Effect of discrete and continuous parameter variation on difficulty in automatic item generation

Binglin Chen$^{1[0000-0001-9033-1281]}$, Craig Zilles$^{1[0000-0003-4601-4398]}$, Matthew West$^{2[0000-0002-7605-0050]}$, and Timothy Bretl$^{3[0000-0001-7883-7300]}$

[1] Department of Computer Science, University of Illinois at Urbana-Champaign
[2] Department of Mechanical Engineering, University of Illinois at Urbana-Champaign
[3] Department of Aerospace Engineering, University of Illinois at Urbana-Champaign
{chen386, zilles, mwest, tbretl}@illinois.edu

Abstract. Automatic item generation enables a diverse array of questions to be generated through the use of question templates and randomly-selected parameters. Such automatic item generators are most useful if the generated item instances are of either equivalent or predictable difficulty. In this study, we analyzed student performance on over 300 item generators from four university-level STEM classes collected over a period of two years. In most cases, we find that the choice of parameters fails to significantly affect the problem difficulty.

In our analysis, we found it useful to distinguish parameters that were drawn from a small number (< 10) of values from those that are drawn from a large—often continuous—range of values. We observed that values from smaller ranges were more likely to significantly impact difficulty, as sometimes they represented different configurations of the problem (e.g., upward force vs. downward force). Through manual review of the problems with significant difficulty variance, we found it was, in general, easy to understand the source of the variance once the data were presented. These results suggest that the use of automatic item generation by college faculty is warranted, because most problems don't exhibit significant difficulty variation, and the few that do can be detected through automatic means and addressed by the faculty member.

Keywords: Automatic item generation · Item models.

1 Introduction

In classroom settings, computerized assessment offers many advantages compared to paper-based assessment, in both formative and summative assessment. In both contexts, it enables automatically grading a wide range of problem types [26], which reduces grading workloads, and provides students with immediate feedback [3], which has been shown to improve learning [22]. In formative assessment, computerized assessment enables mastery-oriented pedagogies [6, 18] where students can repeat problem types until mastery is demonstrated, and it permits assigning different problems to each student to discourage plagiarism [23]. In summative assessment, computer-based testing enables more authentic item types (e.g., programming exams on computers where compilers and debuggers are available), reduces the overhead of exam administration, especially for large classes [27] and permits the use of adaptive testing [19].

Many of these applications of computer-based assessment benefit from large pools of problems. One proposed method of generating a large collection of problems is automatic item generation (AIG) [13, 17], where item instances (items) are generated by instantiating parameterizable problem templates with specific values (see Section 2.1). AIG has been used across a broad range of disciplines [15] and has proven useful for generating large pools of items.

AIG is most useful, however, if the item instances produced by an item generator are of similar difficulty, or at minimum of a predictable difficulty. Previous research has shown that generally there is variation in psychometric properties of the item instances (items) produced by an item generator (item model), but that variation tends to be smaller than the variation between generators [1, 2, 7, 11, 12, 24, 10, 16, 20, 21]. There is a growing consensus that calibration should be done at the level of the item generator, using a multi-level strategy where item instances are nested within item generator [7, 9, 11, 14, 25].

This previous work, however, has largely focused on questions designed by psychometricians for standardized testing. For wide spread adoption of AIG across educational contexts, it is important to understand the extent to which disciplinary experts that are not experts in test construction can construct AIGs.

In this paper, we study AIG item difficulty variance in an ecologically-valid higher-education setting where university faculty members have written AIGs for use in both computerized homework and exams in large-enrollment STEM courses. We analyze student exam performance across a two-year time period on a collection of over 300 AIGs. We believe this also represents the largest reported study of AIGs to date. We describe our experimental data in detail in Section 2.

