in a series of mutations that alter the input to produce a variation. The number of mutations (line 3) is bounded by the configuration parameters.

To generate a single mutation, Edgeworth makes a weighted choice (line 5) of the mutation kinds and enumerates all possible mutations for the chosen kind: **Add** enumerates all possible statements to add (line 7); **Delete** randomly deletes an existing statement (line 9); **Edit** enumerates all possible edits for all statements (line 11) and picks one of them randomly (line 12). The randomness of Edgeworth is controlled by a single random generator seed.

Users can specify filter sets under the "Advanced options" section of the UI, shown in Figure 3 i j. The filters default to "All," which indicates that the mutator may change any statement in the example diagram. While this precise configuration may be useful, we ended up not using them in our evaluation (Sections 5, 6, and 7) and instead achieving our results using only Edgeworth's simpler core set of configuration options, i.e., weights on mutation operators.

5 EXPRESSIVENESS EVALUATION IN THREE DOMAINS (RQ1)

Edgeworth's mutation-based approach is domain-agnostic: it simply applies generic program mutations on any Substance program. Through case studies of Euclidean geometry, general chemistry, and discrete mathematics, we evaluate if this approach is expressive enough for different instructional contexts in STEM. The 3 domains are selected based on their ubiquity in STEM education and visual representations. All three domains have a wide audience in K-12 and higher education, making them rich sources for existing instructional materials. Each domain has canonical visual representations that are explicitly taught to students. Therefore, students can benefit from visual practice in these domains.

We choose problems from existing textbooks or online courses and follow the procedure outlined in Section 4.1 to recast each problem. All problems are included in supporting files.

5.1 Summary Statistics

In the case studies, we reproduced 31 problems in total. Since creating the example diagram (Section 4.1.1) took the most time in this process, we report statistics on the example diagrams here.

On average, Edgeworth's diagram notation is compact and simple. The description for example diagrams are 14.7 lines of code ($\sigma = 4.57$) and 109.9 tokens ($\sigma = 48.6$). In contrast, the average SVG source of these same diagrams have 454.7 lines of code ($\sigma = 184.3$) and 1290.4 tokens ($\sigma = 650.4$). This indicates that Edgeworth provides a concise and compact textual representation of diagrams across all three domains.

5.2 Euclidean Geometry

We sample 17 Euclidean geometry problems from Holt *Geometry* [10], a high school geometry textbook. Figure 5 (left) shows an example problem. The textbook uses a consistent visual style of predominantly black line segments and dots with text labels. Most diagrammatic problems are presented as one diagram followed by one or more multiple-choice problems. We've recast the problems as diagrammatic translation problems.

For this domain, we build on the existing geometry stylesheet from Penrose [57, Section 5.3] for diagram layout. In this domain, Edgeworth weights deletions 20% and edits 80%. There are many different types of entities in geometry, so additions tend to introduce elements to the diagram that obviously do not pertain to the question prompt. Thus in this domain, the Edgeworth mutator applies no additions. The reason we weight edits higher than deletions is that many of our geometry problems ask about specific named points, and deletions can make the diagram invalid by removing points that are mentioned in the prompt.

We use the problem in Figure 5 (left) to demonstrate how Edgeworth generates variations that are meaningful as problem options. In the diagram shown, $\triangle DEC$ and $\triangle DEA$ are congruent by the Side-Angle-Side rule. In particular, they share a side (DE), the sides AE and EC appear to have equal length and are marked as such with a tick, and $\angle DEA$ and $\angle DEC$ are both right angles and therefore equal. Option 4 in Figure 5 involves mutating the scenario by removing the right angle marker which makes it impossible to prove that $\angle DEA$ and $\angle DEC$ are equal. This is an example of the **Delete** mutation described in Section 4.3. The angle appears to be a right angle in Option 3, so this option might serve as a good distractor for students still learning the distinction between the appearance of angles and their markings.

Option 3 involves mutating Option 2 by editing which sides have equal length. In Option 3, sides CD and AE are equal instead of AE and CE. This is an example of the **Replace Arguments** mutation

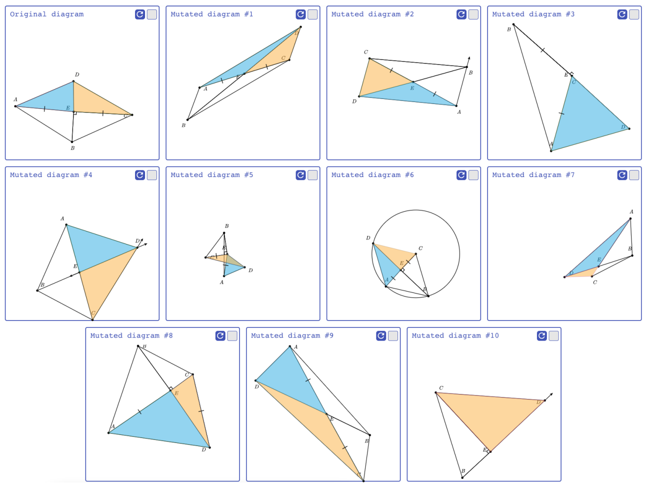


Figure 6: The first ten diagram variations generated by Edgeworth for the problem shown in Figure 5 (left).

