International Journal of Artificial Intelligence in Education (2022) 32:196–232
https://doi.org/10.1007/s40593-021-00252-4

ARTICLE

Complexity and Difficulty of Items in Learning Systems

Radek Pelánek[1] · Tomáš Effenberger[1] · Jaroslav Čechák[1]

Received: 3 August 2020 / Revised: 19 March 2021 / Accepted: 30 March 2021
Published online: 4 May 2021

Abstract

Complexity and difficulty are two closely related but distinct concepts. These concepts are important in the development of intelligent learning systems, e.g., for sequencing items, student modeling, or content management. We show how to use complexity and difficulty measures in the development of learning systems and provide guidance on how to think, reason, and communicate about these notions. To do so, we propose a pragmatic distinction between difficulty and complexity measures. At the same time, we acknowledge the limitations of any simple distinction and discuss several potentially confounding issues: context, biases, and scaffoldings. We also provide an overview of specific measures and their applications in several educational domains and a detailed analysis of measures for problems in introductory programming.

Introduction

Both research and development in artificial intelligence in education are often concerned with questions regarding easiness and difficulty. We try to estimate student skills using student modeling techniques (Pelánek, 2017) in order to present students with appropriately challenging problems. We aim to sequence problems from easier to more difficult to achieve the state of flow (Kiili et al., 2012; Csikszentmihalyi & Csikszentmihalyi, 1992). We design techniques for providing hints for difficult steps (Aleven et al., 2016). Analysis of difficulty is also useful for intelligence amplification (Baker, 2016), as it can point to misspecified problems or suggest ideas for the potential addition of new content, which will make the learning curve smoother (Huang et al., 2020).

But what exactly do the terms "easy" and "difficult" mean? Is there a difference between "difficulty" and "complexity" of items? How do we exactly conceptualize these notions for the purposes of the development of adaptive learning systems? These are the main question that we address in this work.

✉ Radek Pelánek
xpelanek@fi.muni.cz

[1] Faculty of Informatics, Masaryk University, Brno, Czech Republic

The notions of difficulty and complexity are sometimes used as synonyms. However, the literature offers a basic distinction between them (e.g., Mesmer et al. (2012); Beckmann et al. (2017)): complexity is an intrinsic property of a task (item) and is given by its internal structure, whereas difficulty is related to student-task interaction (performance of students). Both of these concepts can be measured in many ways. Difficulty and complexity measures have many applications in the development of adaptive learning systems. It is thus important that we think, reason, and communicate about these notions properly.

One typical application is the sequencing of items. A common pedagogical principle applied in most learning systems (both adaptive and static) is to sequence topics and problems from easier to more difficult (Scheiter & Gerjets, 2002). But what exactly should be the criterion that is used for sequencing? For example, Brusilovsky (1992) considers sequencing with respect to complexity, whereas Chen et al. (2006) propose sequencing based on item difficulty and item similarity. Both of these approaches have disadvantages. Consider, as an example, the sequencing of program comprehension items in introductory programming. When sequencing with respect to complexity, we may put early in the sequence items that use few simple programming concepts but are difficult to answer due to high cognitive demands. When sequencing with respect to difficulty, we may put early in the sequence items that contain advanced programming concepts, but for which the answer is easy to guess due to the structure of the item. This is a specific example where it is useful to explicitly distinguish between complexity and difficulty measures and to consider their suitable combination.

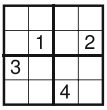
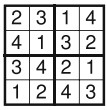
Complexity and difficulty measures have many other applications. Complexity can be used as a proxy for difficulty to alleviate the "cold start" problem in student modeling and check for biases in student data (Effenberger et al., 2019). Estimation of difficulty based on complexity is also useful for the filtering of automatically generated content (Kurdi et al., 2020). Complexity and difficulty measures are also useful for intelligence amplification (Baker, 2016) or design-loop adaptivity (Aleven et al., 2016).

In this work, we consider the complexity and difficulty of items in learning systems, focusing on well-structured problems, i.e., problems with clear instructions and an objective solution that can be checked algorithmically. This covers a wide range of items used in learning systems; specific examples are given in Table 1. For these items, complexity measures are, for example, the length of a text, the count of numbers in an item statement, the number of concepts in an item solution, or the number of steps necessary to solve an item. Difficulty measures are based on students' performance; the basic ones are the failure rate and median response time.

Our aim in this overview paper is to provide a solid starting point that can provide guidance and inspiration for the usage of complexity and difficulty measures in the development of learning systems. To this end, we review literature, provide conceptualizations relevant to the AIED field, discuss specific examples from several educational domains, and perform an in-depth analysis of a particular dataset.

We propose a simple distinction between complexity and difficulty measures: a complexity measure is based only on the item description, it does not use any data about student performance; a difficulty measure is based only on data about student performance, it does not use any data about an item beyond its identification number. This is a pragmatic view that is useful for many practical applications. However, we also highlight its limitation: aspects like the role of context, biases, and scaffoldings make the distinction between complexity and difficulty more convoluted. Our proposal

Table 1 Examples of well-structured problems that fall into the scope of this work

item statement	item solution
What is the capital city of France? a) Berlin, b) Paris, c) London	b)
Does she [play/plays] the guitar?	play
$2^3 + 3^2 - 2 \times 3 = ?$	11
$3x - 1 = 14, x = ?$	5
Peter eats three squares of chocolate every day. How many squares of chocolate does he eat in a week?	21
How many ways can you arrange the letters in 'Rococo'?	60
	

Write a function that for a given `n` prints all its divisors.	`def divisors(n):`
	`  for i in range(1, n+1):`
	`  if n % i == 0:`
	`  print(i)`

is based on previous conceptualizations of complexity and difficulty, which are reviewed in Section 2. We discuss aspects specific to educational items and our pragmatic approach to dealing with complexity and difficulty measures in Section 3.

We complement the general discussion with specific examples from several diverse settings. We have chosen settings for which the current literature provides ample measures, applications, and results: reading and language learning, multiple-choice questions, word problems in mathematics, logic puzzles. In Section 4, we discuss specific complexity and difficulty measures for these domains and outline specific applications of these measures.

We also provide a detailed analysis in a less studied domain: introductory programming. We use extensive data from five types of programming exercises, which allows us to explore the subtle issues involved in the computation of complexity and difficulty measures. The use of five types of exercises also allows us to explore the generalizability of results. These results are reported in Section 5.

Throughout the paper, we stress the need to take into account the purpose of complexity and difficulty measures. As the examples discussed above illustrate, there are many types of use cases, and each of them requires measures with slightly different properties. We thus do not aim at finding *the correct measure* but rather at a wide spectrum of measures and tools for choosing the ones suitable for a specific situation. We elaborate on this point in the final discussion.

Related Work

In this section, we discuss works relevant to general issues concerning the difficulty and complexity concepts. Works concerning specific complexity and difficulty measures for

particular educational domains are discussed in Section 4 and Section 5, where we describe case studies.

Task Complexity and Difficulty

The distinction between complexity and difficulty and characterization of these concepts has been discussed in several previous works, typically with the focus on general "tasks" and complex, ill-structured problems.

One of the first systematic discussions of task complexity was provided by Campbell (1988), who contrasted several views of complexity (as a psychological experience, as a task-person interaction, as a function of objective characteristics) and provided a typology of complex tasks. A more recent review along similar lines is provided by Liu and Li (2012), who propose a general overview of task complexity (in a very broad setting). They define task complexity as "the aggregation of any intrinsic task characteristic that influences the performance of a task." They also propose general "complexity dimensions." Three of these dimensions—size, variety, and relationship—are directly relevant to educational items; these correspond to aspects like the length of an item statement or solution, the number of concepts used, and interactions among concepts. For these dimensions, the interesting question is how to measure them precisely; with this respect, unfortunately, the general framework cannot provide any guidance. Several other described complexity dimensions—ambiguity, variability, unreliability, and incongruity—are not directly relevant to the current discussion because we consider only well-structured problems, where these factors should not be present (or only in a limited manner). Finally, novelty and temporal demand dimensions are related to the contextual aspect of complexity that we discuss later.

Robinson (2001) distinguishes task complexity (cognitive factors), task difficulty (learner factors), and task conditions (interaction factors). He considers tasks in the context of language learning, and the main motivation is the sequencing of tasks in the syllabus. In the context of language tasks, Mesmer et al. (2012) explicitly distinguish text difficulty (based on the performance of readers) and text complexity (based on textual elements) and highlight that treating these terms as synonymous "conflates causes with effects." Amendum et al. (2018) notes the importance of a specific task with the text.

