Educational Measurement: Issues and Practice
Spring 2019, Vol. 38, No. 1, pp. 48–53

A Cost–Benefit Analysis of Automatic Item Generation

Audra E. Kosh, *MetaMetrics Inc.*, Mary Ann Simpson, *NWEA*, Lisa Bickel, Mark Kellogg, and Ellie Sanford-Moore, *MetaMetrics Inc.*

Automatic item generation (AIG)—a means of leveraging technology to create large quantities of items—requires a minimum number of items to offset the sizable upfront investment (i.e., model development and technology deployment) in order to achieve cost savings. In this cost–benefit analysis, we estimated the cost of each step of AIG and manual item writing and applied cost—benefit formulas to calculate the number of items that would have to be produced before the upfront costs of AIG outweigh manual item writing costs in the context of K-12 mathematics items. Results indicated that AIG is more cost-effective than manual item writing when developing, at a minimum, 173 to 247 items within one fine-grained content area (e.g., fourth- through seventh-grade area of figures). The article concludes with a discussion of implications for test developers and the nonmonetary tradeoffs involved in AIG.

Keywords: automatic item generation, cost–benefit analysis, test development

Emerging learning technologies, which often use computer-based interim or summative assessments, provide a rich opportunity for students to receive immediate feedback. However, the use of such technology often requires a large item bank for on-demand administration. One method that can potentially alleviate item development burden is automatic item generation (AIG). In strong-theory AIG, features of items theorized to affect item difficulty are systematically varied, ideally resulting in items with a predicted difficulty.

AIG has several benefits. First, because automatically generated items are created according to a theory of task difficulty, AIG can provide a source of validity evidence based on test development procedures and test content (Bejar, 2013). Second, AIG may provide the ability to estimate psychometric characteristics of items without pretesting, known as precalibration (Gierl & Lai, 2016). In precalibration, features of items used in generation become variables in statistical models to predict item difficulty and/or discrimination (Gierl & Lai, 2016; Sinharay & Johnson, 2013). Precalibration is beneficial because it can minimize field testing requirements, either by eliminating pretesting of isomorphic items altogether or reducing field test sample sizes of examinees (Embretson & Daniel, 2008; Gierl, Zhou, & Alves, 2008; Haladyna & Rodriguez, 2013). Third, due to the use of cognitive models and precalibration, AIG may aid in yielding items across all ranges of difficulty by specifying the desired item difficulty upfront rather than classifying cognitive complexity after an item is written, as is commonly done in traditional item writing (Daniel & Embretson, 2010). The latter can result in too few or too many items at a particular difficulty level (Drasgow, Luecht, & Bennett, 2006). With AIG, however, factors impacting difficulty are specified during item generation, thereby allowing test developers to systematically manipulate item features to arrive at items with particular difficulty targets.

In addition to the aforementioned benefits, efficient and rapid production of large quantities of items may result in cost savings during test development (Gierl & Lai, 2016; Sinharay & Johnson, 2008; Wainer, 2002). However, the upfront investment to implement item generation may outweigh the benefits of eventual automation. To our knowledge, no published analyses of the time and cost tradeoffs involved in each type of item development process exist. In this article, we compare the costs of AIG with the costs of traditional (i.e., manual, handwritten) item development methods in order to understand if and when AIG may indeed have a return on the upfront investment required to generate items. First, to understand the extensive processes required to implement AIG, we give an overview of the steps involved in strong-theory AIG.

Gierl and Lai (2016) outlined a three-step process of strong-theory AIG: (1) develop a cognitive model (i.e., theory of how examinees process items), (2) develop item models (i.e., templates that the computer uses to generate items), and (3) deploy computer technology to generate items. In step 1, researchers, psychometricians, and subject matter experts collaborate to create a theory of which features of a task may affect the difficulty of the task. At this stage, researchers may consult existing literature for empirical research regarding how students process tasks in the domain of interest. For example, articles presenting learning trajectories may provide helpful information about how students progress in their understanding of a topic, which can then be translated into features of a task that are easier or more difficult for students (e.g., Clements, Wilson, & Sarama, 2004). Researchers may also conduct analyses on existing, previously administered items with known empirical difficulties. For example, researchers can create a coding scheme to apply to existing items, code the items for characteristics theorized to affect difficulty (e.g., it may be easier for students

Audra E. Kosh, MetaMetrics Inc.; audrakosh@gmail.com.

to model fractions with denominators 2 and 4 than fractions with denominators 3 and 5), and conduct an analysis of which characteristics most strongly predict the items' empirical difficulties.