Our analysis focuses on how the choice of parameters for a given problem affects its difficulty. Methodologically, we found it necessary to break our analysis of parameters into two separate groups. In the first group, the number of unique values of the parameter was small (< 10), which meant that we had sufficiently many samples for each parameter to use the hybrid Fisher's exact test to check for significant variation in difficulty between the parameters. The analysis of these *discrete* parameters is presented in Section 3. In the other group, the number of unique parameters was large enough that only a few samples were available for many of the parameter values, and for some parameters every student had a unique value. For this *continuous* parameter group, we used the Kolmogorov-Smirnov test, as described in Section 4.

In this analysis, we find that the vast majority of the AIGs studied can be characterized as *uniform generators*, in that their generated instances are of equal difficulty (or at least any differences in difficulty are too small to be detected). In our university-level STEM context, the few *non-uniform generators*—where some instances are significantly harder than others—that we found, were more frequently due to discrete parameters that represented different "configurations" of the problem. University STEM students may be largely insensitive to the specific numbers in problems because they are allowed to use calculators. Our paper makes the following contributions:

1. We demonstrate the utility of using the Kolmogorov-Smirnov test to analyze the difficulty variance in AIG with many unique parameter values,
2. We provide evidence that faculty that are disciplinary content experts, but not experts in test construction or psychometrics, are largely capable of writing AIGs without significant difficulty variation, and
3. We find that once an AIG has been identified as having significant difficulty variance, it is generally easy to understand the source of the variance. This suggests that an automated analysis tool could be used by disciplinary faculty to correct item generators or divide an item generator into a collection of item generators that each have a stable difficulty.

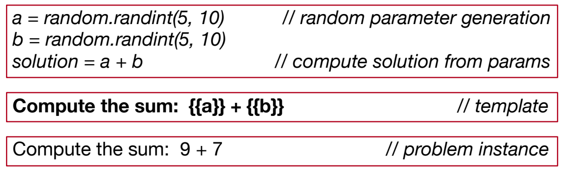
2 Experimental data

The data was collected at a large R1 university in the US from Fall 2016 to Summer 2018. The four courses studied are drawn from introductory sequences in Electrical & Computer Engineering (electronics) and Mechanical Engineering (statics, dynamics, and solid mechanics). These courses administered a significant part of their summative assessments in a Computer-Based Testing Facility (CBTF) via the PrairieLearn system. With IRB approval we obtained all the PrairieLearn data collected in the CBTF.

2.1 PrairieLearn and the Computer-Based Testing Facility

In their simplest conception, two components comprise AIGs: a *template* (or *model*) providing the structure of the question and a computation that randomly parameterizes this template (see Figure 1) [4]. AIGs, however, can be quite sophisticated, if enabled by the authoring tool. The courses studied in this paper use the PrairieLearn LMS [26], which permits authors to write an arbitrary piece of server-side code and use the full power of HTML5/JavaScript to render a question. This enables questions to not just populate text templates, but also programmatically generate images (e.g., pictures of beams with forces in different places) and provide client-side interaction, e.g., providing in-browser CAD tools for students to design finite-state machines.

PrairieLearn can be used for both online homework and for exams. The data for this paper was drawn from exams, which we believe—due to their higher



```
a = random.randint(5, 10)        // random parameter generation
b = random.randint(5, 10)
solution = a + b                 // compute solution from params
```

```
Compute the sum:  {{a}} + {{b}}              // template
```

```
Compute the sum:  9 + 7              // problem instance
```

Fig. 1. A simple automatic item generator involving a computation that generates a pair of random numbers which are used in a template to generate an item instance.

Course	Number of item generators	Number of discrete parameters	Number of continuous parameters	Number of exams	Number of semesters	Number of students	Number of item instances	Number of unique item instances
Class A	92	213	69	3	4	1,670	40,496	12,063
Class B	116	243	269	7	4	1,532	31,538	23,353
Class C	93	398	234	6	7	1,473	58,157	46,799
Class D	77	153	269	7	4	940	21,430	18,792

Table 1. The four courses studied have, in aggregate, 378 unique AIGs.

stakes and better security—to be more reliable. The exams in question were taken in a Computer-Based Testing Facility (CBTF) [27], which is a proctored physical computer lab where the tests are available. To support classes with hundreds of students with an 85-seat computer lab, CBTF exams are asynchronous; students can choose when to take their CBTF exam during a 3-day period. As such, each student's exam is uniquely generated when they take the exam; PrairieLearn supports both AIGs for random problem parameterization and randomly selecting from a pool of AIGs for a given slot on the exam. PrairieLearn exams are not adaptive, so the sub-population receiving any given problem is an unbiased random sample of the class's population.