Beckmann et al. (2017); Beckmann and Goode (2017) explicitly focus on the distinction between complexity and difficulty. They consider complexity to be a cognitive concept that reflects cognitive demands imposed by the task and difficulty to be a psychometric concept that reflects the performance of individuals. They discuss these notions specifically for complex problem solving (controlling a dynamic system with feedback loops).

Complexity and Educational Taxonomies

Well-known educational taxonomies are related to the complexity concept. Bloom's taxonomy and its revision (Bloom et al., 1956; Anderson et al., 2000) describe the taxonomy of educational objectives. The taxonomy divides cognitive objectives into levels (knowledge, comprehension, application, analysis, synthesis, evaluation) that are sorted from easier to more complex (although their complexity may overlap).

The SOLO taxonomy (Biggs & Collis, 1981) focuses on the relation between concepts and is used particularly to evaluate open-ended questions. Its focus is on the complexity of

answers (not questions). The depth of knowledge framework (Webb, 1997) consists of four levels of increasing cognitive complexity: recall and reproduction, application, strategic thinking, and extended thinking. The Knowledge-Learning-Instruction framework (Koedinger et al., 2012) provides a taxonomy of knowledge components and learning processes and links the complexity of knowledge components to instructional design.

Although all of these taxonomies are related to complexity, they are very general and thus coarse-grained. Such taxonomies are useful particularly for comparing different types of tasks. They do not provide guidance on how to measure complexity numerically. Particularly, when we consider tasks of one type (e.g., different linear equations), they do not differentiate between them, even though there may be significant differences between individual instances.

Difficulty and Performance

Difficulty is usually used in connection with human performance on a task. A basic approach to quantifying difficulty is to consider the correctness of answers. A simple difficulty measure is a failure rate (the ratio of answers that are incorrect). Difficulty is a key concept in both adaptive testing and practice. In the context of testing, difficulty is usually measured using Item Response Theory (IRT) models (De Ayala, 2008). In an adaptive practice system, student modeling approaches like the Elo rating system (Pelánek, 2016) or Additive Factors Model (Cen et al., 2006) include difficulty parameters. Wauters et al. (2012) compare different methods for estimating difficulty (e.g., IRT, Elo, student feedback, expert rating).

The correctness of an answer is not the only source of data for measuring difficulty. Another common source of difficulty data is response time (Pelánek, 2017). In the case of problem-solving activities (programming or solving logic puzzles like Sudoku), it is not very natural to analyze the correctness of answers—rather than "incorrectly answered problems," it is more natural to talk about "not solved yet problems." The primary measure of difficulty in these cases is thus problem-solving time (Pelánek & Jarušek, 2015). We may also consider difficulty both with respect to correctness and time (Van Der Linden, 2009).

Context and Biases

Educational items are not solved in isolation. Students typically answer an item in some specific context, e.g., after previously solving several other items. This context influences student behavior and can have a significant effect on how we analyze (or should analyze) the complexity and difficulty of items.

Contextual effects are studied in many areas of human behavior, e.g., social and psychological research clearly shows the significant influence of question order in surveys (Schwarz & Sudman, 2012). In education, a classic illustration of the role of context is the Einstellung effect (Luchins, 1942) — an influence of previous attempts on the selected approach in problem-solving activities (Lovett & Anderson, 1996). Another well-known effect illustrating the role of context is the difference between blocking and interleaving practice (Taylor & Rohrer, 2010). By changing the ordering of items, their difficulty and long-term impact also change: interleaving items of different types decreases immediate success while leading to better long term retention.

The contextual effects can create biases in the collected data, which we use for analysis. Common biases in educational data are caused by fixed ordering of items or by the use of mastery learning, which leads to attrition bias (Nixon et al., 2013; Goutte et al., 2018; Pelánek, 2018; Čechák & Pelánek, 2019). The commonly used techniques in educational data mining or psychometrics typically do not take contextual effect into account, e.g., local item independence is a key assumption in standard item response theory models (Baker, 2001). There are, however, extensions of the basic models that consider contextual effects (Keller et al., 2003; Sao Pedro et al., 2013) or address biases and missing data (Finch, 2008).

Complexity and Difficulty of Educational Items

Based on the review of the literature and our experience with the development of learning systems, we propose a systematic approach to dealing with complexity and difficulty measures for well-structured problems in learning systems.

Item Anatomy and Student Data

Before discussing the complexity and difficulty of items, let us clarify what we understand under "item" and what kind of data can be used to measure complexity and difficulty.

In our discussion, we aim to strike a balance between generality and specificity. We want our discussion to be relevant to many different types of educational content, and at the same time, we want to avoid vague formulations. Thus, we consider only well-structured problems with clear solutions, which are relatively short (the response time is between few seconds and few minutes) and have relatively simple interaction (mostly basic selected response and constructed response). This list of constraints may seem restrictive, but it covers a large portion of items used in education (a sample of illustrations is provided in Table 1).

This type of item has two major parts:

- an *item statement* (what is shown to students): natural language text, image, sound recording, or a structured input describing interactive elements (e.g., a JSON file describing a logic puzzle),
- an *item solution* (what is expected as an answer from student): either specified explicitly (a number, a short text), or implicitly (a code and testing data in programming exercise, a goal state in a logic puzzle).

Each item is used within a learning system in some *context*, i.e., what other items do students solve before and after the given item. This context is determined typically by a mapping of items to knowledge components (students typically practice a sequence of items from a single knowledge component), an algorithm for item selection (e.g., fixed sequence, random, adaptive), and the algorithm's parameters (e.g., specific ordering for a fixed sequence).

The item difficulty is based on student performance; we thus also need *log data* on student answers. The core data on student performance are:

- the identification of a student and item,
- the correctness of an answer (binary value),
- the specific value of an answer,
- response time.

Richer performance data are typically available, but these are often in some way specific to a particular exercise (e.g., information about "solution quality," or information about multiple attempts if the exercise allows repeated attempt after a mistake). For the sake of generality, we will focus on the core data that are relevant to a wide range of exercises.

Complexity versus Difficulty Concepts

In line with the previous work on task complexity and difficulty (Liu & Li, 2012; Beckmann et al., 2017), we propose the following definitions as a basic distinction between complexity and difficulty of educational items:

- Item *complexity* is an intrinsic item characteristic, which aggregates item aspects that influence how students solve the item. Complexity is concerned with the structure of the item itself.
- Item *difficulty* describes how hard it is to solve the item for students.

Note that it is necessary to distinguish between concepts and measures. For example, "complexity as a concept" is a multifacet aggregate (and as such is treated in many of the previous works), whereas "complexity measure" is a specific way to compute a single number per item, which characterizes one aspect of "complexity as a concept." We focus on measures, which are directly applicable to the development of learning systems.

A pragmatic view of the basic difference between complexity and difficulty is to consider the data used to compute a specific measure (Fig. 1). If only an item description is used, then it is a complexity measure; if only the log of student actions is used (where items are identified solely by their IDs), then it is a difficulty measure. As Fig. 1 shows, this view is a simplification due to the presence of one important aspect—the context in which an item is used.

Elementary Complexity and Difficulty Measures

Let us start by discussing basic measures of complexity and difficulty, which do not take the issue of context into account.

