Dimension Extraction Analysis of Student Performance on Problems

Anthony Bucci
119 Armory St.
Cambridge, MA, USA
anthony@bucci.onl

R. Paul Wiegand
Institute for
Simulation & Training
University of Central Florida
Orlando, FL, USA
wiegand@ist.ucf.edu

Amruth N. Kumar
Ramapo College of New Jersey
Mahwah, NJ, USA
amruth@ramapo.edu

Jennifer L. Albert
The Citadel
171 Moultrie Street
Charleston, SC, USA
jalbert@citadel.edu

Alessio Gaspar
University of South Florida
Tampa, FL, USA
alessio@usf.edu

Abstract

We present a preliminary investigation into applying dimension extraction methods from coevolutionary algorithm theory to the analysis of student-problem performance in a computer programming instruction context. Specifically, we explore using the *dimension extraction coevolutionary algorithm* (DECA) from coevolution and co-optimization theory (Bucci 2007), which identifies structural relationships amongst learners and tests by constructing a geometry encoding how learner performance can be distinguished in fundamentally different ways. While DECA was developed for software learners and tests, its foundational ideas can in principle be applied to data generated by human students taking real tests. Here we apply DECA's dimension-extraction algorithm to student-problem data from four semesters of an introduction to programming course where students used an online software tutor to solve a number of predesigned problems. Dimension extraction reveals structures (dimensions) that partially align with the concepts originally designed into the problems. Preliminary results suggest the structure DECA reveals is consistent when the set of students is varied.

1 Introduction

This paper presents results of a preliminary investigation into applying dimension extraction methods from coevolutionary algorithm theory to the analysis of student-problem performance in a computer programming instruction context. Our long-term goal is to improve an existing adaptive software tutor, problets.org, that is designed to help students learn basic programming concepts. We believe that extending this system to generate new problems that both vary in difficulty and are as informative as possible about how to differentiate student performance is an important first step. We begin by focusing on using existing data to understand the informativeness of student assessments.

Our jumping off point is a method that uncovers the underlying structure within student problem performance data. The idea is based on the *dimension extraction coevolutionary algorithm* (DECA) from coevolution and co-optimization theory (Bucci 2007), where a focus on maximizing the informativeness of tests has been shown to help in certain (machine) learning problems. DECA identifies structural relationships amongst learners and tests by constructing a geometry with these that demonstrates how learner performances can be distinguished in fundamentally different ways.

We apply dimension extraction to student-problem data from four semesters of an introduction to programming course where students used an on-line software tutor to work a number of predesigned problems. We show that dimension extraction produces structures that are consistent across minor variations in input, which suggests that the method finds relatively stable structures. Additionally, we show how the dimensions produced are relatively consistent with the conceptual framework with which the problems were originally designed. However, extracted performance-oriented dimensions do not perfectly align with pre-conceived concepts, which motivates the use of information-theoretic, performance-driven methods to inform adaptive intelligent tutoring systems.

2 Background

2.1 Student-Problem Performance Analysis

There are a number data-driven approaches to analyze student-problem performance, for varying purposes. Many times, researchers are interested in building a model to predict student performance based on existing data (Oyelade, Oladipupo, and Obaguwa 2010). From a more exploratory perspective, Konold *et al.* (1999) applied a multi-stage k-means clustering algorithm to try to understand the decomposition of the results of an intelligence test for children. Kerr *et al.* used cluster analysis to identify key student strategies and error patterns in an educational video game context (2012). Clustering is a useful tool in performance analysis

since it provides a way to group student or problem data based on some measure of similarity; however, these methods are highly sensitive to the measure of distance used. Learning about how the data is *structured* both amongst items within a group as well as understanding the interpretation of different groups requires additional methods.

An obvious alternative to traditional clustering methods is multidimensional scaling (MDS). MDS gives a projection and rotation in a space such that items are organized into separate dimensions. Because of this, MDS has also been applied to analyzing student performance. Davison (1994; 1996) uses MDS as an exploratory method to identify major structural patterns in student performance data. Davis and Skay (1991) compare MDS to factor analysis (e.g., Q-Factor analysis (Cattell 1967)) for uncovering structural properties of subjects versus measures in psychological studies as a means to motivate and build a response model for how different kinds of subjects respond to different kinds of psychological instruments. Ding (2001) uses a similar method in order to classify different types of students for a vocational interest test. MDS is capable of producing geometries of points in a space with useful properties; however, those geometries are in terms of the underlying (perhaps arbitrary) distance measure rather than explicitly depending on distinctions in problem/student performance.