2.2 Detailed data specification

For each class in each semester, we obtained the information of all generated items in the form **(class ID, semester ID, exam ID, generator ID, problem parameters, student ID, score)**. The class, semester, exam, generator, and student IDs are unique identifiers that differentiate between classes, semesters, exams within a semester (retained across semesters), generators, and students. The problem parameters are a list of parameter-value pairs used by the item generator to generate an item. The value of each parameter can be an integer, a real number, a string, or a container (dictionary or array) containing further parameters. The score is an integer that is 1 if the student's submitted answer was correct and 0 if the submitted answer was incorrect. For questions in PrairieLearn that allowed multiple attempts, we only extracted the first scored student submission. Course details can be found in Table 1.

To facilitate the analysis, we performed the two preprocessing steps. First, we dropped the semester ID to merge data across semesters to obtain the best statistical power possible. We did not, however, drop exam ID (giving us 432 unique combinations of class ID, exam ID and generator ID) as some AIGs were used in multiple exams within a semester, and students' performance on an AIG might vary across the different exams. Second, we flattened containers (dictionaries or arrays) with less than 10 elements to facilitate studying the parameters in the containers independently. The 10-element cutoff prevents flattening arrays that contain hundreds of real values. After this flattening, the AIGs had a total of 1,848 parameters (4.88 parameters/AIG).

Some of these parameters had a small number of discrete values, while other were drawn from large, often continuous, ranges (e.g., floating-point numbers). These two classes of parameters required different statistical approaches and

	Parameter value		
Score	left	right	Total
0	399	275	674
1	96	250	346
Total	495	525	1,020

Table 2. Example contingency table.

so were analyzed independently. As our AIGs, in general, had both discrete and continuous parameters, we have chosen to formulate the analysis in terms of parameters to identify the parameters that have a significant impact on difficulty.

3 Discrete parameters

3.1 Analysis method for discrete parameters

For each unique combination of (class ID, exam ID, generator ID), we first considered all of the discrete parameters of the item generators. For each discrete parameter, we computed a $2 \times n$ contingency table between the score and the parameter values, where n is the number of unique values of the parameter. Table 2 shows an example contingency table where $n = 2$. With the contingency table, we applied the hybrid Fisher's exact test to obtain a p-value describing whether the score is dependent on the parameter values. The hybrid Fisher's exact test is a combination of Fisher's exact test and the chi-squared test. Specifically, a chi-squared test is performed when Cochran's rule [8] (no cell has expected counts less than 1 and more than 80% of the cells have expected counts at least 5) is satisfied, otherwise Fisher's exact test is performed. The null hypothesis of the test states that the score is independent of the parameter values. A small p-value would indicate that the null hypothesis is not likely true and thus we may want to reject it.

With the above method, the data resulted in 1,223 contingency tables. Such a large number of hypothesis tests introduces the multiple testing problem, which states that running large number of hypothesis tests without correction will unavoidably end up with rejected null hypotheses (discoveries) that shouldn't have been rejected. For example, if we run 1,000 hypothesis tests with $\alpha = 0.05$, even if all null hypotheses are true, we still expect about 50 ($= 1,000 \times 0.05$) of them to be rejected. To address this issue, we employed the Benjamini-Yekutieli procedure [5] which provides a bound on the percentage of rejected null hypotheses that shouldn't have been rejected, regardless of the dependency structure of these tests.