Complexity Measures

From the general dimensions of complexity (Liu & Li, 2012), for well-structured educational items, the most relevant complexity dimensions are size, variety, and relationships. Complexity measures are based on the item statement and item solution, which can be used to measure size, variety, and relationships in many ways. For example, in word problems in mathematics, size can be measured as the length of the text, variety as the number of distinct operations that need to be performed to reach the solution, and relationships can quantify the depth of the computation tree (i.e.,

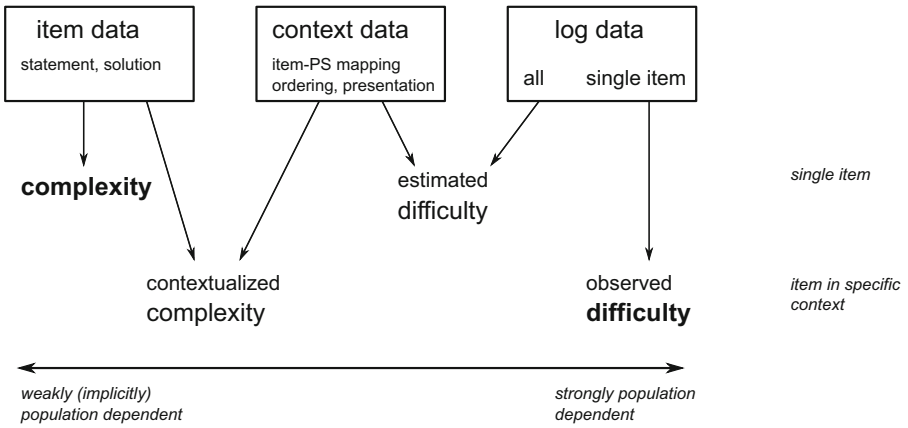


Fig. 1 The range of complexity and difficulty measures and the data that are used to compute these measures

whether the individual steps are independent or dependent on each other). Complexity measures are always specific for a particular domain and type of item. There are, however, several general basic approaches.

A basic complexity measure is the *length* of an item. This can be measured in various ways: Do we take into account item statement, item solution, or both? What unit of measurement do we use (e.g., characters, words, lines of code)? For most items, there is some choice of the length that is natural (although not necessarily useful).

Items in many domains contain text in natural language. It is dominant in language learning but also present in mathematics (word problems) or programming (specification of a problem). For such cases, we may use *text complexity measures*, e.g., using readability measures (Benjamin, 2012).

Another common approach to measuring complexity is to take into account the number of *concepts* needed to solve an item. For example, in the example $2^3 + 3^2 - 2 \times 3$, the concepts may be addition, subtraction, multiplication, and exponentiation. This approach requires clarification of several points:

- How do we choose concepts and their granularity? For example, should addition and subtraction be treated as two separate concepts?
- What do we count? We can use, for example, a unique count ("variability"), a simple sum (taking the number of occurrences of concepts into account), or a weighted sum (taking into account estimated difficulty of concepts).
- Where do we count the concepts? We can detect concepts in item statement (which makes sense for math examples in Table 1), item solution (which makes sense for the programming example in Table 1), or in the item solving process (which makes sense in the case of logic puzzles).

For specific types of items, there are many other potential complexity measures, which can, for example, try to capture concept interaction or the structure of solution paths ("multiplicity, redundancy, and uniqueness" dimensions described by Liu and Li (2012)). We can, of course, also construct aggregated measures, which combine several aspects of complexity.

Difficulty Measures

There are two basic measures for "observed difficulty" that are applicable for nearly all types of items: the *failure rate* and *median response time*. The failure rate expresses the proportion of students who failed to solve the item correctly. It is a simple complement of the success rate, which is sometimes preferred. As a measure of difficulty, the failure rate is more natural as greater difficulty corresponds to a higher failure rate. As response times are typically highly skewed (Van Der Linden, 2009; Pelánek & Jarušek, 2015), the median is a preferable measure of centrality than the mean. Alternatively, it is possible to use the mean of log-transformed times.

For some types of items, there are other simple difficulty measures:

- items allowing multiple attempts: *number of attempts*,
- items with hints: the *hint usage* (e.g., expressed as the proportion of students who take a hint),
- items with solutions of different quality: *solution quality* (e.g., the compactness of code in a programming exercise),
- items with a constructed response: *number of edits* required to construct the answer.

Role of Context

As Fig. 1 illustrates, the basic distinction between complexity and difficulty is convoluted by the role of context: What is the group of students that answer the item? What items (learning materials) have the students seen before answering the item? For a specific illustration of the role of context on complexity and difficulty, Table 2 describes several of our previously used examples in three different contexts. In each case, the third context leads to greater difficulty than the first one.

The basic approach is to consider the role of context as follows:

- Item complexity is objective (absolute), i.e., independent of specific solvers and the context of item use.
- Item difficulty is relative, i.e., concerned with the performance of a particular student population in a given context.

However, this is a simplified view. Each complexity measure is in some way (often implicitly) population dependent. For example, a complexity measure may be based on the number of concepts used in an item. This may seem independent of the student population. But what is a single concept? Do we distinguish addition and subtraction as two separate concepts? That may depend on the student population.

On the other hand, difficulty measures can be "decontextualized." For example, simple difficulty measures like the failure rate are clearly dependent on the particular student population, but item response theory models do produce difficulty estimates that are "group invariant" (Baker, 2001). A similar effect is often achieved by the use of the Elo rating system (Pelánek, 2016), which can be seen as a lightweight variant of IRT models. Other student modeling techniques, e.g., the Additive Factors Model (Cen

Table 2 Illustration of the role of context

Item	Contexts
How many ways can you arrange the letters in 'Rococo'?	1. Word problems on permutations with repetition. 2. Word problems on combinatorics. 3. Items on various part of mathematics.
Does she [play/plays] the guitar?	1. Questions in present simples. 2. Positive sentences, negative sentences, and question in present simple. 3. Tenses in English.
Write a function that for a given `n` prints all its divisors.	1. The problem is solved after a sequence of problems using functions and the modulo operator. 2. The first problem in the application which requires the use of the modulo operator. 3. The first problem that students solve in an interactive web application.

et al., 2006), can also incorporate the information about the subpopulation into estimates (although the model measures difficulty only for concepts, not for items).

As Fig. 1 depicts, we may consider complexity measures that explicitly take the context data into account to produce "contextualized complexity." We may also use the context data to try to "remove the context bias" from the basic difficulty measures and to produce estimated (decontextualized) difficulty measures. Consequently, when we take the context into account, complexity and difficulty measures end up lying on a spectrum rather than at two clear poles.

Taking Use Case into Account

As our literature review and the above-given discussion suggest, the notions of complexity and difficulty are very difficult to capture precisely. It thus does not seem very sensible to try to find *the correct measure* of item difficulty or complexity. Rather, it is meaningful to focus on *useful measures*—measures that will enable us to improve in some way our learning systems. The usefulness of measures depends on the specific goal that we are trying to achieve. There are several use cases of difficulty and complexity measures, and each of them requires different types of measures.

For example, in some applications, we need complexity measures that predict well difficulty. In these cases, we often do not care about the interpretability of a measure; the primary criterion of usefulness is predictive accuracy. In other cases, however, we may be more interested in multiple measures that are interpretable and not necessarily strongly correlated. We highlight the different use cases in the subsequent discussion of measures in specific educational domains.

Item Complexity and Difficulty in Specific Domains

In the previous section, we discussed item complexity and difficulty in general. Now, we consider several specific domains and discuss issues specific to these particular

situations. Our main aim is illustration, not completeness. We try to illustrate how the general principles discussed above are realized in several domains. We also aim to highlight different uses of complexity and difficulty measures. In this section, we thus focus on the breadth of discussion and on covering results reported in previous studies. For the following section, we pick one domain (introductory programming) and provide deeper analysis and novel results.

Reading and Language Learning

Analysis of text complexity and difficulty is relevant not only for applications in language learning (both a first and a second language) but also in many other domains since natural language text is very often part of an item statement, e.g., in word problems in mathematics or as a description of programming tasks. It is also one of the few areas where the distinction between complexity and difficulty has been clearly made in previous research (Mesmer et al., 2012).

Text complexity measures, often called *readability formulas*, have been studied for a long time (Benjamin, 2012). Basic measures are typically linear combinations of shallow text features, e.g., Flesch-Kincaid grade level formula, one of the most well-known, is given as $0.39ASL + 11.8AWL - 15.59$, where ASL is average sentence length and AWL is average word length. Although still widely used, simple measures of this type are subject to frequent critique, e.g., by Bailin and Grafstein (2001). Several more complex approaches to measuring text complexity have been developed, e.g., Coh-metrix (Graesser et al., 2004), TextEvaluator (Sheehan et al., 2014).

Applications

A typical application of text complexity measures is to predict text difficulty and thus to enable the selection and recommendation of texts for reading for a particular student population or individual. The readability formulas are sometimes used as a tool for authors while writing a text and are integrated into many word processors. It has been noted, however, that this practice can be detrimental (Benjamin, 2012).