2.2 Co-optimization & Coevolution

Co-optimization problems are distinguished from optimization problems by the presence of two or more types of entity (e.g., student or problem) that might vary and can interact with one another with a measurable outcome or score (Popovici et al. 2012). This is directly analogous to student-problem performance analysis. In this paper, we have computer science students and a variety of computer programming problems. A given student might attempt and receive outcomes on some subset of problems. This means that the relative performance among students is determined by the problems with which they interact, just as is so in co-optimization problems.

Foundational theory in co-optimization and coevolutionary algorithms concerns the development *dimension extraction* methods (Bucci, Pollack, and de Jong 2004; de Jong and Bucci 2006; Bucci 2007; de Jong and Bucci 2008). Abstractly, this line of work posits that the information gleaned from interactions, for instance the scores of students solving problems, can be decomposed into a vector-space-like *coordinate system*. Within a coordinate system, there are potentially *multiple axes* or *dimensions*, consisting of a linearly ordered subset of the set of all entities. Entities further along a given axis are "no worse than" those preceding it. Across two different axes, entities are incomparable to one another, in the sense that an entity on one axis will be better in some ways, but worse in other ways, than an entity on another axis. This method has been used for a number of purposes, including automatically identifying key conceptual tactics in the game of Nim (de Jong and Bucci 2008).

Wiegand *et al.* (2016) recently used dimension extraction analysis as a means of uncovering problems in a dataset that are both challenging and *informative* — that is, problems for which not only do fewer students get right, but also help differentiate the general performance of students on problems. However, this work does not provide insight into the effectiveness of dimension extraction in terms of how it orders items *within* each dimension.

2.3 Applications to Programming Education

Extracting the underlying dimensions of a large interaction space opens new possibilities when applied to educational data. In the case of the interaction space between student and programming practice problems, such dimensions capture underlying key concepts.

Dimension extraction offers a data-driven approach that supplements the research designs traditionally leveraged in this type of work. For instance, Goldman and his colleagues (2008) used a Delphi process to identify and rank a list of topics based on their importance and difficulty. Meanwhile, Hertz and Ford (2013) instead surveyed CS1/CS2 instructors in order to measure the amount of time spent on various topics. This measurement was then correlated with student performance.

Furthermore, dimension extraction techniques also enable us to address the issue of granularity (Porter, Taylor, and Webb 2014) by identifying low-level concepts on most of the topics typically covered in introductory programming.

While several concept inventories have been attempted on various Computer Science subjects (e.g., digital logic, operating systems, algorithms) digital logic (Herman, Zilles, and Loui 2014) algorithms (Vahrenhold and Wolfgang 2014; Taylor et al. 2014), none has yet been completed on introductory programming. To create a concept inventory for introductory programming, assessment items must be created on a continuum of specific concepts within each topic that students do not understand. Therefore, dimension extraction is a necessary first step to achieve a better understanding of key concepts and how they may lie on the continuum.

3 Methods

3.1 Problets.org

The data analyzed in this study was gathered through *Problets.org* Intelligent Tutoring System. This afforded us the benefits of relying on a tool that has been used by third-party educators in their introductory courses for over a decade and that has been continually and extensively evaluated (e.g., (Kumar 2015a; 2015b)).

Students used different software tutors, named *problets*, each dedicated to covering the various concepts related to a specific topic typically studied in introductory programming courses. The concepts address different skills: evaluating expressions, tracing programs (e.g., predicting the output), debugging programs and identifying the state of variables in a program.

Each problet adapts the sequence of problems to the learning needs of the student. A predefined set of *pretest problems* is first used, one per concept covered by the problet, to gauge the student's prior knowledge. Thereafter, students are only presented with *practice problems* on the concepts on which they solved the pretest problem incorrectly.