In step 2, the features theorized to affect item difficulty, as identified in step 1, are used to create item models. An item model is "a template which highlights how the features in an assessment task can be manipulated to produce new items" (Gierl, Lai, Hogan, & Matovinovic, 2015, p. 2). In other words, the item model encompasses specifications for writing an item, and those specifications contain enough detail to warrant creation through algorithmic, computerized means (Bejar, 2010).

In step 3, computer technology is used to generate items based on the item models developed in step 2. This stage is where much of the software engineering labor of item generation takes place. Although to our knowledge there is no open-source technology for item generation, test developers have created their own in-house programs for item generation such as Item Generator (more commonly known as IGOR; Gierl, Zhou, & Alves, 2008), the Mathematics Test Creation Assistant (Singley & Bennett, 2002), the Enhanced Automatic Question Creator (Gutl, Lankmayr, Weinhofer, & Hofler, 2011), and the Mathematics Item Generator (Kellogg et al., 2015). Once a testing program has its own item generation tool, production of generated items should theoretically take minimal time. However, creating item generation tools requires an upfront investment, maintenance of the tool, ongoing enhancements to support new content or item types, and ongoing quality checking of the generated items.

In addition to the three aforementioned steps outlined by Gierl and Lai (2016), we add a fourth step: evaluate both the cognitive model used to create item models and the psychometric performance of item models. This includes conducting a field test with several instances of generated items for each item model, analyzing the psychometric properties of both the generated items and item models (e.g., variance of item difficulty within an item model), and evaluating how well the features specified in the cognitive model predict item difficulty. Evaluation of the cognitive model and item models provides evidence to support further item generation for operational use and possibly item precalibration.

These four steps illustrate that AIG is anything but automatic. Extensive work is required to develop cognitive models and item models; automation only happens in the final step and only after the infrastructure for AIG technology has been developed. Bejar (2002) points out that "to make item generation possible, it is necessary to understand the construct and the goal of the examination in far more detail that would otherwise be the case" (p. 202). For this reason—namely, that AIG requires an in-depth understanding of test development and the construct under consideration and an extensive time investment to initially implement—it is not immediately apparent than AIG is more cost-effective than traditional item writing.

The in-depth work required to implement AIG raises the question of if—or under which circumstances—AIG is truly more efficient than traditional, manual item writing whereby trained subject matter experts write individual items. The attempt to answer this question lies in a cost–benefit analysis: a comparison of the costs associated with each item development method—AIG and traditional item writing—to determine the break-even point when the eventual rapid production of automation outweighs the high start-up costs related to creating the automated process (i.e., the point at which AIG is more cost-effective than traditional item writing).

The exact methodology used to conduct a cost–benefit analysis varies based on the context and purpose of the costs under consideration. Generally, the method includes estimating the costs of automating the production of a widget (i.e., C_A) and the costs of manually creating one of those widgets (i.e., C_M). To determine when the automated method is more cost-effective, one can set the automated costs less than the product of the manual per-widget creation costs and the number of widgets created, namely

$$C_A < C_M * N_M, \tag{1}$$

where N_M is the number of widgets produced manually. Solving Equation 1 for N_M then estimates how many manually produced widgets one needs for the automated method to outweigh the cost of manual development, namely,

$$N_M > \frac{C_A}{C_M}. \tag{2}$$

Thus, when the number of widgets, N_M, exceeds the ratio of automated costs to manual per-widget costs, it is more cost-effective to use the automated method as compared to manual development. For example, if it costs $25 to manually create one widget and $500 to automate widget production, it is more cost-effective to use automation if production needs call for more than 20 widgets. Although these formulas appear simplistic, the task of accurately quantifying C_A and C_M can be quite complex, and the actual formulas used in practice may include additional variables depending on the context.