In second language learning, complexity measures can be used as a tool for the development of grammar questions—they can be used to pre-calibrate the difficulty of new items before student data are available and to provide insight into reasons for item difficulty. For examples of such analysis, see Pandarova et al. (2019), who analyze fill-the-gap grammar exercises, or Ayako Hoshino(2010), who analyze multiple-choice questions. For vocabulary learning, measures are a useful tool for defining homogeneous groups of words for practice. A vocabulary learning application needs to structure thousands of words into manageable groups; it is beneficial for practice when these groups are similar with respect to difficulty and complexity.

Complexity Measures for Individual Words

The complexity of individual words can be measured along several basic dimensions:

– basic word structure: the length of the word, the number of syllables, grapheme-to-phoneme ratio (Rosa & Eskenazi, 2011),

- word familiarity: the frequency of word usage (typically measured as the logarithm of the number of word occurrences in a corpus) or age-of-acquisition (Kuperman et al., 2012),
- how abstract and hard to image the word is: denoted as concreteness, imaginability, or measured using a list of academic vocabulary (Graesser et al., 2004; Sheehan et al., 2014),
- semantic complexity: aspects like polysemy or ambiguity of words, can be measured, for example, as the number of meanings provided in a dictionary or WordNet (Miller, 1998).

In the case of second language learning, there are additional considerations, as the difficulty of words may differ from native speakers, e.g., due to the similarity of words in different languages. Specialized classifications of words have been prepared for this purpose. For the Common European Framework of Reference levels (Milton, 2010), Sohsah et al. (2015) collected expert (teacher) opinions for a sample of words and used machine learning techniques to extend them to a large set of words. Uemura and Ishikawa (2004) presents a similar effort targeted at Asian language learners.

Sentence and Text Complexity Measures

The complexity of sentences or whole texts can be again measured along several basic dimensions:

- typical vocabulary used: averages of measures for single words (e.g., average length, familiarity, concreteness),
- vocabulary richness: the number of unique words, type:token ratio (which measures the number of repetitions of individual words),
- syntactic sentence complexity: the length of sentences, the depth of a parse tree, the number of (syntactically) valid parse trees, the mean number of modifiers, the average number of words before the main verb,
- measures based on part-of-speech tagging: the presence or the count of individual tags, the density of tags (e.g., high density of pronouns may be difficult for comprehension (Graesser et al., 2004)),
- text cohesion (Graesser et al., 2004), which can be measured using vocabulary used (presence of specific connectives or causal verbs), co-reference (repeated words in adjacent sentences), or semantics (similarity of words or sentences based on LSA inference or word embeddings).

These are just the basic directions for measuring text complexity. Researchers have also considered other dimensions, e.g., the degree of narrativity, argumentation, or conversational style. These measures are often based on a combination of the above-described measures. Basic readability measures like Flesch-Kincaid are simple linear combinations of elementary measures. More complex approaches like Coh-metrix (Graesser et al., 2004) or TextEvaluator (Sheehan et al., 2014) report a wide range of measures and perform their grouping based on human expert judgment or by machine learning techniques (e.g., principal component analysis). Sheehan et al. (2008) highlights the genre effects—the values of some complexity measures require different interpretations for informational and literary texts.

A complexity measure should also take into account the task that the student should perform with the text. For example, a common task in language learning is the fill-the-

gap exercise. In this case, the complexity measure can put additional weight on the complexity of the missing word or take into account the size of the context that is necessary for filling in the word.

Difficulty Measures

The difficulty of texts and learning tasks can also be measured in many ways. One option is to focus on reading fluency; in that case, we measure the accuracy and speed of reading. Another option is to focus on reading comprehension or understanding of grammar; this is typically tested by multiple-choice questions or fill-the-gap exercises, and the straightforward difficulty measure is the failure rate. Additionally, difficulty measures can take into account the response time, the number of incorrect attempts, or the number of edits while writing the response.

Amendum et al. (2018) provide a recent meta-analysis of studies with experiments concerning the relation of text complexity and text difficulty for elementary grades, with difficulty measured both as reading fluency and reading comprehension. The report shows that researchers have found a wide range of measures correlation, depending on the used complexity and difficulty measures and on the specific aspects of a study (population, grade, used texts).

An important aspect is whether we measure difficulty with respect to recall or recognition. A constructed response (students have to write the answer) requires an active recall of knowledge. A selected response (students only select the answer from available choices) requires only recognition of an answer. Both of these types of questions are commonly used for language learning tasks, and each of them can lead to quite different estimates of difficulty. Fig. 2 provides a specific illustration. It shows the relation between the failure rate in the recall and recognition exercises for several groups of words. Notice, for example, that "brown" and "yellow" have a similar failure rate as "green" and "black" with respect to recognition but are more difficult with respect to correct recall (in this case, mainly due to spelling mistakes). On the other hand, "stop" is easy to recall and spell correctly, but in the recognition exercise, the failure rate is relatively high due to confusion with "stand." This example illustrates another issue with difficulty measures in language learning: dependence on the native language of students. Our analysis is based on the performance of Czech students; "stop" and "stand" have similarly sounding translations in Czech.

Multiple-Choice Questions

A multiple-choice question (MCQ) is probably the most commonly used type of item used in educational systems. The use of MCQs is widespread, particularly in testing, but MCQs are also useful for learning and practice (Gierl et al., 2017). From the perspective of complexity analysis, an advantage of MCQs is their clear item anatomy:

- a stem, i.e., the introductory text stating a question or a problem (*What is the capital city of France?*),
- the correct answer (*Paris*),
- distractors, i.e., incorrect options (*Berlin*, *London*).

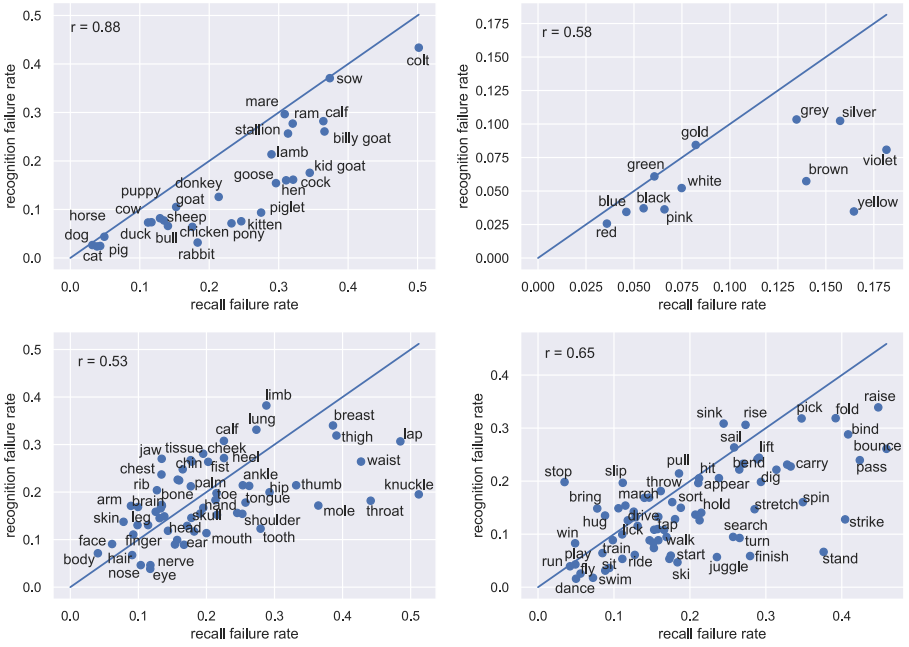


Fig. 2 The relation between two measures of difficulty for four vocabulary groups (the data are from the `umimeanglicky.cz` system; English as a second language; Czech students)

Applications

The typical application of complexity and difficulty measures in the case of MCQs is the prediction of difficulty based on complexity measures, e.g., in the context of medical questions (Yaneva et al., 2019; Baldwin et al., 2020) or English vocabulary learning (Susanti et al., 2016).

This prediction is useful particularly for automatic question generation (Kurdi et al., 2020) since it allows the generator to create questions of suitable difficulty. The prediction is also useful for manually created questions—in adaptive systems, it can eliminate (or shorten) the need for calibration of questions (Pandarova et al., 2019). Other applications, such as feedback for content authors, are also useful.

Measures

The basic complexity measures for MCQs are based on the readability measures discussed above. The question stem is typically textual, and thus we can apply the general text measures and linguistic features (Yaneva et al., 2019; Baldwin et al., 2020). In addition to these general measures, it is useful to focus on the concepts mentioned in the text. For example, the question "What is the capital city of Angola?" has the same sentence structure as the question about France, and thus the basic text measures cannot capture the difference between these two questions. To do so, we can use, for example, techniques based on word frequency (Benedetto et al., 2020) or word embeddings (Mikolov et al., 2013).