described in Section 4.3. A student might incorrectly select Option 3 if they believed in a Side-Angle congruence rule, where a single angle and single side being equal could prove congruence. Finally, in Option 1 $\angle CEB$ neither is marked as a right angle nor appears as a right angle. A student might incorrectly select Option 1 if they believed in a Side-Side congruence rule, where two sides being equal could prove congruence.

Figure 6 shows the first 10 variations Edgeworth generated from the example diagram. To create our problem, shown in Figure 5 (left), we selected the original diagram and two incorrect variations (numbers 1 and 8), plus another variation in an extended pool (number 16). As shown in Figure 6, there are many other viable answer choices in the first 10 variations. Many of the diagrams involve extra details that are irrelevant to the problem, like the circle in number 6 or the vector above point B in number 2. These extra details can be pedagogically useful for teaching students to filter irrelevant information in the domain. Some of the other diagrams are very obviously incorrect, like number 10 which doesn't show a blue triangle, or number 7 where the blue triangle is much larger than the orange triangle; these can be useful for building confidence when students are first learning.

5.3 General Chemistry: Lewis Structures

We chose 7 chemistry problems on Lewis structure from an online General Chemistry 1 course [1]. These problems test students' understanding of how atoms bond together based on formal charges. The module introduces students to the *octet rule*: the tendency of main group atoms to form enough bonds to obtain eight valence electrons. Lewis structure diagrams show bonds among atoms and valence electrons on atoms typically following the octet rule.

We extend the existing Penrose chemistry stylesheet to include notation and layout rules for Lewis structures. To permit incorrect diagrams, the chemistry stylesheet must not enforce the octet rule. It does specify that an atom can have any number of bonds and that it can have 0, 2, 4, or 6 valence electrons. These specifications cover all problem scenarios in this Lewis structure module.[2] In accordance with stylistic conventions in the field, Edgeworth automatically lays out atoms, bonds, and electrons to maximize bond angles and repel electrons from bonds . For molecules involved in all 7 problems, the layout algorithm produces high-quality diagrams without any manual manipulation needed from the author.

To configure Edgeworth for this domain, we weight edits 100%. We exclusively weight on edits because we observed that variations of molecules never add or delete atoms and bonds. Although valence electrons may be added or deleted, they are modeled as predicates that can be edited to change the number of electrons for an atom (e.g., `ZeroValenceElectrons(H)` $\longrightarrow$ `TwoValenceElectrons(H)`) via a **Replace Function** mutation operation.

Figure 5 (middle) shows an Edgeworth Lewis structure problem for hydrogen cyanide, with the correct diagram in the top-left. Incorrect choices for this problem can be generated via mutation. For instance, if a student forgets that nitrogen must have eight surrounding electrons, they might choose the bottom-left option, which was generated by removing the valence electrons around nitrogen. Or, if a student does not know that hydrogen should only have two electrons instead of eight, they might select the top-right choice, which was generated by mutating the number of electrons around hydrogen from zero to six. Finally, if a student does not know that free electrons should be minimized, they might pick the bottom-right diagram, which was generated by mutating up the number of electrons around carbon and nitrogen and changing the triple bond to a double bond.

5.4 Discrete Math: Graphs

We draw 7 graph theory problems from the "Graphs" chapter of *Discrete Mathematics and Its Applications* [50, Chapter 10]. We model our visual representation after the style used in the textbook, allowing students to recognize Edgeworth-generated diagrams as they are already accustomed to recognizing graph diagrams. We created a new Penrose stylesheet for four subdomains of graphs (directed vs not, and multigraph vs not). For each of these subdomains, Edgeworth automatically lays out graph nodes, edges, loops, arrows, and labels in configurations that minimize confusing overlap of diagram elements. As with the other domains, no manual tweaking is necessary to obtain high-quality diagrams for problem variations.

To configure Edgeworth for the graph domain, we weight additions 50%, deletions 40%, and edits 10%. We disfavor edits in this domain because most of them are not useful: **Replace Function** is inapplicable for any of our graph subdomains, and **Swap Arguments** only applies to directed graphs. **Replace Arguments** is meaningful, but most desirable mutations for graphs are better represented by the addition or deletion of edges, or sometimes nodes. For instance, a bipartite graph can become not-so by adding edges, or a strongly-connected graph can become not-so by deleting edges.