As indicated in a report by Hanover Research (2012), "cost–benefit analysis is either not currently prevalent in K-12 education or these analyses are not made available to the public" (p. 2). The authors of that report suggest that the former is more likely the case. With finite resources devoted to education, it is alarming that such few, if any, stakeholders evaluate the relative costs and benefits of spending on particular resources. The purpose of this study is to estimate item development costs associated with strong-theory AIG and traditional item writing in order to determine the threshold at which AIG costs are less than traditional item writing costs. The research question is: how many items must be produced before AIG is more cost-effective than traditional item writing?

Cost–Benefit Analysis: AIG and Traditional Item Writing

We explored the research question in the context of K-12 multiple-choice mathematics items from the work of two item development teams—one team following traditional item writing and one team using AIG—within the same testing program. We conducted analyses within the sole domain of mathematics because mathematics items cover a wide range of content and graphics (e.g., base 10 pieces to model decimals, coordinate planes, three-dimensional figures), each with its own start-up costs related to developing content (e.g., distractor rationales based on common student misconceptions, graphics programming). For traditional item writing, the team developed a batch of 240 individually written mathematics items. For AIG, subject matter experts, psychometricians, researchers, and software engineers collaborated to produce an item generator that incorporated approximately 40 item models within the fourth- through seventh-grade mathematics measurement strand that varied features of the items (e.g., squares vs. circles and whole numbers vs. decimals) relating to calculating area. Both methods incorporated graphics, respectively produced by a graphic artist or computer-generated, and both types of item development processes used distractors based on common student errors. For example, if an item required calculating area, a distractor might include a calculation of perimeter, indicating that the student confused the concepts of area and perimeter. The estimates we present assume that all items—both traditionally written items and generated items—are held to the same quality standards.

We began by developing a detailed list of all tasks in the item development process for both traditional item writing and AIG. For traditional item writing, tasks included all aspects of item development such as drafting items, performing a variety of reviews (e.g., reviews for content appropriateness by subject matter expert reviews for sensitivity to various student populations, proofreading, etc.), producing graphics as needed (e.g., coordinate planes and number lines), and cold solving whereby an individual uninvolved in the previous item development steps solves the item. Item development tasks also included tasks specific to field testing, such as obtaining approval by an institutional review board, recruiting field test participants, grouping items onto forms, administering tests, providing incentives to field test participants, and analyzing field test item data to determine which items to keep, revise, or remove. Tasks in AIG included all aspects of the four steps described above related to developing a cognitive model, developing item models (including rigorous reviews similar to the reviews performed on manually written items), creating and deploying technology to generate items, and evaluating the cognitive model and item models used in item generation by field testing items generated from the item models and analyzing their psychometric properties.

The tasks involved in AIG and traditional item writing share many aspects (e.g., a similar review process of items and item models and the need to field test both traditionally written items and a sample of automatically generated items), though the tasks differ in two main areas. First, writing an item model for AIG requires more time and skill than manually writing a single item. When writing item models, the author must define acceptable values for substitution in the item model and critically think through all possible instances of item stems and distractors generated from that model to ensure that all generated items function similarly with respect to the mathematics content of the item, the reasonableness of any real-world situations presented in the item, the ability to generate appropriate graphics for the item, plural/singular subject-verb agreement used in the stem, and plausibility of distractors produced for the item. Due to these complexities, writing and reviewing item models takes longer than writing and reviewing individual items. Second, another difference of tasks in AIG as compared to traditional item writing is the creation and deployment of computer technology for AIG, which in many cases requires substantial effort by software engineers and subsequent quality checking.