The tutor uses parametrized templates to generate problems. In a parametrized template, the problem is described in terms of meta-variables with constraints, e.g., a template $\langle r_1 \in \{2, \ldots, 5\}\rangle + \langle r_2 \in \{6, \ldots, 9\}\rangle$ is used to generate an addition expression, r_1 being chosen randomly in the range 2 through 5 and r_2 in the range 6 through 9. So, typically, no two students see the same problem and no student sees the same problem twice. Our data analysis was done in terms of problem templates rather than specific problems.

In this study, we used pretest and practice data of students who used a problet on arithmetic expression evaluation. This problet featured 25 concepts. Arithmetic expression evaluation is especially suitable for empirical analysis because a typical problem tests multiple concepts and concepts are often inter-connected, e.g., 3 + 4 * 5 tests correct evaluation of addition and multiplication, as well both their precedence. The precedence of addition and subtraction are the same and can be treated as one "bundled" concept. Because of the one-to-many relationship between problems and concepts, extracting the state of knowledge of a student based on the student's performance on a series of problems is not a straightforward task.

3.2 Population & Sample

Participants were students enrolled in introductory programming courses offered by high schools, community colleges and four-year colleges. Problets were used for after-class assignments after the related topics had been covered in class. Our data was gathered on one problet that dealt with the topic of arithmetic expressions. It was used by a total of 1319 students during the fall 2012, fall 2013, spring 2013 and spring 2014 semesters.

Table 1 summarizes the distribution of our sample with respect to several factors. Problets users were free to opt to not provide information regarding any but two of the factors being considered. Both the programming language used by the tutor, i.e. "Language" factor, and the type of institution, i.e. "School" factor, were determined when the instructor requested access to *Problets.org* for his or her students.

For each factor, we therefore specify the percentage of the students who did not provide the corresponding data in column *% skip*. These percentages are below 22% for all factors, thus indicating that the resulting information is most likely representative of the entire sample. We also provide, for each factor, the distribution of responses. All results are rounded to the nearest integer.

From this sample, we extracted a subset of students who actually interacted with all the practice problems featured in our study. This allowed us to focus on a rectangular matrix providing pass or fail information with respect to all available tests. This resulted in a subset of 17 students that characteristics are summarized in table 2, using the same format than table 1. As mentioned above, this reduction in data stems from simplifying conditions of our preliminary study meant to baseline our ideas. In studies following this paper, we aggregate at a concept level and can apply these methods to hundreds of students.

3.3 Dimension Extraction for Problem Analysis

Dimension Extraction Coevolutionary Algorithm (DECA) (de Jong and Bucci 2006) is a coevolutionary algorithm that uses a dimension-extraction method, as discussed in Sect. 2.2. Coevolutionary algorithms search through *candidate solutions* and *tests*, but here we harness DECA's internal dimension-extraction to examine the performance relationships between students and problems. Dimension-extraction can be done from the perspective of students or problems; however, in this paper we confine our attention to problem analysis.

We applied dimension extraction to the subset of the four-semester dataset discussed in Sect. 3.2. The result of this *problem analysis* is a coordinate system of problems, where each dimension ordered a subset of the problems. Problems further along a dimension are not passed by the same students as those lower on the dimension, plus possibly more. Problems on different dimensions are those on which students performed incomparably in a multiobjective sense. Intuitively, each dimension corresponds to a different conceptual way to differentiate student performance, and the problems that are highest on each dimension are the problems most students did not solve.

Application of this kind of analysis to the performance of students on computer programming problems potentially provides several sources of insight. First, the number of dimensions extracted from problem analysis gives an implicit measure of the number of different ways that problems distinguish student performance. Additionally, by examining how problems are arranged within a dimension, we can can learn which problems are consistent in how they distinguish performance, as well as how such problems *scaffold* (get progressively harder) along any given dimensional axis.

What remains to be determined is what relationship these extracted dimensions have to the concepts that were *a priori* designed into the problem sets to begin with, or the extent to which small changes in the students considered for the analysis will change the structure returned by dimension extraction. This latter issue is an issue of *consistency* with respect to the algorithm.

We address the question of alignment by comparing the dimension extraction results to a concept map given by the subject matter expert who designed *problets.org*, described in Sect. 4.1. We address the consistency issue conducting leave-one-out and leave-two-out validation extractions — that is, we remove a student (or two) and run the extraction again, doing so for each student (and pair of students). The results of these comparisons are described below.