Figure 5 (right) shows an Edgeworth problem asking which of four directed graphs have an Euler circuit. The bottom-right diagram does not have an Euler circuit, as can be seen by observing that the sum of a's in-degree and out-degree is odd. In contrast, for the diagram in the bottom-left generated by deleting edge (a, d), every node has an even sum of in-degree and out-degree, and indeed there does exist an Euler circuit. This condition on degree is only sufficient for undirected graphs, though; the diagram in the top-right is generated by flipping edge (b, c) from the bottom-left diagram, but does not have an Euler circuit, thwarting the simple degree counting heuristic. Finally, the simple diagram in the top-left is generated by deleting d and trivially has an Euler circuit.

6 RELIABILITY EVALUATION (RQ2)

Edgeworth's approach involves random mutations. The mutation operations are type-safe, but type-safety does not prevent degenerate diagram layouts. For instance, `Point A, B; Triangle t := MkTriangle(A, A, B)` typechecks. However, since the triangle described in this scenario involves the `Point A` twice, Edgeworth will produce a line segment, not a triangle from this scenario. Are Edgeworth suggestions dominated by these nonsensical scenarios? In this section, we evaluate whether Edgeworth can reliably suggest diagrams that are valid answer options to multiple-choice translation problems.

6.1 Methods

The goal of Edgeworth is to generate enough diagram variations to assemble a four-choice multiple-choice problem for a given prompt. To this end, we use the following classification scheme for diagram variations: a variation can be a **Correct** or **Incorrect** answer to the prompt, or **Discard**ed because the diagram is invalid for missing key components or lacking readability.

For RQ2, we define "relatively few variations" to be 10 diagrams, and consider Edgeworth to have generated a translation problem in n variations if at that point we have (possibly including the original diagram) at least one **Correct** diagram, at least one **Incorrect** diagram, and in total at least four diagrams that are either **Correct** or **Incorrect**.

To evaluate this coding scheme, we randomly sampled 2 problems from each of our 3 domains, for 60 generated diagrams total. The first two authors each coded all 60 of those sample diagrams, after which we calculated the Cohen's κ [14] statistic. Then with the assumption that our coding scheme has reasonable inter-rater reliability, at least one author coded all remaining diagrams, allowing us to determine the number of our prompts for which Edgeworth was able to successfully generate a multiple-choice problem. The coding results are included in supporting files.

6.2 Results

6.2.1 Reliability of problem generation. For RQ2, we found that Edgeworth generated valid multiple-choice problems for 27/31 prompts within 10 variations, and for 30/31 problems within 20 variations. For each of these four failures with 10 variations, Edgeworth did generate at least four **Correct** examples, but we had to **Discard** all the other diagrams, leaving no **Incorrect** examples. For the one remaining failure with 20 variations, Edgeworth never succeeded even after we increased the number of variations to 50.

6.2.2 Distribution. The original diagram is a **Correct** answer for every prompt, except for the two Euler circuit prompts, in which the original diagram is **Incorrect**. For Edgeworth-generated variations, the full distribution of classes is shown in Table 1.

[2]"Odd electron molecules are very rare and cannot achieve full octets of electrons around atoms because of the odd number of electrons." [1]

	Correct	**Incorrect**	**Discard**	*total*
geometry	52	54	64	170
chemistry	3	54	13	70
discrete	28	25	17	70
total	85	133	94	310

Table 1: Distribution of diagram variation classes.

The chemistry domain had a far smaller proportion of **Correct** variations than the other two domains because the only way for a variation to be **Correct** is for it to coincidentally be identical to the original diagram. Interestingly, in the other two domains, there were about the same number of **Correct** and **Incorrect** variations.

In the geometry domain, **Discard**ed diagrams were primarily either diagrams missing elements referred to in the question prompt, or diagrams that were visually degenerate (e.g., everything compressed into a single line). In chemistry, we **Discard**ed diagrams where the molecule was disconnected. Finally, in the graph domain, we **Discard**ed diagrams in which some nodes were labeled and others were unlabeled (i.e., Edgeworth had inserted new unlabeled nodes when all nodes in the original diagram were labeled).

6.2.3 Inter-rater Agreement. We sampled two problems per domain from the problems collected in Section 5 to evaluate inter-rater agreement (six problems total, 19% of the dataset). We found perfect agreement on that sample, so $\kappa = 1$.