3.2 Results for discrete parameters

We conducted the hybrid Fisher's test on the 1,223 contingency tables and obtained the same number of p-values. We sorted the p-values from the smallest to the largest and plotted them in this sorted order in Figure 2. The way the Benjamini-Yekutieli procedure works is that it draws a line passing through the origin with slope $\alpha/(NH_N)$ where N is the number of hypothesis tests to be per-

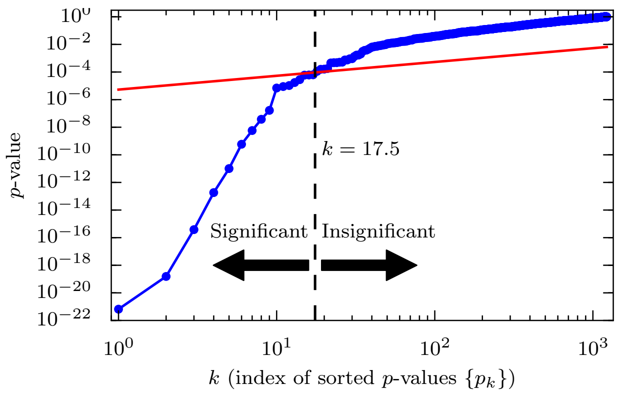


Fig. 2. Distribution of p-values for small parameters (log-log).

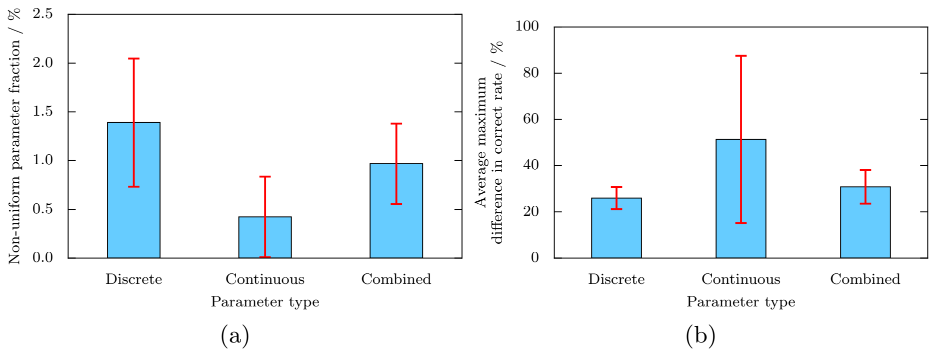


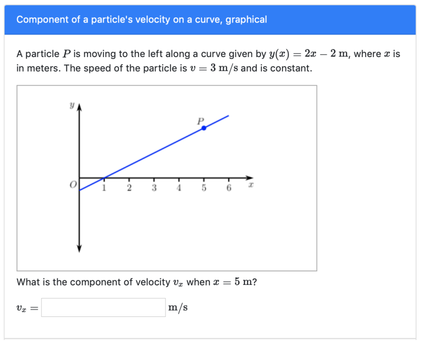
Fig. 3. (a) The non-uniform parameter fraction for each parameter type. (b) The average maximum difference in correct rate for parameters that we detected as non-uniform.

formed and H_N is the partial sum of the first N terms in the harmonic series. The Benjamini-Yekutieli procedure states that the points below this straight line correspond to significant results where the points above correspond to insignificant results. Figure 2 shows there are 17 points below the line, which suggests that there are 17 significant cases (that is, 17 discrete parameters causing non-uniform generators).

With the significant cases identified, we computed the non-uniform parameter fraction for discrete parameters to be 1.39% (95% CI [0.73, 2.05]), and plotted it as the first bar of Figure 3a. We also computed the average maximum difference in correct rate for discrete parameters, which is calculated by iterating over each non-uniform parameter discovered by discrete parameter analysis, finding the pair of parameter values that gives the maximum difference in correct rate, and then averaging this over all non-uniform parameters. The result is 25.97% (95% CI [21.14, 30.80]), and we plotted it as the first bar of Figure 3b.