Following creation of detailed task lists, we consulted with team members from both the AIG team and the traditional item writing team regarding how much time they spent on each task. Both teams were experienced in their own type of item development methods and had already completed several similar projects prior to reporting estimates, hence our estimates assume little to no learning curve. As aforementioned, the traditional item writing estimates are based on developing a batch of 240 items, and the traditional item writing team had completed numerous of these 240-item batches prior to reporting to their estimates. As for the automatically generated items, the AIG team had previously developed several different generators based on different mathematics content areas, and the estimates reported here are based on the time to create an additional generator for a new mathematics content area. For example, the team had existing generators for topics such as operations with fractions and operations with whole numbers, and the new generator required content related to finding the area of figures. Additionally, we specifically chose mathematics content that was not as straightforward as generating mathematics fluency items; the content we focused on (i.e., finding area in fourth- through seventh-grade mathematics) included various types of figures, word problems, and formulas.

Team members individually reported their estimates of time spent on each task without seeing estimates of the opposite team's reported time; that is, software engineers did not know the amount of time the traditional item writing team reported, and the traditional item writing team did not know the amount of time software engineers reported. Some employees (e.g., project managers and psychometricians) worked on both teams; in those cases, the employees reported time estimates for their work on each task and team separately. By using the same employees on both teams where relevant, the study ensures that any variance in time estimates on related tasks for each team is due to true variation in the complexity of the task for the respective item development methodology as opposed to variation in employee productivity rates. For example, since the same psychometrician analyzed field-test data from manually written items and automatically generated items, time estimates for conducting analyses reflect variation in the complexity of data from each item development methodology (i.e., analyses of AIG items typically are more complex because of the hierarchical nature of multiple unique items developed from one AIG item model).

Both teams were motivated to report time as accurately as possible because they knew they would eventually be held to the timelines they had estimated for future project work. For example, although a software engineer might be tempted to underestimate time due to a desire to make his/her method seem more efficient, the software engineer would then be required to adhere to the schedule he/she had specified on the next round of item generation work. Hence, employees wanted to avoid reporting an unrealistic, underestimated amount of time.

Moreover, both teams were aware that the results of the cost–benefit analysis would not affect their employment by putting them at risk for being laid off. All employees worked on multiple projects beyond item development; therefore, even if the testing program chose to use one method of item development over another, there were many other projects requiring employees' skills and time. Thus, employees from both teams were motivated to report time estimates accurately, though reports were made retroactively rather than recorded daily during the item development period.

After employees provided their individual, blinded estimates of time to complete each step of either the AIG or traditional item development process, a representative from human resources used employees' hourly pay rates (or equivalent hourly rates converted from annual salaries) to complete a spreadsheet template that translated employees' time to costs and included an adjustment for the cost of employee benefits. The human resource representative then edited the spreadsheet to retain the total cost for each task in the item development process for AIG and traditional item development, removed individual employee pay rates, and returned the spreadsheet to the researchers for analysis. By using actual employee pay rates, this study accounts for the equivalent cost of time for employees with different skills. For example, senior software engineers designing complex automated procedures may be more expensive to employ than subject matter experts manually writing items.

The first step in the cost benefit analysis of AIG and traditional item writing is to estimate the cost of producing one operationally acceptable manually written item. Due to the fine level of granularity and variation in the time to produce one item, our estimates were based off of the time to create a batch of 240 items, as previously described. Thus, assuming a field test survival rate of r (e.g., if 80% of items perform well psychometrically in field testing, then $r = .8$), the cost to produce one manually written item, C_{M}, is

$$C_{\mathrm{M}} = \frac{C_{\mathrm{W}} + C_{\mathrm{F}}}{240r}, \tag{3}$$

where C_{W} is the cost to write and review a batch of 240 items, and C_{F} is the cost to field test a batch of 240 items.