7 EXPERT WALKTHROUGH DEMONSTRATION AND FEEDBACK

The intended users of Edgeworth are educators who create problems. These users are very important to the education system since other teachers make use of their problems. Therefore, we recruited educators who created visual practice problems in multiple domains and educational settings to evaluate ecological validity of Edgeworth-generated problems (Item RQ3). While an expert survey may suffice for rating problem quality, we opted for walkthrough demonstration, based on prior research on evaluation methods by Ledo et al. [36], to gather additional qualitative feedback on the value of having the toolkit in their day-to-day work.

7.1 Participants and Procedure

We recruited domain expert educators of chemistry, geometry, and graph theory. Experts were invited based on their extensive teaching experience in the domain and past experience in *authoring* diagrammatic content. In contrast to the criteria in the formative study (Section 3), this study selected participants based on their domain-specific expertise in authoring problems. Recruited educators came from a wide range of institutions, including MOOC platforms, liberal arts colleges, community colleges, research universities, and secondary schools. The average teaching experience among the 9 expert educators (E1–E9) was 10.33 years, with a standard deviation of 8.39 years, highlighting a broad range of teaching experience. One of the participants is the author of problems in one of the domains in Section 5.

Each expert participated in a 60- to 90-minute session via video conferencing, which was recorded with their consent. At the start of each session, we demonstrated the workflow of Edgeworth end-to-end, as described in Section 4.1, on one problem outside of the expert's domain. For the remainder of the session, we asked the expert to evaluate two to four problems sampled from Section 5 in their domain. Per problem, the expert rated 10 diagram variations based on the categories described in Section 6.1. In addition, we asked participants to provide more granular feedback on diagram quality. After rating the diagram variations, they were asked to pick diagrams to assemble a four-choice diagrammatic translation problem. After the problem was assembled and shown on the interface, we asked (1) if they would use the problem in their instruction and (2) how they would author the diagram using their own workflow. The full study protocol is included in supporting files.

7.2 Ecological Validity of Generated Problems (RQ3)

Overall, experts were happy with the problems they assembled with Edgeworth-generated diagrams. Experts (E1–9) indicated that they would use all of the problems they created using Edgeworth in their coursework. Other experts said they would use Edgeworth-generated problems "early in the learning process" (E3) and "as a warm up exercise at the start of the next lecture" (E4). In addition, the problems can be used to review previously introduced concepts. For example, E3 found the diagram variations that break the octet rule to be useful for "after you've also introduced expanded octet or non-octet-rule things." Experts plan to use Edgeworth-generated problem to "focus on things that students struggle with" (E3) and when introducing concepts that are "all about visualization" (E5) such as planarity of graphs. E7 was excited to use problems we created in the expressiveness evaluation (Section 5) in their class because they were "going to be covering everything [on the list]." In addition to just asking students to select correct diagrams, E3 also pointed out that by prompting students to "tell me what is wrong rather than just which is the correct one," the problem can be used to "dive deeper." Similarly, E4 proposed to use Edgeworth problems as "an interactive warm-up for reviewing the last lecture, where students vote on and explain why a diagram is correct." E7 even plans to use Edgeworth as "a creative instead of assessment piece" and "have the students be the teacher … playing this role more, they get better at tests, because they understand what the test makers are doing."

7.3 Expert Feedback

Experts reacted positively to Edgeworth. They found Edgeworth to be a "perfect fit" (E1, E6, E8) for generating multiple-choice problems, especially "low-stake" (E2, E3, E5, E6, E8, E9) quizzes that "incentivize [students] to keep up with the class" (E8). Experts said the automatic layout of Edgeworth "draws things really fast" (E5), "saves you the time of drawing multiple structures" (E3), and produces "beautiful" (E4, E7) diagrams. Comparing with their existing tools, Edgeworth is a "nice time-saver" (E3) and the translation problems they authored during the session would take an "enormous amount of work" (E4), "infinitely longer than this took" (E6).

Notably, experts pointed out that Edgeworth aids creativity by promoting "recognition over recall" (E6). Specifically, Edgeworth helps with "the thinking about how to come up with the graphs" and simplifies the diagram layout such that "you just generate some mutations that you click refresh until it looks nice" (E6). E2 liked that "it can come up with different possibilities than the ones that would be immediately apparent to me."

In addition, experts commented that EDGEWORTH can enable them to give students more practice. For instance, E4 noted that "there's a feedback loop where … if I had a really good tool for generating nice multi-choice questions, then I could envision doing that much more frequently."