3.3 Manual analysis of outliers for discrete parameters

Figure 4a shows one of the seventeen non-uniform AIGs with a problematic discrete parameter. This AIG asked students to find either the x or y component of the velocity of a particle, given its speed, a description of its path, and its



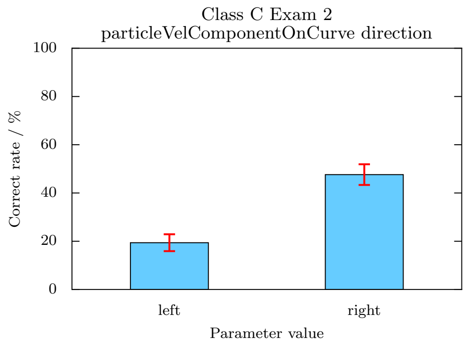
Component of a particle's velocity on a curve, graphical

A particle P is moving to the left along a curve given by $y(x) = 2x - 2\ \mathrm{m}$, where x is in meters. The speed of the particle is $v = 3\ \mathrm{m/s}$ and is constant.

What is the component of velocity v_x when $x = 5\ \mathrm{m}$?

$v_x =$ m/s

(a)



(b)

Fig. 4. (a) A non-uniform AIG with a problematic discrete parameter. (b) Percent correct as a function of a two-valued parameter.

direction—either “left” or “right”—along this path. Figure 4b shows that students are significantly more likely to answer the item generator incorrectly when the direction of motion is “left.” This result is easily explained by noting that a common mistake is neglecting to flip the sign of the velocity—from positive to negative—when the particle is moving to the left. Post-hoc analysis verified this hypothesis. As shown in Figure 4b, percent incorrect was originally 80.6% for left and 52.4% for right. If we treat all cases in which students made a sign error as correct, then these results become 53.0% for left and 50.3% for right—in other words, the significant variation disappears. Difficulty variation in the sixteen other non-uniform generators could be similarly explained.

4 Continuous parameters

4.1 Analysis method for continuous parameters

Similar to the discrete parameter case, we considered the continuous parameters for each unique combination of (class ID, exam ID, problem ID). We excluded any string– or container–valued parameters from the analysis. For each parameter, we grouped the parameter values based on the corresponding score. This divides parameter values into two groups (those answered correctly and those answered incorrectly). We then applied the Kolmogorov-Smirnov test to the two groups of parameter values to examine if the two groups of parameter values are likely to be drawn from the same distribution. If the p-value from the Kolmogorov-Smirnov test is small, then it is likely that the two groups of parameter values are from different distributions, which indicates that this particular parameter can alter the difficulty of the item generated.

With the above method, the data resulted in 947 hypotheses tests to be performed. We again applied the Benjamini-Yekutieli procedure to resolve the multiple testing problem as in the discrete parameter case.

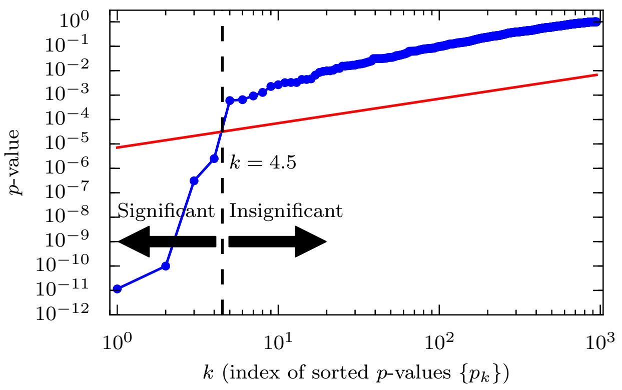


Fig. 5. Distribution of p-values for continuous parameters, log log version.

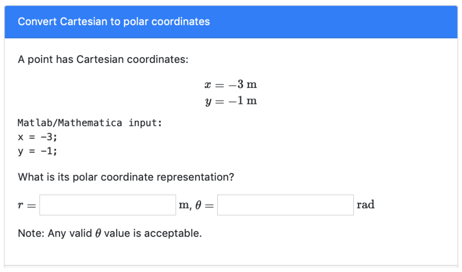
4.2 Results for continuous parameters

We again plotted the sorted p-values against the index and the straight line used by the Benjamini-Yekutieli procedure in Figure 5. As the figure shows, there are 4 points below the line, which indicates that there are 4 significant cases (that is, 4 continuous parameters producing non-uniform generators).