A specific feature of the multiple-choice question format is the use of distractors (Gierl et al., 2017; Mitkov et al., 2009). From the perspective of complexity, a key aspect of distractors is their similarity to the correct answer since more similar distractors require more complex processing from students (Ascalon et al., 2007). The similarity can be measured in several ways, for example:

- semantic similarity based on knowledge measured using ontologies (Leo et al., 2019; Papasalouros et al., 2008; Lin et al., 2015; Nuthong & Witosurapot, 2017) or knowledge graphs (Seyler et al., 2017), e.g., using distance or depth within an ontology,
- semantic similarity based on language features and text processing measured using WordNet (Mitkov et al., 2009), word embeddings (Mikolov et al., 2013), or latent semantic analysis,
- syntactic similarity measured using edit distance (Ayako Hoshino, 2010), phonetic similarity (Mitkov et al., 2009), and potentially extended with part-of-speech tags and other word properties,
- the visual similarity in the case of images (e.g., animal pictures).

For difficulty measures, the most commonly used one is the basic failure rate. As MCQs are often used in adaptive systems, it is useful to consider also difficulty parameter of IRT models, which are able to take into account biases caused by adaptivity. Response time, another common difficulty measure, is typically of secondary importance in the case of MCQs.

Word Problems in Mathematics

In the domain of mathematics, a specific interesting area is word problems. Analysis of word problems is very challenging due to the complex interaction of reading skills and computation skills. Daroczy et al. (2015) provide an extensive overview of linguistics and numerical factors of word problem complexity and their relation to difficulty. Barbu and Beal (2010) discuss specifically the impact of linguistic complexity of word problems on performance for second language English learners. The aspects influencing the difficulty are often non-intuitive, even for experienced teachers. Koedinger and Nathan (2004) compared the difficulty of word problems and mathematically equivalent equations to evaluate the impact of problem representation on difficulty. They note that although word problems were easier than equations, most teachers had the opposite expectation.

Applications

In this case, the main type of application of complexity and difficulty measures is "insight." The analysis can provide information about aspects of word problems that influence difficulty—such information is useful for content authors as a guideline for creating new problems. It can also be useful for the automatic generation of word problems (Polozov et al., 2015), where it can provide methods for controlling the difficulty of generated problems (Khodeir et al., 2018; Wang & Su, 2016). The insight is also useful for the management of content and meta-data, e.g., specifying the

mapping of items to knowledge components or removing unsuitable items. Beyond the development of learning systems, insight into word problems is also relevant for cognitive science and pedagogy research (Daroczy et al., 2015; Koedinger & Nathan, 2004).

Measures

The complexity measures for word problems can be divided into two basic types. At first, there are measures that can be relatively simply computed algorithmically, but these are typically "shallow," covering only the surface features of word problems. These include basic properties of a word problem text (e.g., the length of a text, the number of words, the average length of a sentence, readability formulas) and basic analysis of numbers used in the problem statement (e.g., the total count of numbers, the magnitude of numbers, the type of numbers: small integer, decimal, fraction).

To better capture the complexity of word problems, it is necessary to consider also "deep" measures that correspond better to cognitive processes used by solvers. Such measures can be based on the number and type of concepts used to compute the solution. We may also take into account the necessity of the use of "common knowledge" in the computation (e.g., to find a solution, a solver may need to know the number of days in a week or the number of legs of a spider) or the presence of irrelevant data. These deep measures are, however, very difficult to compute algorithmically just from the text of the word problem.

To evaluate algorithmically computed text features, current studies typically either use manual labeling to obtain the deep features or consider only a set of closely related problems (e.g., addition and subtraction under 20). We have explored the relations among complexity and difficulty measures for a diverse set of 1000 word problems from the system `umimematiku.cz`. The analysis showed that the shallow, easily computable complexity measures correlate only very weakly with difficulty. The only larger correlation is between the length of the text and the response time—which is an expected and not very applicable result.

Logic Puzzles

Logic puzzles are an interesting bridge between educational applications, cognitive science research, and computer games. Complexity and difficulty are key aspects in the design of computer games (Aponte et al., 2011). For example, Linehan et al. (2014) analyzed the complexity of four successful puzzle games; they measure complexity as a number of steps and discuss the relation of this "objective" complexity and difficulty as performance. They also discuss the relation of these measures to sequencing and the role of the scaffolding of learning.

The domain of logic puzzles is very wide, see, e.g., the taxonomy proposed by Hufkens and Browne (2019). For the sake of concreteness, we consider in our discussion mainly transport puzzles (e.g., Sokoban, Rush hour) and constraint satisfaction puzzles (sometimes also called "Japanese logic puzzles," the most well-known example being Sudoku).

Applications

Puzzles have been used in basic cognitive science research, e.g., Kotovsky et al. (1985) analyzed several izomorfs of the famous Tower of Hanoi puzzle, showing that some of them are 16 times more difficult than others; this research highlights the impact of problem representation on difficulty.

The main practical application of measures is to predict difficulty based on complexity measures. Such a prediction is useful for the sequencing of puzzles or presentation of puzzles to users. A typical example is the indication of Sudoku difficulty in newspapers. A specific application area where the notions of difficulty and complexity are critical is procedural content generation (Togelius et al., 2011). Estimation of difficulty is an important step in the automatic creation of game content (evaluating and filtering candidate puzzles).

Measures

The basic complexity measures for puzzles are the size of the problem (e.g., the size of the grid, the number of elements in the puzzle) and the length of a solution, i.e., the number of steps necessary for solving the puzzle.

Many puzzles (particularly transport puzzles) can be expressed as a search within a state space of puzzle configuration. Properties of this state space (e.g., the structure of paths, redundancy, bottlenecks) can be used as complexity measures; additionally, it is possible to build a computational model that traverses the state space and simulates human behavior (Jarušek & Pelánek, 2011).

Primi (2001) analyzed Raven's Progressive Matrices (often used in "IQ tests"), identifying the perceptual organization and the amount of information (a measure combining the number of elements and the number of rules) as complexity measures most predictive of difficulty.

Another approach is based on constraint relaxation (Pelánek, 2014), which is based on the observation that difficult puzzles often have "close-but-not-valid" solutions. We can thus formulate complexity measures based on the increase in the number of solutions (or decrease in the shortest path length) when some constraint in the puzzle is relaxed (we allow one repeated number in a Sudoku row or one step through an obstacle in Sokoban).

Concerning difficulty measures, a specific aspect of puzzles is that it is mostly not meaningful to talk about "incorrect answers." Rather, some solution attempts are "not finished yet." We can measure difficulty by unfinished rate, but this measure is often not very informative, e.g., it does not distinguish between easy and extremely easy puzzles (they all have the unfinished rate close to zero). In the case of puzzles, the primary focus of difficulty measures is thus problem-solving time.

For an illustration of specific results, Table 3 provides a summary of the results reported by Jarušek and Pelánek (2011) and Pelánek (2014). The table shows the correlation of various complexity measures with median problem-solving time (a basic difficulty measure for logic puzzles). The results show that simple complexity measures (size, length of solution) correlate mostly only weakly with difficulty. However, for some puzzles (Rush hour), even such simple measures can provide a good prediction of difficulty. Where basic complexity measures provide weak correlation with difficulty, it

Table 3 Correlations between complexity measures and difficulty measured as the median problem-solving time for several puzzles

Puzzle	Complexity measure	r
Sudoku	number of givens	0.25
	constraint relaxation	0.56
	linear combination of measures	0.84
Sokoban	path length	0.19
	computational model	0.39
Rush Hour	path length	0.77

is often possible to design complexity measures that lead to high correlation; however, these measures are often quite puzzle-specific and non-trivial even for puzzles with simple rules (as is the case of Sudoku).

Complexity and Difficulty in Introductory Programming

In the previous section, we focused on the breadth of the discussion. Now, we demonstrate a deeper analysis of difficulty and complexity measures for one educational domain: introductory programming.

We use data from five exercises summarized in Table 4. The first three exercises use a block-based programming interface (Bau et al., 2017), and the other two use Python programming. Some items include scaffolding in the form of an initial partial program, see Fig. 3. All solutions have between 2 and 24 lines (or blocks corresponding to lines), with medians ranging from 5 lines in the Spaceship exercise to the 9.5 lines in the Shapes exercise.