EDGEWORTH sometimes produces isomorphic diagrams, i.e., diagrams with identical content but different layouts. These diagrams occur when EDGEWORTH's mutations have no net impact on the example diagram, e.g., the mutator removes an edge from a graph and adds it back. Surprisingly, experts found value in these isomorphic diagrams. In their geometry course, E2 said that their textbook's diagrams "get drawn the same way over and over again. And some students get stuck into thinking that the concept is only communicated when the diagram is drawn [exactly] that way." When assembling a problem about the *HCN* molecule, E3 compared two isomorphic variations, and picked one over another because "it's drawn the opposite …which is interesting and I think students are going to get it wrong." Similarly, E1 finds isomorphic diagrams to be useful for "molecules with resonance structures." E5 found isomorphic planar graphs to be particularly useful because students find them "painstaking to visualize when they just started." E5 planned to use EDGEWORTH to "draw a graph that doesn't look like it could be planar first, but then untangle it to show that the graph is actually planar."

8 CONCLUSION AND FUTURE WORK

To address the need for visual practice materials in diverse STEM domains (**D1**), we designed EDGEWORTH to use a domain-agnostic approach to generate diagrammatic problems. Using EDGEWORTH, the author is only responsible for creating a single diagram. From this diagram, EDGEWORTH generates diagram variations and fully automates diagram layout (**D3**). The author visually selects variations to assemble a problem with multiple diagram variations (**D2**).

To evaluate if EDGEWORTH can express multiple STEM domains (RQ1), we conducted case studies in geometry, general chemistry, and discrete mathematics (Section 5) to show that EDGEWORTH can express real-world problems and create pedagogically meaningful variations to the example diagrams. The diagrams generated by EDGEWORTH also need to have consistently high visual quality and content diversity (RQ2). To understand if EDGEWORTH meets this goal, we analyzed EDGEWORTH-generated diagrams for all problems from the case studies (Section 6). The results show that EDGEWORTH can reliably generate diagrams suitable for 27/31 four-choice diagrammatic problems within 10 variations, and 30/31 problems within 20 variations. Finally, we conducted walkthrough demonstrations with 9 domain expert educators to understand the ecological validity of EDGEWORTH-generated problems (RQ3) and get holistic feedback on the tool (Section 7). Experts indicated that EDGEWORTH generates useful diagrammatic problems for their instructional contexts. Moreover, they commented on how EDGEWORTH can speed up the problem design process and thereby scale up visual practice in classrooms.

We now discuss potential future directions for EDGEWORTH.

8.1 Mixed-Initiative Integration with AI

We believe EDGEWORTH is both a product of existing AI techniques and a promising platform to assess both domain-specific and general-purpose AI technologies in visual practice authoring. Like many classical AI systems, EDGEWORTH makes use of a symbolic description language (Section 4.2) and mutates the description of the example diagram to search for viable variations. The description language then generates layout constraints that compile to energy functions, the gradients of which drive an optimizer to arrange the diagram layout. In the educational setting, EDGEWORTH provides a mixed-initiative [5] experience: authors focus on specifying the content and the general direction of variations, while EDGEWORTH fully automates the details of variation generation and layout.

Moreover, the high-level description language of EDGEWORTH promotes a potentially more robust integration with LLMs. Current methods of automatic diagram generation using LLMs mainly generate low-level SVG elements and often "fail to maintain accurate geometric relations or only generating outputs of limited complexity such as single icons or font characters" [9]. Jain et al. [29] showed that GPT-4 does not do a good job of generating low-level visual code like SVG. In contrast, when prompted systematically, GPT-4 can reliably generate higher-level PENROSE programs which yield correct and legible diagrams.

A promising direction for future work is to augment EDGEWORTH with an LLM. When authoring a diagram, like the example diagram described in Section 4.1.1, the EDGEWORTH user can specify the diagram in natural language and this augmented version of EDGEWORTH can prompt an LLM to generate the example diagram in EDGEWORTH's notation.

8.2 Towards an abundance of adaptable visual learning materials

In 1954, Jacques Hadamard studied how mathematicians and physicists work, and many of them reported they worked by reasoning about hard problems in visual terms [25]. When citing Hadamard in his 1987 talk, Alan Kay lamented the fact that while experts "do their thing" visually, students are still learning symbolically [30]. We found that the educators we interviewed echoed Kay's concerns (Section 3). Notably, educators spend significant effort crafting visual learning materials that suit their needs in the classroom. We believe this effort means much more than copy-pasting and low-level tweaking of shapes in a diagram. Instead, educators encode their teaching context and their expertise in this process. Computational tools should provide enough support to provide better ergonomics for the authoring and adaptation of visual learning materials. As our first step, we built EDGEWORTH to let educators use one example diagram as the leverage to generate variations of diagrammatic multiple choice problems. There are ample opportunities to use EDGEWORTH to create *problem variations*, too. By simply increasing the number of variations and/or using a variation as a new example diagram, the author can use EDGEWORTH to generate diagrams for related problems on the same topic. We plan to study how to leverage EDGEWORTH's scalable diagram production for instructional contexts of larger scale.