Similarly to the discrete parameter case, we computed the non-uniform parameter fraction for continuous parameters to be 0.42% (95% CI [0.01, 0.84]), and plotted it as the second bar of Figure 3a. We also computed the average maximum difference in correct rate by treating the continuous parameters as discrete and using the procedure in Section 3.2, to obtain 51.37% (95% CI [15.21, 87.53]), and we plotted this as the second bar of Figure 3b.

With the number of significant cases computed for both discrete and continuous parameters, we performed a t-test to examine if there is a difference in non-uniform parameter fraction for these two types of parameters. The p-value for this test is 0.0146, which indicates that there is a statistically significant difference in the probability of a discrete parameter producing a non-uniform generator versus the probability of a continuous parameter doing so. We also performed a t-test on the average maximum difference in correct rate between the two types of parameters, and the p-value is 0.1093, which suggests that there isn't enough evidence to conclude that one type of parameter is more damaging to fairness than the other.

To understand the overall behavior of parameters, we also computed the non-uniform parameter fraction for the two types of parameters combined, which is 0.97% (95% CI [0.56, 1.38]). We plotted it as the third bar of Figure 3a. We also computed the average maximum difference in correct rate for both type of parameters combined, which is 30.81% (95% CI [23.58, 38.04]). We plotted it as the third bar of Figure 3b. In total, there were 20 non-uniform generators (17 discrete and 4 continuous non-uniform parameters, with two of them appearing on a single generator).


Convert Cartesian to polar coordinates

A point has Cartesian coordinates:

$$x = -3\ \mathrm{m}$$
$$y = -1\ \mathrm{m}$$

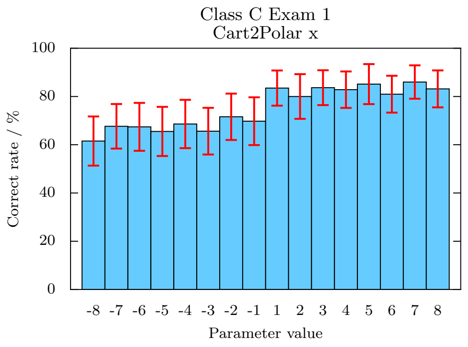
```
Matlab/Mathematica input:
x = -3;
y = -1;
```

What is its polar coordinate representation?

$r =$ ____ m, $\theta =$ ____ rad

Note: Any valid θ value is acceptable.

(a)



(b)

Fig. 6. (a) One of the non-uniform AIGs we found where a continuous parameter significantly affects difficulty. (b) Percent correct as it varies with this continuous parameter—note the change in difficulty between positive and negative values.

4.3 Manual analysis of outliers for continuous parameters

Figure 6a shows one of the four non-uniform AIGs with a problematic continuous parameter. This generator asked students to find the polar coordinates (r, θ) of a point, given its Cartesian coordinates (x, y). Figure 6b shows, in particular, that students are significantly more likely to answer this AIG correctly when $x > 0$. This result is easily explained by noting that θ must be computed with an arc-tangent and that a common mistake is not to use the four-quadrant arc-tangent function. The arc-tangent and the four-quadrant arc-tangent are the same for precisely those points for which $x > 0$. Post-hoc analysis verified this hypothesis. As shown in Figure 6b, percent incorrect was originally 32.8% when $x < 0$ and 16.8% when $x > 0$. If we treat all cases in which students used the arc-tangent instead of the four-quadrant arc-tangent as correct, then these results become 20.8% when $x < 0$ and 16.8% when $x > 0$—in other words, the significant variation disappears. The continuous parameter variation in difficulty indicated by our analysis in the other three item generators can be similarly explained.