Similarly, to estimate the cost of using AIG to produce 40 item models within one content area, C_{A}, we use the equation

$$C_{\mathrm{A}} = C_{\mathrm{S_1}} + C_{\mathrm{S_2}} + C_{\mathrm{S_{3a}}} + C_{\mathrm{S_{3b}}} + C_{\mathrm{S_4}} + M, \tag{4}$$

where $C_{\mathrm{S_1}}$ is the cost of step 1 in item generation (i.e., developing a cognitive model); $C_{\mathrm{S_2}}$ is the cost of step 2 in item generation (i.e., creating item models); $C_{\mathrm{S_{3a}}}$ and $C_{\mathrm{S_{3b}}}$ are the costs of step 3 of item generation (i.e., producing computer technology to generate items), separated into initial costs for programming infrastructure ($C_{\mathrm{S_{3a}}}$) and subsequent content-specific programming costs ($C_{\mathrm{S_{3b}}}$; e.g., specialized graphics for particular item models or adding new content-specific distractors); $C_{\mathrm{S_4}}$ is the cost of step 4 in item generation (i.e., evaluating the cognitive model and item models), and M is the cost of maintaining the generator for one year.

Substituting Equations 3 and 4 into Equation 2, we find that the point at which AIG is more cost-effective than traditional item writing is

$$N_{\mathrm{M}} > \frac{C_{\mathrm{S_1}} + C_{\mathrm{S_2}} + C_{\mathrm{S_{3a}}} + C_{\mathrm{S_{3b}}} + C_{\mathrm{S_4}} + M}{\frac{C_{\mathrm{W}} + C_{\mathrm{F}}}{240r}}. \tag{5}$$

The values of $C_{\mathrm{S_1}}, C_{\mathrm{S_2}}, C_{\mathrm{S_{3a}}}, C_{\mathrm{S_{3b}}}, C_{\mathrm{S_4}}, M, C_{\mathrm{W}}$, and C_{F} were all derived from the aforementioned estimates of employees time as converted to dollar amounts using actual salaries. The field test survival rate, r, is a parameter we allow to vary, and the result, N, is the solved-for number of items at which the total costs of AIG outweigh the cost of traditional item writing.

To maintain confidentiality of the testing program and employees' pay rates, we report costs in this article relative to the cost of implementing traditional item development for a single item with various different field testing survival rates. Namely, let $C_{\mathrm{M}r\,=\,.8}$ be the cost of developing and field testing one item through the traditional method where we assume a field test survival rate of 80% (i.e., $r = .8$). Table 1 shows two rows for total costs of AIG for various values of r. The first line, "Total, with one-time technology start-up costs," represents the total costs of initial implementation of AIG on a batch of 40 item models, expressed as the break-even point N_{M}. Also, because programming costs vary by creating initial AIG infrastructure and content-specific programming for future AIG batches, we show a second total line, "Total, without one-time technology start-up costs," which includes content-specific programming costs for a subsequent batch of 40 item models but excludes programming costs related to developing initial infrastructure. In other words, it is more expensive to implement AIG for an initial batch of item models because technology infrastructure does not exist yet. Subsequent batches of content areas merely require tweaking or amending programs to accommodate new content or item types. Finally, Table 1 also includes the breakdown of relative costs of each step in AIG. These values sum to produce the aforementioned totals according to Equation 4.

Considering the most simplistic case to aid interpretation, assume all items survive field testing (i.e., $r = 1$) and assume, for illustrative purposes only, it costs \$1 to produce a single manually written item. With these assumptions, our results show it costs \$247.49 to automatically generate items based on 40 item models within a single content area covering three to five related mathematics skills modeled together within one cognitive model. That is, AIG is about 247 times as expensive as producing one manually written item. The total cost of \$247.49 breaks down as \$14.22 for developing a cognitive model, \$36.15 for developing item models, \$139.66 for creating and deploying technology to generate items (i.e., \$121.55 for technology infrastructure for the first batch of AIG implementation and \$18.11 for content-specific programming for subsequent AIG batches of 40 item models), \$27.59 for evaluating the cognitive model and item models, and \$29.87 for one year's worth of generator maintenance.