REFERENCES

[1] [n. d.]. General Chemistry 1. https://oli.cmu.edu/courses/general-chemistry-1/

[2] Shaaron Ainsworth, Vaughan Prain, and Russell Tytler. 2011. Drawing to learn in science. *Science* 333, 6046 (8 2011), 1096–1097. https://doi.org/10.1126/SCIENCE.1204153/SUPPL{_}FILE/1204153.AINSWORTH.SOM.PDF

[3] Vincent Aleven, Bruce M. McLaren, Jonathan Sewall, and Kenneth R. Koedinger. 2006. The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In *International Conference on Intelligent Tutoring Systems*, Vol. 4053 LNCS. Springer, Springer, Berlin, Heidelberg, Jhongli, 61–70. https://doi.org/10.1007/11774303{_}7

[4] Vincent Aleven, Jonathan Sewall, Bruce M. McLaren, and Kenneth R. Koedinger. 2006. Rapid authoring of Intelligent Tutors for real-world and experimental use. *Proceedings - Sixth International Conference on Advanced Learning Technologies, ICALT 2006* 2006 (2006), 847–851. https://doi.org/10.1109/ICALT.2006.1652575

[5] James F. Allen. 1999. Mixed-initiative interaction. , 14–16 pages. https://doi.org/10.1109/5254.796083

[6] Erik Andersen, Sumit Gulwani, and Zoran Popović. 2013. A trace-based framework for analyzing and synthesizing educational progressions. *Conference on Human Factors in Computing Systems - Proceedings* (2013), 773–782. https://doi.org/10.1145/2470654.2470764

[7] Hiroaki Ayabe, Emmanuel Manalo, and Erica de Vries. 2022. Problem-appropriate diagram instruction for improving mathematical word problem solving. *Frontiers in Psychology* 13 (10 2022), 6113. https://doi.org/10.3389/fpsyg.2022.992625

[8] David Barrett, Felix Hill, Adam Santoro, Ari Morcos, and Timothy Lillicrap. 2018. Measuring abstract reasoning in neural networks. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 511–520. https://proceedings.mlr.press/v80/barrett18a.html

[9] Jonas Belouadi, Anne Lauscher, and Steffen Eger. 2024. AutomaTikZ: Text-Guided Synthesis of Scientific Vector Graphics with TikZ.

[10] Edward B Burger, David J Chard, Earlene J Hall, Paul A Kennedy, Steven J Leinwand, Freddie L Renfro, Dale G Seymour, and Bert K Wattis. 2007. *Holt geometry*. Holt, Rinehart and Winston.

[11] Patricia A. Carpenter, Marcel Adam Just, and Peter Shell. 1990. What one intelligence test measures: A theoretical account of the processing in the Raven progressive matrices test. *Psychological Review* 97, 3 (1990), 404–431. https://doi.org/10.1037/0033-295X.97.3.404

[12] Michelene T.H. Chi and Ruth Wylie. 2014. The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes. *Educational Psychologist* 49, 4 (10 2014), 219–243. https://doi.org/10.1080/00461520.2014.965823

[13] Pakey P.M. Chik and Mun Ling Lo. 2004. Simultaneity and the enacted object of learning. In *Classroom Discourse and the Space of Learning*. Vol. 9781410609. Routledge, 89–112. https://doi.org/10.4324/9781410609762

[14] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20, 1 (1960), 37–46.

[15] Louis Deslauriers, Logan S. McCarty, Kelly Miller, Kristina Callaghan, and Greg Kestin. 2019. Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom. *Proceedings of the National Academy of Sciences of the United States of America* 116, 39 (9 2019), 19251–19257. https://doi.org/10.1073/pnas.1821936116

[16] Andrea A. DiSessa. 2004. Metarepresentation: Native competence and targets for instruction. *Cognition and Instruction* 22, 3 (2004), 293–331. https://doi.org/10.1207/s1532690xci2203{_}2

[17] Susan E. Embretson. 1998. A Cognitive Design System Approach to Generating Valid Tests: Application to Abstract Reasoning. *Psychological Methods* 3, 3 (1998), 380–396. https://doi.org/10.1037/1082-989X.3.3.380

[18] K Anders Ericsson. 2006. The Influence of Experience and Deliberate Practice on the Development of Superior Expert Performance. In *The Cambridge handbook of expertise and expert performance*. Cambridge University Press, New York, NY, US, 683–703. https://doi.org/10.1017/CBO9780511816796.038

[19] Jean Rémy Falleri, Floréal Morandat, Xavier Blanc, Matias Martinez, and Martin Monperrus. 2014. Fine-grained and accurate source code differencing. In *ASE 2014 - Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*. Association for Computing Machinery, Inc, 313–323. https://doi.org/10.1145/2642937.2642982

[20] Mark J Gierl and Thomas M Haladyna. 2012. *Automatic item generation: Theory and practice*. Routledge.