5 Limitations

There are three limitations in the current work. The first limitation is that the continuous parameter analysis only considered parameters with numerical values. This was done so that there was a natural ordering of parameter values, which is necessary for the Kolmogorov-Smirnov test to be applied (the K-S test needs to construct a cumulative density function). For parameters that have string-valued or complex-object-valued parameters, it is not immediately clear how to apply our analysis framework. We note, however, that our analysis of discrete parameters does not suffer from this issue.

The second limitation concerns the interactions of parameters. Both the hybrid Fisher's exact test and the Kolmogorov-Smirnov test can only handle a variable with a single dimension, so they are not applicable to multi-dimensional variables. This can fail to capture interesting interactions between parameters that could make the item instance easier or harder. For example, suppose an item generator has two parameters, which give coordinates in the typical Cartesian coordinate system. It is entirely possible that item instances with points in the first and third quadrants are harder while those with points in the second and forth quadrants are easier. None of the methods that focus on a single parameter at a time would be able to capture this difference. The only way to discover this kind of interaction would be to apply methods that can handle multi-dimensional data. Unfortunately, the increase of dimensions can result in the curse of dimensionality, since the increase of dimensionality means that more data points are necessary for any method to draw a solid conclusion. This would be an interesting direction for future research.

The third limitation relates to the study's generalizability. Since the data is collected in introductory engineering courses at an R1 institution in the US, it is unclear how well these results apply to K-12 education and other institution types in other countries. It is also unclear how applicable these results are to non-STEM disciplines, as the questions in the courses studied were highly numerical.

6 Conclusion

Of the 378 AIGs that we studied, we found only 20 of them (5.3%) to have parameters that led to statistically significant difficulty variation. From this study, we conclude that there is reason to be cautiously optimistic about the potential for university STEM faculty to author uniform AIGs. We believe that AIGs have the potential to improve the efficiency and effectiveness of many educational contexts, and this result suggests that instructors can be trusted to develop AIGs.

Furthermore, we found that the source of the variation of non-uniform AIGs was frequently very easy to comprehend when the parameter that led to the variance was identified. This suggests to us that an automated analysis run after every exam, much in the way that standard psychometrics tests are run, can be used to bring problematic generators to the faculty member's attention. All of the instances we observed could be fixed by one of three methods: (1) removing a particularly problematic parameter value (e.g., 0), (2) splitting the generator into multiple generators (e.g., one for quadrants 1 and 2 and the other for 3 and 4), or (3) shifting the range of a given parameter. We did have one generator that required a few hours to track down the source of the difficulty variance, and it came down to an incorrect problem solving strategy working for part of the parameter range and not for the rest; this could be addressed by excluding the problematic part of the range from the question.

Finally, we observed that discrete parameters were more likely to be the cause of non-uniform generators than continuous parameters. One hypothesis for this observation is that, in the engineering problems we studied, the continuous parameters often represented the real-numbered values of forces or voltages or distances. Our students generally use calculators when working with these numbers and the equations necessary to solve the problems, making the precise numerical values less important in determining question difficulty.