4 Results

4.1 Pedagogical Interpretation

DECA discovered 12 orthogonal ways in which problem templates fundamentally distinguished student performance. Problem templates along a given dimension can be compared to one another — later templates are strictly "harder" than earlier templates. Problem templates on different dimensions are, in a multiobjective sense, incomparable.

Factor	% skip	Responses
Language	0%	64% Java, 35% C++, 1% VB
School	0%	88% Undergraduate, 9% community college, 2% K12
Gender	16%	62% male, 22% female
Race	20%	51% Caucasian, 15% Asian, 4% Hispanic/Latino, 5% African American, 6% Other
Major	18%	25% Computer Science, 25% Engineering, 4% Social Sciences, 11% Other Science (e.g. Physics, Chemistry, Biology...), 18% Other
Year	16%	33% Freshman, 24% Sophomore, 13% Junior, 9% Senior, 1 % Graduate, 2% Others

Table 1: Characterization of our Sample

Factor	% skip	Responses
Language	0%	59% Java, 35% C++, 6% VB
School	0%	65% Undergraduate, 29% community college, 6% K12
Gender	12%	59% male, 29% female
Race	12%	53% Caucasian, 6% Asian, 6% Hispanic/Latino, 12% African American, 6% Native American, 6% Other
Major	12%	41% Computer Science, 35% Engineering, 6% Business, 6% Other
Year	12%	29% Freshman, 35% Sophomore, 18% Junior, 6% Senior

Table 2: Characterization of our Sample's subset

- Seven dimensions consisted of two or more problem templates — the longest dimension consisted of 5 problem templates. This was followed by one with 4 templates, two with 3 templates and finally, three with 2 templates.
- Five dimensions consisted of one template each. It is conceivable that with additional data, these problem templates could seed longer dimensions, and their "singleton" nature is more an artifact of the sample size. Therefore, we ignored these dimensions in our analysis.

We analyzed the concepts covered by the problem templates in the seven non-singleton dimensions. The concepts themselves were grouped by similarity as follows:

- Addition and subtraction being similar, we grouped their correct evaluation, precedence and associativity.
- The precedence and associativity of multiplication, division and remainder operators being the same, we grouped them together.
- Each of the following was treated as one conceptual group: correct evaluation of multiplication; correct evaluation of division — both integer and real division; correct evaluation of remainder operation — with integer and real operands; evaluation of parentheses; divide-by-zero error; and coercion necessitated by real operands.

So, in all, we analyzed the 7 dimensions yielded by DECA along 8 conceptual groups, as shown in Table 3. In the table, each dimension is identified by the number of problem templates in it (column 1). Our analysis showed:

- The dimension consisting of four templates was dominated by coercion ("Real" column in the table).
- One dimension consisting of three templates was dominated by correct evaluation of addition/subtraction ($+, -$ column in the table). The other dimension was dominated by coercion and evaluation of remainder operator with real operands, the latter being one of the two concepts in the column $\%$ in the table.
- A dimension consisting of two templates was dominated by integer division, which was one of the two concepts in the column titled $/$ in the table. Another was dominated by divide-by-zero error ($\#/0$ in the table).

In the above cases, the dimensions extracted by DECA grouped problem templates around concepts or conceptual groups that corroborate educators' intuition. This is interesting in that the problem templates in each dimension were not originally designed to serve the conceptual groups identified by DECA. But, given that each non-trivial problem with two or more operators inherently co-opts multiple concepts, DECA "discovered" the salient concepts common to all the templates in each dimension.

The conceptual groupings of the two remaining dimensions, however, are not as clear:

- The largest dimension with five templates was dominated by multiple conceptual groups: correct evaluation of addition/subtraction, precedence of multiplication/division/remainder and evaluation of remainder.
- The remaining dimension with two templates did not have a dominant conceptual theme (last row in the table).

Neither dimension seems focused around any particular conceptual groups. Since five out of seven multi-problem dimensions extracted by DECA cohered around clearly identifiable concepts, it is reasonable to assume that the concepts pointed to by these two dimensions, while not intuitively discernible, are important for learning all the same.