Applications

The goal of our analysis is to explore the relationship between various difficulty and complexity measures and how they are impacted by scaffolding and data collection

Table 4 Data used for the analysis of programming exercises. The *scaffolding* denotes how many items provide an initial partial program to complete. Attempts include both successful and unsuccessful sessions. The median time is computed from the successful attempts only

Exercise	Levels	Items	Scaffolding	Attempts	Median time
Spaceship	9	85	0%	135,000	50 sec
Shapes	7	54	44%	34,000	82 sec
Turtle Blockly	8	77	60%	117,000	81 sec
Turtle Python	5	46	89%	34,000	92 sec
Python	9	73	100%	21,000	180 sec

initial program:

```
right(15)
forward(100)
```

solution:

```
for i in range(4):
  right(15)
  forward(100)
  right(150)
  forward(100)
  left(75)
```

Fig. 3 Compass – example of an item from Turtle Python exercise

biases. In particular, our goal is not to find the single *best* difficulty or complexity measure. It is worth reiterating that there is no such thing as the best difficulty or complexity measure—the suitable choice depends on the intended application.

For instance, Alvarez and Scott (2010); Sheard et al. (2013) considered the use of complexity measures for estimating the difficulty of programming assignments and exam questions. For such use, we seek a complexity measure that highly correlates with a difficulty measure. Which difficulty measure to use for evaluation depends on which aspect of student performance we care about; the failure rate is insufficient if we are concerned about the speed of coding and quality of the resulting code. We might care about multiple aspects of performance, in which case we would use multiple difficulty measures for evaluation. When reporting results, clearly distinguishing between the complexity and difficulty measures would be crucial since only the complexity measures can be used in the real situation—no performance data are available at the time of difficulty estimation.

Now consider a different use case: monitoring the quality of already deployed items. A mismatch between the complexity and difficulty can help us to detect misbehaving items; for example, an item intended to practice loops that many students solve with just a sequence of commands. Note that if such mismatch occurred in the previous case (difficulty estimation), we would try to improve the complexity measure, even at the cost of lower interpretability. In contrast, interpretability is helpful when diagnosing issues with items, and the reasonable action might well be updating the item rather than trying to fix the complexity measure.

Measures

In programming exercises, complexity can be measured using either the problem statement or solution. Using the problem statement, we can measure its length, linguistic complexity, and references to external domains (Sheard et al., 2013). Using the solution, we can measure its length, the number of programming concepts, the number of flow-of-control structures (Alvarez & Scott, 2010), cyclomatic complexity (Whalley & Kasto, 2014), and Halstead complexity measures based on the number of operators and operands (Ihantola & Petersen, 2019).

The complexity can be influenced by other aspects, such as background story, scaffolding (e.g., an initial partial program to complete), and the context in which the item is solved. The background story seems not to significantly impact the difficulty of introductory programming problems (Bouvier et al., 2016; Craig et al., 2017). In contrast, a problem-solving context, such as previously practiced problems, can alter Bloom's level of intellectual complexity (Thompson et al., 2008; Gluga et al., 2012).

In the analysis that follows, we explore complexity measures that are based on the number of concepts since these are universally applicable across all programming

exercises. We either search for predefined keywords or extract the concepts by traversing the abstract syntax tree; the latter option allows us to detect some programming patterns such as nested loops and recursion. We compare several ways of aggregating the numbers of occurrences of individual concepts into a single number, and we also explore the impact of incorporating scaffolding into the concept-based complexity measures.

We compare these complexity measures to three simple difficulty measures: the failure rate, median response time, and the median number of attempts. As the attempts we typically count all executions, except for the Python exercise, where only evaluations on hidden test cases are recorded. In addition, we explore two partially decontextualized estimates of difficulty using item difficulty parameters of IRT and Elo models adapted to predict problem-solving times (Pelánek & Jarušek, 2015; Pelánek, 2016).

Concepts

One way to measure complexity in introductory programming is to consider all the concepts needed to solve the item. To define a specific measure for a given exercise, we need to decide what the relevant concepts are and how to aggregate them into a single number.

Concept Extraction

Concepts can be extracted either from the item statement or the solution. The solution, being a program, is much more amenable to automatic processing. Regardless of whether the students use a block-based or a text-based programming language, the program can be transformed into an abstract syntax tree (AST), allowing for concept extraction techniques that are independent of the specific exercise. The item statements, in contrast, are more diverse; they can include a grid world (Spaceship exercise), a picture to draw (Shapes and Turtle exercises), or a text statement in a natural language, together with some test cases (Python exercise). Moreover, the involved concepts are usually more apparent from the solution than from the statement.

A straightforward approach to extract concepts from the solution is to search for a set of keywords, such as `for` and `left`. Alternatively, the concepts can be extracted by traversing the abstract syntax tree of the solution. As the nodes of the tree typically correspond to reasonable concepts, this approach can be applied to any introductory programming exercise, without the need to specify keywords to search for. It may be useful to look not only at the node types but also at their content; for example, to distinguish calls of different functions, to find local variables, and possibly to extract individual literal values, such as angles in the Turtle exercises. Looking at the structure of the abstract syntax tree, we can even detect programming patterns such as nested loops and recursion.

Granularity of Concepts

Instead of attributing each keyword or AST node type to a distinct concept, we may define some meaningful groups. It seems, for instance, plausible to consider all comparison operators as a single concept. The granularity decision is not just a binary

one; the concepts form a hierarchy. For example, `left(75)` command could be considered as a fine-grained concept or as an instance of one of the following broader concepts: left, rotate, turtle movement, turtle commands, or, most broadly, any commands. We need to decide which level of detail to use.

The difference between various granularities has an especially marked effect once we use specific values in AST nodes. Since considering each literal value as a distinct concept would result in dozens of concepts, most of which are not much meaningful, we may wish to group all angles, all numbers, or even all literals as a single concept.

As an example of various granularity levels, consider the Compass item (Fig. 3). Distinguishing individual commands and literals, we may obtain up to 9 fine-grained concepts: `for`, `forward`, `left`, `right`, `int4`, `length100`, `angle15`, `angle75`, `angle150`. By grouping similar commands and types of literals, we end up with six medium-grained concepts: `for`, `move`, `rotate`, `int`, `lengths`, `angles`. We may even decide to use just three coarse-grained concepts: `loops`, `commands`, `literals`.

It is unclear which granularity is useful to distinguish. Partially, it depends on the expected population of users; distinguishing between various arithmetic operators may be important for the primary school students, but probably not so much for high school students. Furthermore, with higher granularity, there is a trade-off between interpretability and the predictive power, so the purpose of the complexity measure should be taken into account as well.

Aggregation of Concepts

Having found the counts of individual concepts, we now need to aggregate them into a single number. Two basic approaches are counting the total number of occurrences of all concepts *(counts aggregation)* and counting just the number of unique concepts *(binary aggregation)*. In the case of binary aggregation, the contribution of each involved concept is the same, irrespective of how many times it appears in the solution. Between these two extremes, there are many ways to account for the number of occurrences while avoiding the excessive contribution of a single concept, for example, by logarithmically scaling the counts *(log-counts aggregation)*.

To illustrate these three aggregations, consider the Compass item (Fig. 3) again. Assume that we extract just keywords (no literals), group all turtle commands, and obtain two coarse-grained concepts: `for` (1×) and `commands` (5×). The binary aggregation results in complexity 2, while the counts aggregation to complexity 6. For log-counts aggregation, we transform the counts by $\ln(1+n)$ (to ensure nonzero values) and obtain complexity $\ln(1+1)+\ln(1+5) \approx 0.7+1.8=2.5$.

The difficulty of different concepts is not the same; for instance, the forward command is an easier concept than for-loop. To account for the difficulty of individual concepts, we can assign them weights before aggregating their contribution. The weights could be either set manually by the author of the items or estimated from the available data. For example, in the case that we have already collected performance data on many other items from the exercise, we can fit the weights to maximize the predictive performance of the complexity measure (Benedetto et al., 2020; Yaneva et al., 2019).