We used these dollar amounts of AIG only in the most simplistic, illustrative case that assumes one manually written item costs merely \$1 to produce. For a more general interpretation, rounding the aforementioned dollar amounts to the nearest whole number yields the approximate number of manually written items equivalent in costs to the cost of each step in the item-generation process. For example, based on the total presented in Table 1 for all items surviving field testing (i.e., $r = 1$), it costs the equivalent of 247 manually written items to implement AIG—including technology start-up costs—or the equivalent costs of 126 manually written items to implement AIG for a new batch of 40 item models within one cognitive model if technology start-up costs have already been incurred. This means that, during a first-time implementation of AIG, AIG is more cost-effective than traditional item writing if a testing program requires 247 or more items within one content area. For subsequent rounds of AIG implementation (i.e., programming start-up costs have already been incurred), AIG is more cost-effective than traditional item writing if a testing program requires 126 or more items within one content area. As a reminder, content area in our context refers to a set of approximately three to five related mathematics skills that can be modeled with a single cognitive model, such as calculating the area of quadrilaterals, triangles, and circles in our example. We do not refer to a content area as including all of mathematics, all of history, all of chemistry, and so on.

Table 1 also includes results for different field-testing survival rates. For example, if assuming that 90% of manually written items perform well psychometrically during field testing and may be retained as operational items, the cost per item of manual item writing decreases because fewer items are discarded or revised. Thus, as the cost per item of manual writing decreases, the start-up costs to implement AIG become more expensive relative to the cost of traditional item writing. Although out AIG cost estimates did account for field testing of generated items to ensure that all 40 item models are appropriate for operational testing, we purposefully did not account for survival rates of automatically generated items or item models because (1) theoretically, the cognitive modeling stage of AIG already attempts to control for the psychometric properties of the item, particularly item difficulty, which should result in well-performing generated items; and (2) it is assumed that any items that did not perform well could be immediately replaced by newly generated items. In other words, unlike traditional item writing in which new items must be written to replace poorly performing items, AIG essentially provides an infinite supply of items. Even if all 40 item models did not survive field testing, the AIG process still results in a substantial quantity of items, albeit with less variety than the original 40 item models would have produced.

Discussion

Based on the totals in Table 1, AIG is more cost-effective than traditional item writing during initial implementation if a total of 173 to 247 or more items are needed within the same content area, where that content area includes approximately three to five related skills that can be modeled with a single cognitive model. After an initial implementation of AIG (i.e., programming infrastructure already exists), AIG remains cost-effective if producing around 88 to 126 or more items within a specific content area, with variation in the minimum number of items needed for costs to break even due to varying assumptions of field test survival rates of manually written items.

When interpreting these results, test developers should consider how many items with related content their testing programs require. To do this, test developers should first specify which skills their test assesses that would group well together under a single AIG cognitive model. In this cost–benefit analysis, our cognitive model included three to five specific fourth- through seventh-grade skills related to finding area. Other examples of mathematics skills with potential for modeling in a single cognitive model include seventh-grade operations with integers or high-school content about measures of central tendency. Understanding which skills group well together for cognitive modeling is critical in order to determine the scope of content produced in one batch of AIG implementation, since different content requires different cognitive models and item models. After test developers determine which skills form a group for cognitive modeling, the test developer should count how many items the testing program requires within that group of skills in a particular time frame (e.g., 1 year). Then, the test developer can compare the number of items required to the results of this analyses whereby a minimum of 173 to 247 items are necessary for cost-effective AIG. For example, if a K-12 testing program only used five items per year aligned to performing operations with integers, it clearly would take many years before the upfront costs of AIG outweighed manual item writing costs for the five annual items. On the other hand, educational technology providers or robust testing programs may administer frequent interim assessments, use computer adaptive tests, and/or desire that students receive different items from each other for test security purposes. In these cases, it may be beneficial for test developers to implement AIG after careful consideration of how many items with similar content the testing program requires.