[21] Mark J. Gierl and Hollis Lai. 2012. The Role of Item Models in Automatic Item Generation. *International Journal of Testing* 12, 3 (7 2012), 273–298. https://doi.org/10.1080/15305058.2011.635830

[22] Robert L. Goldstone, David H. Landy, and Ji Y. Son. 2010. The Education of Perception. *Topics in Cognitive Science* 2, 2 (4 2010), 265–284. https://doi.org/10.1111/J.1756-8765.2009.01055.X

[23] Sumit Gulwani. 2014. Example-based learning in computer-aided STEM education. *Commun. ACM* 57, 8 (2014), 70–80. https://doi.org/10.1145/2634273

[24] Sumit Gulwani, Vijay Anand Korthikanti, and Ashish Tiwari. 2011. Synthesizing geometry constructions. *ACM SIGPLAN Notices* 46, 6 (6 2011), 50–61. https://doi.org/10.1145/1993316.1993505

[25] Jacques Hadamard. 1997. *The Mathematician's Mind*. Princeton University Press. https://doi.org/10.1515/9780691212906/HTML

[26] D F Halpern, A Graesser, and M Hakel. 2007. Learning principles to guide pedagogy and the design of learning environments. *Association for Psychological Science* (2007).

[27] Heinz Holling, Jonas P. Bertling, and Nina Zeuch. 2009. Automatic item generation of probability word problems. *Studies in Educational Evaluation* 35, 2 (2009), 71–76. https://doi.org/10.1016/j.stueduc.2009.10.004 Assessment of Competencies.

[28] Lutz F. Hornke and Michael W. Habon. 1986. Rule-Based Item Bank Construction and Evaluation Within the Linear Logistic Framework. *Applied Psychological Measurement* 10, 4 (12 1986), 369–380. https://doi.org/10.1177/014662168601000405

[29] Rijul Jain, Wode Ni, and Joshua Sunshine. 2023. Generating Domain-Specific Programs for Diagram Authoring with Large Language Models. In *Companion Proceedings of the 2023 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity* (Cascais, Portugal) *(SPLASH'23)*. Association for Computing Machinery, New York, NY, USA, 70–71. https://doi.org/10.1145/3618305.3623612

[30] Alan Kay. 1987. Doing with images makes symbols. https://archive.org/details/AlanKeyD1987

[31] Philip J. Kellman, Christine M. Massey, and Ji Y. Son. 2010. Perceptual learning modules in mathematics: Enhancing students' pattern recognition, structure extraction, and fluency. *Topics in Cognitive Science* 2, 2 (4 2010), 285–305. https://doi.org/10.1111/j.1756-8765.2009.01053.x

[32] K Koedinger. 1990. Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science* 14, 4 (12 1990), 511–550. https://doi.org/10.1016/0364-0213(90)90008-K

[33] Kenneth R. Koedinger and Atsushi Terao. 2019. A Cognitive Task Analysis of Using Pictures To Support Pre-Algebraic Reasoning. *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society* (4 2019), 542–547. https://doi.org/10.4324/9781315782379-129

[34] Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, and Salam Al-Emari. 2020. A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education* 30 (2020), 121–204.

[35] Jill H. Larkin and Herbert A. Simon. 1987. Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science* 11, 1 (1 1987), 65–100. https://doi.org/10.1016/S0364-0213(87)80026-5

[36] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation strategies for HCI Toolkit research. *Conference on Human Factors in Computing Systems - Proceedings* 2018-April (4 2018). https://doi.org/10.1145/3173574.3173610

[37] Dor Ma'ayan, Wode Ni, Katherine Ye, Chinmay Kulkarni, and Joshua Sunshine. 2020. How Domain Experts Create Conceptual Diagrams and Implications for Tool Design. *Conference on Human Factors in Computing Systems - Proceedings* 20 (4 2020). https://doi.org/10.1145/3313831.3376253

[38] Ference Marton. 2006. Sameness and Difference in Transfer. *Journal of the Learning Sciences* 15, 4 (2006), 499–535. https://doi.org/10.1207/S15327809JLS1504{_}3

[39] Richard Mayer. 2020. *Multimedia Learning*. Cambridge University Press. https://doi.org/10.1017/9781316941355

[40] W. M. McCracken and W. C. Newstetter. 2001. Text to diagram to symbol: Representational transformations in problem-solving. *Proceedings - Frontiers in Education Conference* 2 (2001). https://doi.org/10.1109/FIE.2001.963721

[41] Mitchell J Nathan, Ana C Stephens, D K Masarik, Martha W Alibali, and Kenneth R Koedinger. 2002. Representational fluency in middle school: A classroom study. In *Proceedings of the twenty-fourth annual meeting of the North American chapter of the International Group for the Psychology of Mathematics Education*, Vol. 1. ERIC Clearinghouse for Science, Mathematics and Environmental Education˜…, 462–472.