An alternative approach to set the concept weights, which does not rely on historical performance data, is to assign lower weights to the concepts that appear frequently, leading to a statistic known as *inverse document frequency* (IDF) in the context of ranking relevant documents for a given term (Robertson, 2004). Treating concepts as terms and items as documents, we obtain the following formula for a smoothed IDF: $\ln(N/n) + 1$, where N is the total number of items in the exercise, n is the number of items involving the concept, and the $+1$ term ensures that all weights are strictly positive.

Let us illustrate the IDF on the Compass item. The for-loop is used in 36 out of 46 items, so IDF of `for` is $\ln(46/36) + 1 \approx 1.25$. Turtle commands are used in all items, so IDF of `commands` is $\ln(46/46) + 1 = 1$. The weights can be combined with any of the three aggregations; the only change is that the sums are now weighted. For example, the IDF-weighted log-counts complexity is $\mathrm{IDF}(\text{for}) \cdot \ln(1 + n(\mathit{for})) + \mathrm{IDF}(\text{commands}) \cdot \ln(1 + n(\mathit{commands})) \approx 2.7$.

Analysis

We compared five concept extraction approaches described in Section 5.3: (1) keywords-based, (2) node types in the AST, (3) AST-based concepts utilizing node content but excluding literals, (4) AST-based concepts utilizing node content and including literals, and (5) AST-based concepts including literals and two programming patterns: nested loops and recursion. We aggregated the concepts using log-normalized counts weighted by IDF and measured Spearman's rank correlation coefficient with the median solving time. The results are shown in Fig. 4.

Utilizing AST leads to a higher correlation than keywords-based extraction across all five exercises, even though the basic AST approach is more automatic in the sense that it does not require specification of keywords for the given exercise. Even among the AST-based approaches, there are sometimes significant differences. Whether to include literals depends on the exercise: if figuring out the numbers requires some math (as is the case for lengths and angles in the Shapes and Turtle exercises), then including the literals is likely to improve the correlation.

To explore the granularity, we used the full set of AST-based concepts, including literals and programming patterns, and manually defined a hierarchical clustering of the concepts into seven levels in each exercise. The lowest granularity levels contain about four concepts (e.g., commands, control flow, variables, math), while the highest granularity levels have tens of concepts (individual functions, operators, and even

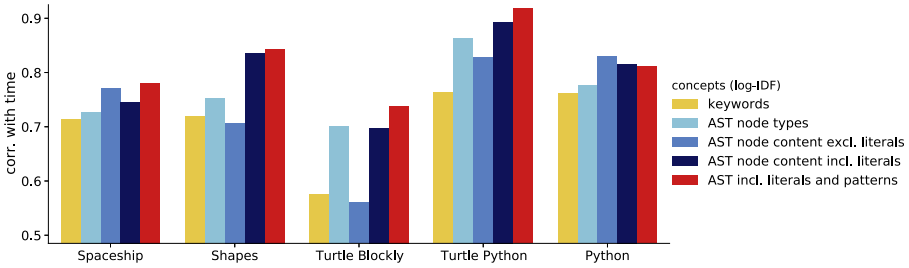


Fig. 4 Impact of used concepts on the Spearman correlation with median solving time, using log-normalized counts weighted by IDF as a complexity measure

literals). To create the hierarchy, we started with the most granular concepts and gradually grouped those that we perceived as the most similar, e.g., `left` and `right` as `turn`, then `turn`, `forward`, and `backward` as `move`, etc.

For each granularity, we computed six aggregated complexity measures, using three options for normalizing the concept counts (binary, log-normalized, untransformed counts) and optionally weighting the concepts by IDF. Finally, for each granularity and aggregation, we measured the Spearman's rank correlation coefficient with the median solving time. The results are shown in Fig. 5.

The granularity seems to be a crucial choice, especially for the binary aggregation. In our case, the highest correlations with the time difficulty were achieved at the highest granularities. Whether the increase in the correlation is worth the increased number of concepts and worse interpretation of the complexity measure depends on the intended application.

The type of aggregation can be a crucial choice as well, but the differences between the aggregations tend to get smaller if the granularity is sufficiently high. Across examined exercises, the log-normalized counts with IDF weights usually achieve either best or close to best correlations with the time difficulty. We have also measured correlation with the failure rate; the log-IDF measure is also among the best in this case, although it is usually slightly outperformed by the binary-IDF measure.

Relations Among Measures

There are many complexity and difficulty measures, but it is not obvious how they relate to each other and which of them to use for decision making. Some measure similar aspects and are thus redundant; others add new information to the mix.

Measure Similarity

The similarity of individual measures is illustrated in Fig. 6. This figure shows Pearson correlation coefficients for every pair of measures across our exercises. The first six measures represent different aspects of difficulty, and the last three represent complexity. Concepts coarse, medium, and fine are log-counts aggregated AST-based concepts, including literals and programming patterns of granularity 1, 4, and 7 (as used in Section 5.3).

We identified four general emerging patterns. (1) Difficulty measures correlate more with other difficulty measures than with complexity measures and vice versa. (2) There

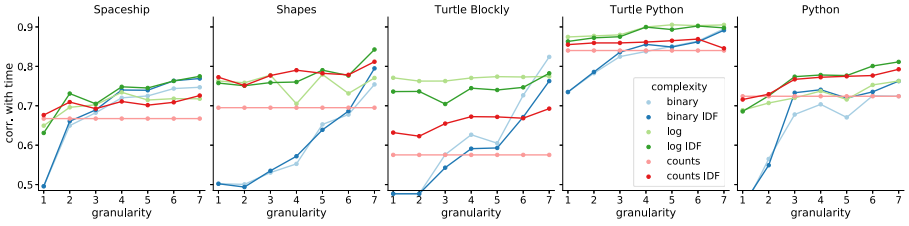


Fig. 5 Impact of concept granularity in the complexity measures on the Spearman correlation with median solving time. Each line corresponds to one possible aggregation of the concepts into a single number

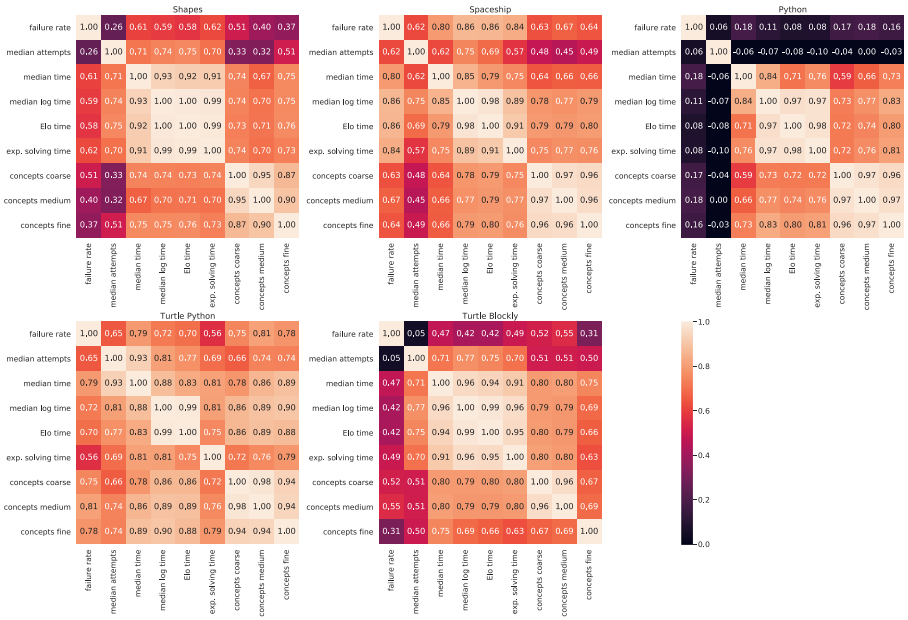


Fig. 6 Pearson's correlation coefficients of difficulty and complexity measures for programming exercises

are differences in correlations within difficulty and complexity groups of measures across exercises. (3) Failure rate and median attempts tend to correlate less with other measures. One possible explanation for low correlations of failure rate is the effect of biases that are discussed later in Section 5.6. (4) Measures median log time and Elo time correlate almost perfectly in all exercises.

Canonical Correlation Analysis

Fig. 6 showed that difficulty measures and complexity measures form two groups that do not correlate as well with each other. However, for a task like problem sequencing, it is desirable to estimate the difficulty of a new problem solely based on complexity. To improve the correlations, we turn to Canonical Correlation Analysis (CCA) (Hotelling, 1936; Thompson, 1984).