Note that there was no exclusivity among the conceptual groups or the dimensions extracted by DECA:

Dim Prob	$+, -$	$*, /, \%$	$*$	$/$	$\%$	()	#/0	Real
5	5	5	1	1	4	3	0	3
4		1	2	1	1			**4**
3	2	2		2	**2**			**3**
3	**5**	1		1				1
2	4	4	1	**2**	1			
2	2	1					**2**	
2		1	1	1			1	1

Table 3: Number of occurrences of each conceptual group in the different dimensions. For example, the dimension in the last row contains two problems: 5 * 3.0, which accounts for the entries in * (correct evaluation of multiplication) and Real (real operand) columns, and 9 / 3 / 0, which accounts for the entries in *,/,% (associativity of division), / (correct evaluation of division) and #/0 (divide-by-zero) columns.

- The problems in a dimension dominated by a conceptual group were not limited to just that group — they tested concepts from other groups also, although not in as concerted a manner as the dominant group.
- A conceptual group could dominate multiple dimensions.

This highlights the interconnected nature of the domain (i.e., each problem covers multiple concepts) and makes the dimensions extracted, and hence, the concepts discovered by DECA all the more interesting. In this sense, these results are promising.

4.2 Consistency

While running DECA on the full dataset produced 12 dimensions, it is not clear the algorithm would produce similar results on similar datasets. To begin to address DECA's sensitivity to input data, we performed leave-one-out and (exhaustive) leave-two-out cross validation to estimate the mean number of dimensions found by DECA. The results are summarized in Table 4.

To be specific, for each fold in the leave-one-out cross validation, we produced a subset of the original dataset consisting of 16 of the 17 students and all 27 templates. We ran DECA on this subset and recorded the number of dimensions extracted. We repeated this for each of the 17 students, leaving out a different student for each iteration. We thus had 17 number of dimension estimates. The first line of Table 4 reports the sample mean, sample standard deviation, minimum and maximum for these estimates.

The leave-two-out cross validation was similar. However, each subset was created from the original dataset by omitting 2 distinct students. We ran DECA on each dataset, for a total of 272 estimates of number of dimensions. The second line of Table 4 reports the sample mean, sample standard deviation, minimum and maximum for these estimates.

It is worth remarking that the leave-two-out cross validation omits over 10% of the original dataset while still estimating a number of dimensions close to the 12 found for the full dataset.

5 Discussion

DECA was originally developed in the context of coevolutionary algorithms that were intended, among other things, to adaptively train computer programs to perform a desired task. In that context, DECA has shown a number of desirable theoretical and pragmatic properties. The dimension-extracting component of the algorithm was extended in (de Jong and Bucci 2008) and applied to analyzing instances of the game Nim, where it was found the extracted dimensions corresponded closely to our intuitions about what makes for strong or weak moves in that game. The present work can be considered a first assessment of whether the benefits DECA has demonstrated in the context of coevolutionary algorithms transfer to the domain of adaptive intelligent tutoring systems.

In Sect. 4.1 we analyzed extracted problem template dimensions and found relationships with the concepts built into the problem templates by design. The relationship was not straightforward–some dimensions found by DECA appeared to be dominated by a single concept, while others mixed multiple concepts–and further work is needed to elucidate what, if anything, these dimensions might signify pedagogically. We also observed a "compression" in the sense of (de Jong and Bucci 2008): 12 dimensions were found among the 27 problem templates. We feel that while not conclusive, these results suggest our original hope of transferring DECA's strengths to adaptive intelligent tutoring systems shows promise.

In Sect. 4.2 we performed a simple analysis of the consistency in the number of dimensions DECA finds in this dataset. We observed that while 12 dimensions were found in the full dataset containing 17 students, anywhere from 9 to 12 dimensions might be found in a subset of 15 or 16 students, with an average number of dimensions found roughly 11. We have not performed comparisons between the dimensions found in subsets of the data with the dimensions found in the full dataset, but plan to do so in future work.

We should emphasize that the dataset used, consisting of 27 problem templates and 17 students, is small. DECA found 12 dimensions in this data, but 5 of them consisted of a single template and did not lend themselves to comparison with intuitive concepts. In future work we hope to analyze larger datasets and develop appropriate measures of statistical significance that allow for stronger inference about the number of dimensions found and their relationships with known concepts. In our view the present work is best viewed

	sample mean number of dims	s^2 number of dims	min number of dims	max number of dims
leave-one-out	11.06	0.43	10	12
leave-two-out	10.96	0.60	9	12

Table 4: Sample mean and sample standard deviation of extracted number of dimensions in leave-one-out and leave-two-out cross validation

as exploratory and suggestive, not predictive.