[42] Fred G.W.C. Paas and Jeroen J.G. Van Merriënboer. 1994. Variability of Worked Examples and Transfer of Geometrical Problem-Solving Skills: A Cognitive-Load Approach. *Journal of Educational Psychology* 86, 1 (1994), 122–133. https://doi.org/10.1037/0022-0663.86.1.122

[43] Harold Pashler, Patrice M Bain, Brian A Bottge, Arthur Graesser, Kenneth Koedinger, Mark McDaniel, and Janet Metcalfe. 2007. *Organizing Instruction and Study to Improve Student Learning*. Technical Report. NCER, IES" U.S. Department of Education, Washington, DC.

[44] Rahul Patel, Shashwat Sanghavi, Dhruv Gupta, and Mehul S Raval. 2015. CheckIt - A low cost mobile OMR system. In *TENCON 2015 - 2015 IEEE Region 10 Conference*. 1–5. https://doi.org/10.1109/TENCON.2015.7372983

[45] Martina A Rau. 2013. *Conceptual learning with multiple graphical representations: Intelligent tutoring systems support for sense-making and fluency-building processes*. Ph. D. Dissertation. Carnegie Mellon University.

[46] Martina Angela Rau. 2017. A Framework for Educational Technologies that Support Representational Competencies. *IEEE Transactions on Learning Technologies* 10, 3 (7 2017), 290–305. https://doi.org/10.1109/TLT.2016.2623303

[47] Leena Razzaq, Jozsef Patvarczki, Shane F. Almeida, Manasi Vartak, Mingyu Feng, Neil T. Heffernan, and Kenneth R. Koedinger. 2009. The ASSISTment builder:

Supporting the life cycle of tutoring system content creation. *IEEE Transactions on Learning Technologies* 2, 2 (2009), 157–166. https://doi.org/10.1109/TLT.2009.23

[48] Jihyun Rho, Martina A. Rau, and Barry D. Van Veen. 2022. Preparing Future Learning with Novel Visuals by Supporting Representational Competencies. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 13355 LNCS. Springer Science and Business Media Deutschland GmbH, 66–77. https://doi.org/10.1007/978-3-031-11644-5{_}6

[49] Doug Rohrer and Kelli Taylor. 2007. The shuffling of mathematics problems improves learning. *Instructional Science* 35, 6 (11 2007), 481–498. https://doi.org/10.1007/s11251-007-9015-8

[50] Kenneth H Rosen. 1999. *Discrete mathematics & applications* (7th ed.). McGraw-Hill.

[51] Heidi L. Schnackenberg, Howard J. Sullivan, Lars F. Leader, and Elizabeth E.K. Jones. 1998. Learner preferences and achievement under differing amounts of learner practice. *Educational Technology Research and Development* 46, 2 (1998), 5–16. https://doi.org/10.1007/bf02299786

[52] Rohit Singh, Sumit Gulwani, and Sriram Rajamani. 2012. Automatically Generating Algebra Problems. *Proceedings of the AAAI Conference on Artificial Intelligence* 26, 1 (2012), 1620–1628. https://doi.org/10.1609/AAAI.V26I1.8341

[53] Kurt VanLehn. 2006. The Behavior of Tutoring Systems. *International Journal of Artificial Intelligence in Education* 16, 3 (1 2006), 227–265.

[54] Ke Wang and Zhendong Su. 2015. Automatic generation of Raven's progressive Matrices. In *IJCAI International Joint Conference on Artificial Intelligence*, Vol. 2015-Janua. 903–909.

[55] Daniel Weitekamp, Erik Harpstead, and Ken R. Koedinger. 2020. An Interaction Design for Machine Teaching to Develop AI Tutors. *Conference on Human Factors in Computing Systems - Proceedings* (4 2020). https://doi.org/10.1145/3313831.3376226

[56] Yuan Yang, Deepayan Sanyal, Joel Michelson, James Ainooson, and Maithilee Kunda. 2022. Automatic Item Generation of Figural Analogy Problems: A Review and Outlook. (1 2022). https://arxiv.org/abs/2201.08450v1http://arxiv.org/abs/2201.08450

[57] Katherine Ye, Wode Ni, Max Krieger, Dor Ma'ayan, Jenna Wise, Jonathan Aldrich, Joshua Sunshine, and Keenan Crane. 2020. Penrose: From Mathematical Notation to Beautiful Diagrams. *ACM Transactions on Graphics* 39, 4 (7 2020), 16. https://doi.org/10.1145/3386569.3392375