As a limitation of this study, we only considered costs related to production and field testing of items and did not incorporate potential nonmonetary tradeoffs in either traditional item writing or AIG. Examples of nonmonetary tradeoffs, some of which favor AIG and some of which favor manual item writing, include:

Table 1. Number of Manually Written Items Equivalent in Costs to Costs of Various Steps of AIG

		Field Test Survival Rate of Manually Written Items (r)			
Step of AIG	Calculated Value	.7	.8	.9	1
Step 1: Develop cognitive model for specific content area	C_{S1}/C_{Mr}	9.95	11.37	12.80	14.22
Step 2: Develop 40 item models	C_{S2}/C_{Mr}	25.31	28.92	32.54	36.15
Step 3: Develop and deploy technology to generate items	$(C_{S3a} + C_{S3b})/C_{Mr}$	97.76	111.73	125.69	139.66
3a. One-time technology start-up costs (e.g., programming infrastructure)	C_{S3a}/C_{Mr}	85.09	97.24	109.40	121.55
3b. Content-specific technology programming for desired item models	C_{S3b}/C_{Mr}	12.68	14.49	16.30	18.11
Step 4: Evaluate cognitive model and item models	C_{S4}/C_{Mr}	19.31	22.07	24.83	27.59
Maintain generator for 1 year	M	20.91	23.90	26.89	29.87
Total (with one-time technology start-up costs)	N_M	173.24	197.99	222.74	247.49
Total (without one-time technology start-up costs)	$N_M - C_{S3a}/C_{Mr}$	88.16	100.75	113.34	125.94

- Item exposure rates: On one hand, AIG controls item exposure by increasing the number of items in the item bank, thereby reducing threats to test security and likewise threats to score validity (Gierl & Haladyna, 2013; Gierl & Lai, 2016; Wainer, 2002). On the other hand, if a single AIG item model is unintentionally released, then all isomorphic items generated from that model are essentially exposed as well, meaning that AIG, in this case, can raise additional test security concerns.
- Heterogeneity of items: Because AIG can generate hundreds or thousands of items from a single item model, items may lack diversity because too many items stem from the same template. For example, an item model consisting of a mathematics word problem would generate a plethora of word problems with different numbers but the same context, whereas traditional item writing would yield diverse word problem contexts for every unique item.
- Scheduling demands: After developing AIG infrastructure, test developers can likely meet quicker turnaround times for producing items than is feasible when manually developing items. Thus, when a need for new items arises—assuming previous completion of AIG tasks—users of AIG can simply deploy existing item model content and technology to rapidly create items rather than beginning the test development process from scratch, which shortens timelines for test development.
- Validity evidence based on test development procedures: Cizek (2015) presented four sources of evidence for validation of intended test score inferences, including evidence based on test development procedures. Because strong-theory AIG relies on a cognitive model that specifies which features of a task affect difficulty, the use of strong-theory AIG may provide validity evidence based on test development procedures. When the percent of variance in item difficulty explained by construct-relevant variables is high, a validity argument is supported because there is less variance due to construct-irrelevant factors (Bejar, 2013).

As additional limitations, we estimated test development costs for both AIG and traditional item writing for a particular testing program within a single domain (i.e., K-12 mathematics), estimates of employee time were reported retrospectively, and the content area and item type we chose (i.e., multiple-choice mathematics area items)—although more complex than simple numerical fluency items—were relatively basic as compared to the range of possible content areas and item types available. Furthermore, our analyses focused strictly on generating individual items; we did not consider how items might fit together to create forms, and we do not claim that test forms should exclusively contain generated items. Most importantly, we note that results would likely vary based on the nuances of each unique testing program, and test developers should consider their specific needs, purpose, and resources in order to make an informed decision about the most appropriate item development methodology. We encourage future research regarding how AIG costs may vary within other content areas, item types, and grade levels.

References

Bejar, I. I. (2002). Adaptive generative testing: From conception to implementation. In S. Irvine & P. Kyllonen (Eds.), *Item generation for test development* (pp. 199–218). Mahwah, NJ: Lawrence Erlbaum.

Bejar, I. I. (2010). Recent development and prospects in item generation. In S. Embretson (Ed.), *Measuring psychological constructs* (pp. 201–226). Washington, DC: American Psychological Association.

Bejar, I. I. (2013). Item generation: Implications for a validity argument. In M. J. Gierl & T. M. Haladyna (Eds.), *Automatic item generation: Theory and practice* (pp. 40–56). New York, NY: Routledge.