6 Acknowledgments

This material is based in part upon work supported by the National Science Foundation under awards #1504634, #1502564, and #1504634.

References

Bucci, A.; Pollack, J.; and de Jong, E. 2004. Automated extraction of problem structure. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '04, 501–512.

Bucci, A. 2007. *Emergent Geometric Organization and Informative Dimensions in Coevolutionary Algorithms*. Ph.D. Dissertation, Brandeis University, Boston, MA.

Cattell, R. 1967. The three basic factor analysis research designs: Their interrelations and derivatives. In *Problems in human assessment*. McGraw-Hill.

Davison, M., and Skay, C. 1991. Multidimensional scaling and factor models of test and item responses. *Psychological Bulletin* 110(3):551–556.

Davison, M. 1994. Multidimensional scaling models of personality responding. In *Differentiating normal and abnormal personality*. Springer.

Davison, M. 1996. Multidimensional scaling and factor models of test and item responses. Technical report, University of Minnesota.

de Jong, E., and Bucci, A. 2006. DECA: Dimension extracting coevolutionary algorithm. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, 313–320. New York, NY, USA: ACM.

de Jong, E., and Bucci, A. 2008. Objective set compression: Test-based problems and multi-objective optimization. In *Multi-Objective Problem Solving from Nature: From Concepts to Applications*. Springer.

Ding, C. 2001. Profile analysis: multidimensional scaling approach. *Practical Assessment, Research & Eval.* 7(16).

Goldman, K.; Gross, P.; Heeren, C.; Herman, G.; Kaczmarczyk, L.; Loui, M.; and Zilles, C. 2008. Identifying important and difficult concepts in introductory computing courses using a delphi process. *ACM SIGCSE Bulletin* 40(1):256–260.

Herman, G. L.; Zilles, C.; and Loui, M. C. 2014. A psychometric evaluation of the digital logic concept inventory. *Computer Science Education* 12(4):277–303.

Hertz, M., and Ford, S. 2013. Investigating factors of student learning in introductory courses. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, 195–200. New York, NY, USA: ACM.

Kerr, D., and Chung, G. 2012. Identifying key features of student performance in educational video games and simulations through cluster analysis. *Journal of Educational Data Mining* 4(1).

Konold, T.; Glutting, J.; McDermott, P.; Kush, J.; and Watkins, M. 1999. Structure and diagnostic benefits of a normative subtest taxonomy developed from the wisc-iii standardization sample. *J. of School Psyc.* 37(1):29–48.

Kumar, A. 2015a. The effectiveness of visualization for learning expression evaluation. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, 362–367. New York, NY, USA: ACM.

Kumar, A. 2015b. Solving code-tracing problems and its effect on code-writing skills pertaining to program semantics. In *Proceedings of the 20th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2015, Vilnius, Lithuania, July 6-8, 2015*, 314–319.

Oyelade, O.; Oladipupo, O.; and Obaguwa, I. 2010. Application of k-means clustering algorithm for prediction of students academic performance. *International Journal of Computer Science and Information Security* 7(1):292–295.

Popovici, E.; Bucci, A.; Wiegand, R.; and de Jong, E. 2012. Coevolutionary principles. In *Handbook of Natural Computing*. Springer. 987–1033.

Porter, L.; Taylor, C.; and Webb, K. 2014. Leveraging open source principles for flexible concept inventory development. In *Proc. of the 2014 conference on Innovation & technology in computer science education*, 243–248. ACM.

Taylor, C.; Zingaro, D.; Porter, L.; Webb, K.; Lee, C.; and Clancy, M. 2014. Computer science concept inventories: past and future. *Computer Science Educ.* 24(4):253–276.

Vahrenhold, J., and Wolfgang, P. 2014. Developing and validating test items for first-year computer science courses. *Computer Science Education* 24(4):304–333.

Wiegand, R.; Bucci, A.; Kumar, A.; Albert, J.; and Gaspar, A. 2016. A data-driven analysis of informatively hard concepts in introductory programming. In *Proceedings of the 47th ACM Technical Symposium on Computer Science Education*, SIGCSE '16. New York, NY, USA: ACM.