References

1. Arendasy, M.E., Sommer, M.: Using automatic item generation to meet the increasing item demands of high-stakes educational and occupational assessment. Learning and Individual Differences **22**(1), 112–117 (2012)
2. Attali, Y.: Automatic item generation unleashed: An evaluation of a large-scale deployment of item models. In: International Conference on Artificial Intelligence in Education. pp. 17–29. Springer (2018)
3. Attali, Y., Powers, D.: Immediate feedback and opportunity to revise answers to open-ended questions. Educational and Psychological Measurement **70**(1), 22–35 (2010)
4. Bejar, I.I.: Generative testing: from conception to implementation. In: Irvine, S., Kyllonen, P. (eds.) Item Generation for Test Development. Lawrence Erlbaum Associates (2002)
5. Benjamini, Y., Yekutieli, D.: The control of the false discovery rate in multiple testing under dependency. Annals of Statistics pp. 1165–1188 (2001)
6. Bloom, B.S.: Learning for mastery. Evaluation Comment **1** (1968)
7. Cho, S.J., De Boeck, P., Embretson, S., Rabe-Hesketh, S.: Additive multilevel item structure models with random residuals: Item modeling for explanation and item generation. Psychometrika **79**(1), 84–104 (2014)
8. Cochran, W.G.: Some methods for strengthening the common χ^2 tests. Biometrics **10**(4), 417–451 (1954)
9. Embretson, S.: Generating items during testing: Psychometric issues and models. Psychometrika **64**(4), 407–433 (1999). https://doi.org/10.1007/BF02294564
10. Enright, M.K., Morley, M., Sheehan, K.M.: Items by design: The impact of systematic feature variation on item statistical characteristics. Applied Measurement in Education **15**(1), 49–74 (2002)
11. Geerlings, H., Glas, C.A., van der Linden, W.J.: Modeling rule-based item generation. Psychometrika **76**(2), 337–359 (2011)
12. Geerlings, H., van der Linden, W.J., Glas, C.A.: Optimal test design with rule-based item generation. Applied Psychological Measurement **37**(2), 140–161 (2013)
13. Gierl, M., Haladyna, T.: Automatic Item Generation: Theory and practice. Routledge (2013)
14. Glas, C.A., van der Linden, W.J.: Computerized adaptive testing with item cloning. Applied Psychological Measurement **27**(4), 247–261 (2003)
15. Haladyna, T.M.: Automatic item generation: A historical perspective. In: Gierl, M., Haladyna, T. (eds.) Automatic item generation: Theory and practice, pp. 23–35. Routledge (2013)
16. Hively II, W., Patterson, H.L., Page, S.H.: A "universe-defined" system of arithmetic achievement tests. Journal of Educational Measurement **5**(4), 275–290 (1968)
17. Irvine, S., Kyllonen, P.: Item Generation for Test Development. Lawrence Erlbaum Associates (2002)
18. Kulik, C.L.C., Kulik, J.A., Bangert-Drowns, R.L.: Effectiveness of mastery learning programs: A meta-analysis. Review of Educational Research **60**, 265 (1990)
19. van der Linden, W.J., Glas, C.A.: Elements of adaptive testing. Springer (2010)
20. Macready, G.B.: The use of generalizability theory for assessing relations among items within domains in diagnostic testing. Applied Psychological Measurement **7**(2), 149–157 (1983)
21. Meisner, R., Luecht, R., Reckase, M.: The comparability of the statistical characteristics of test items generated by computer algorithms. Tech. Rep. ACT Research Report Series Nop. 93-9, ACT, Inc. (1993)

22. Opitz, B., Ferdinand, N.K., Mecklinger, A.: Timing matters: the impact of immediate and delayed feedback on artificial language learning. Frontiers in Human Neuroscience **5**, 8 (2011)
23. Rasila, A., Havola, L., Majander, H., Malinen, J.: Automatic assessment in engineering mathematics: evaluation of the impact. In: ReflekTori 2010 Symposium of Engineering Education. pp. 37–45 (12 2010)
24. Sinharay, S., Johnson, M.S.: Use of item models in a large-scale admissions test: A case study. International Journal of Testing **8**(3), 209–236 (2008)
25. Sinharay, S., Johnson, M.S.: Statistical modeling of automatically generated items. In: Gierl, M., Haladyna, T. (eds.) Automatic item generation: Theory and practice, pp. 183–195. Routledge (2013)
26. West, M., Herman, G.L., Zilles, C.: PrairieLearn: Mastery-based online problem solving with adaptive scoring and recommendations driven by machine learning. In: 2015 ASEE Annual Conference & Exposition. ASEE Conferences, Seattle, Washington (June 2015)
27. Zilles, C., West, M., Mussulman, D., Bretl, T.: Making testing less trying: Lessons learned from operating a Computer-Based Testing Facility. In: 2018 IEEE Frontiers in Education (FIE) Conference. San Jose, California (2018)