Cizek, G. J. (2015). Validating test score meaning and defending test score use: Different aims, different methods. *Assessment in Education: Principles, Policy and Practice*, *23*(2), 212–225. https://doi.org/10.1080/0969594X.2015.1063479

Clements, D. H., Wilson, D. C., & Sarama, J. (2004). Young children's composition of geometric figures: A learning trajectory. *Mathematical Thinking and Learning*, *6*(2), 163–184. https://doi.org/10.1207/s15327833mtl0602_5

Daniel, R. C., & Embretson, S. E. (2010). Designing cognitive complexity in mathematical problem-solving items. *Applied Psychological Measurement*, *34*, 348–364. https://doi.org/10.1177/0146621609349801

Drasgow, F., Luecht, R. M., & Bennett, R. E. (2006). Technology and testing. In R. L. Brennan (Ed.), *Educational measurement* (4th ed., pp. 471–516). Westport, CT: Praeger.

Embretson, S. E., & Daniel, R. C. (2008). Understanding and quantifying cognitive complexity level in mathematical problem solving items. *Psychology Science*, *50*, 328–344.

Gierl, M. J., & Haladyna, T. M. (2013). Automatic item generation: An introduction. In M. J. Gierl & T. M. Haladyna (Eds.), *Automatic item generation: Theory and practice* (pp. 3–12). New York, NY: Routledge.

Gierl, M. J., & Lai, H. (2016). Automatic item generation. In S. Lane, M. R. Raymond, & T. M. Haladyna (Eds.), *Handbook of test development* (2nd ed., pp. 410–429). New York, NY: Routledge.

Gierl, M. J., Lai, H., Hogan, J. B., & Matovinovic, D. (2015). A method for generating educational test items that are aligned to the Common Core State Standards. *Journal for Applied Testing Technology 16*(1), 1–18.

Gierl, M. J., Zhou, J., & Alves, C. (2008). Developing a taxonomy of item model types to promote assessment engineering. *Journal of Technology, Learning and Assessment*, *7*(2), 1–51.

Gutl, C., Lankmayr, K., Weinhofer, J., & Hofler, M. (2011). Enhanced automatic question creator – EAQC: Concept, development, and evaluation of an automatic test item creation tool to foster modern e-education. *Electronic Journal of e-Learning*, *9*(1), 23–38.

Haladyna, T., & Rodriguez, M. (2013). Developing the test item. In T. Haladyna & M. Rodriguez (Eds.), *Developing and validating test items* (pp. 17–27). New York, NY: Routledge.

Hanover Research. (2012). *Cost–benefit analysis in K-12 education*. Washington, DC: Hanover Research. Retrieved from https://www.gssaweb.org/wp-content/uploads/2015/04/Cost-Benefit-Analysis-in-K-12-Education-1.pdf

Kellogg, M., Rauch, S., Leathers, R., Simpson, M. A., Lines, D., Bickel, L., & Elmore, J. (2015, April). *Construction of a dynamic item generator for K-12 mathematics*. Paper presented at the 2015 National Council on Measurement in Education Conference, Chicago, IL.

Singley, M. K., & Bennett, R. E. (2002). Item generation and beyond: Applications to schema theory to mathematics assessment. In S. H. Irvine & P. C. Kyllonen (Eds.), *Item generation for test development* (pp. 361–384). Mahwah, NJ: Lawrence Erlbaum.

Sinharay, S., & Johnson, M. S. (2008). Use of item models in a large-scale admissions test: A case study. *International Journal of Testing*, *8*(3), 209–236. https://doi.org/10.1080/15305050802262019

Sinharay, S., & Johnson, M. (2013). Statistical modeling of automatically generated items. In M. J. Gierl & T. M. Haladyna (Eds.), *Automatic item generation: Theory and practice* (pp. 183–195). New York, NY: Routledge.

Wainer, H. (2002). On the automatic generation of test items: Some whens, whys, and hows. In S. Irvine & P. Kyllonen (Eds.), *Item generation for test development* (pp. 287–305). Mahwah, NJ: Lawrence Erlbaum.