CCA is a statistical method for analyzing correlations between two or more sets of random variables. It does so by finding linear combinations of variables in each set, creating canonical components. The coefficients in linear combinations are chosen in such a way that the correlation between canonical components is maximized.

CCA can be naturally used to study the relationship between difficulty and complexity measures. It can be used to study two questions: 1) Which aspects of student performance (difficulty) can be predicted based on the properties of item statements (complexity)? 2) How should we sequence problems? The first question is answered by analyzing correlations (referred to as loadings) of individual measures with the corresponding canonical component. Difficulty measures with higher loadings are better explained by complexity measures. The second question is answered by looking at canonical components that provide a compromise between difficulty and complexity. The CCA computation automatically filters measures that are "reasonable." If we were

to use some unrelated measure of problem complexity (difficulty), it would not have any relation to student performance (problem properties), and thus it would have a small loading.

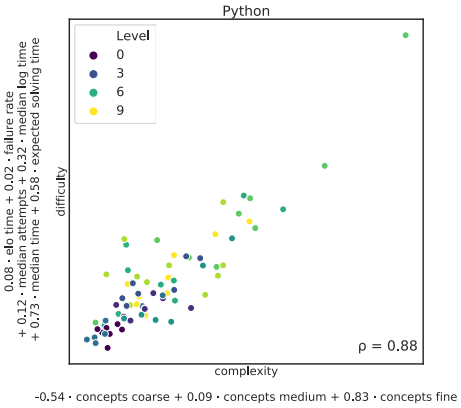
Python Exercise Example

We now illustrate the computation of CCA on the Python exercise. The two sets of variables are difficulty measures (failure rate, median attempts, median time, median log time, Elo time, expected solving time) and complexity measures (concepts coarse, concepts medium, concepts fine). The exact linear combinations of measures for each canonical component are specified in Fig. 7 next to each axis.

Fig. 7 shows a scatter plot of combined difficulty and complexity measures, together with the Pearson correlation coefficient. Each dot represents a single problem, and its color represents the level it belongs to. It shows a reasonable correlation and suggests the feasibility of estimating difficulty based on complexity. Fig. 7 then shows correlations of individual measures with the canonical component. For Python exercise, median log time has a very good correlation with difficulty, and it will be best estimated using complexity measures. On the other hand, the median attempts measure does not correlate at all, which suggests it is not a useful measure for this exercise.

Results for All Exercises

We applied the same analysis to all programming exercises; the results are in Fig. 8 and Table 5. Correlations between *difficulty* and *complexity* is very good in all five exercises. The relation between them is mostly linear and subsequent levels contain both more complex and difficult problems. The only exceptions to this are Spaceship exercise, where *difficulty* is not increasing as much with *complexity*, and Python exercise, where the level focused on drawing with text is significantly more difficult and complex. There are visible gaps between problems in Spaceship and Turtle Python



measure	loading
failure rate	0.16
median attempts	-0.01
median time	0.93
median log time	0.98
Elo time	0.91
expected solving time	0.94
concepts coarse	0.80
concepts medium	0.87
concepts fine	0.94

Fig. 7 Left: Scatter plot of *difficulty* and *complexity* canonical components for Python exercise. Next to each component, there is its definition in the form of a linear combination of measures. The ρ in the bottom right corner is the Pearson correlation coefficient. Right: Correlations (i.e., loadings) of individual difficulty and complexity measures with their corresponding canonical component

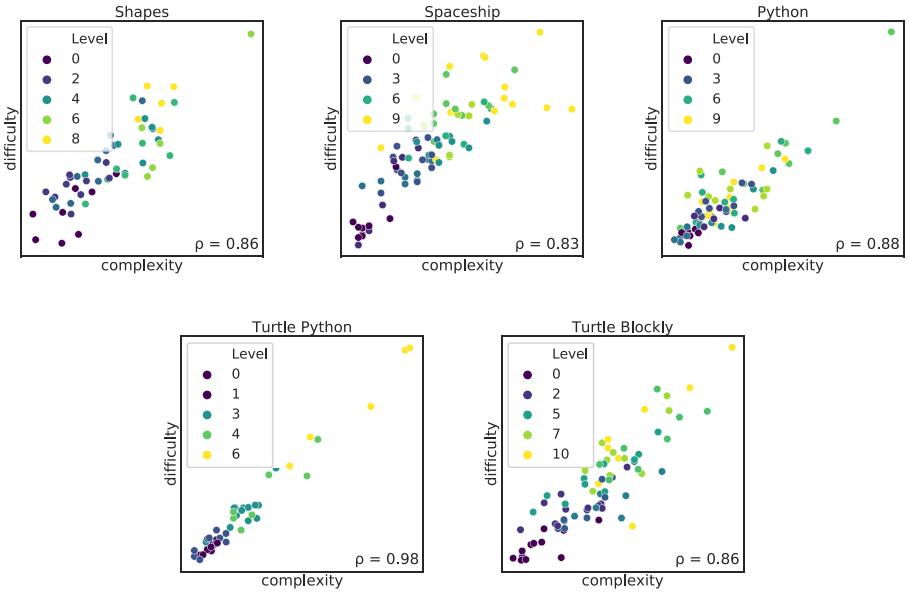


Fig. 8 Measures for programming problems: CCA projection

exercises that could be filled with new, appropriately complex problems to make the transition between levels smoother.

Table 5 allows a deeper inspection of individual measure importance. Correlations of failure rate vary due to biases that are discussed later in 5.6. Median attempts measure is inconsistent between exercises as well. This time because of implementation details and what is considered an attempt in different exercises. Median time and median log time have very good correlations across exercises. Median log time is better suited for the Spaceship exercise, where solving times increase quickly for the last problems. Conversely, median time is better suited for the Turtle Python exercise,

Table 5 Correlations of individual difficulty and complexity measures with their corresponding canonical component across all exercises

Measure	Shapes	Spaceship	Python	Turtle P.	Turtle B.	Avg.
failure rate	0.58	0.78	0.16	0.85	0.59	0.59
median attempts	0.51	0.60	− 0.01	0.84	0.64	0.52
median time	0.93	0.80	0.93	0.98	0.98	0.92
median log time	0.91	0.96	0.98	0.91	0.95	0.94
Elo time	0.90	0.97	0.91	0.88	0.95	0.92
expected solving time	0.89	0.93	0.94	0.81	0.94	0.90
difficulty avg.	0.79	0.84	0.65	0.88	0.84	
concepts coarse	0.97	0.99	0.80	0.87	0.96	0.92
concepts medium	0.91	0.97	0.87	0.93	0.97	0.93
concepts fine	0.94	0.99	0.94	0.95	0.83	0.93
complexity avg.	0.94	0.98	0.87	0.92	0.92	

where solving times grow linearly with increasing complexity. Elo time and expected solving time also behave similarly, and their correlations are negatively affected by outliers seen in Fig. 8.

Differences in complexity measures are much less pronounced than in difficulty measures. All granularities of concepts correlate very well. Finer concepts are more important for text-based exercises (Python and Turtle Python exercise). For block-based exercises, even coarse concepts are sufficient predictors of problem complexity.

Scaffoldings

Programming problems are notoriously known to be too difficult for students due to the high cognitive load they impose. One strategy to decrease the intrinsic cognitive load of programming problems is to provide scaffolding in the form of a partial program to complete (Van Merrienboer & Krammer, 1990; Kelleher & Hnin, 2019). Students can examine and run the provided code, and they need to modify or extend it to solve the problem. By reducing the intrinsic cognitive load, such scaffolding can considerably impact the difficulty of the problem. How to incorporate the information about the scaffolding into a complexity measure is, however, unclear.

Four out of the five examined exercises use scaffolding in the form of an initial partial program. Only some items have such scaffolding, and this proportion varies across exercises; specifically, these proportions in the Shapes, Turtle Blockly, Turtle Python, and Python exercise are 44%, 60%, 89%, and 100%, respectively. The extent of the scaffolding in the individual items varies as well; from just a single command or a function header, to a whole function to use, to almost a correct solution that requires just a slight modification, basically serving as a worked example. The median number of lines (or Blockly blocks that correspond to a new line) in the provided initial program is 2.5 or 3 in all four exercises.

Scaffolding-Aware Measures

One simple way to account for the scaffolding in a complexity measure is to reduce the weight of the concepts that appear in the scaffolding. We have used the following weights, parametrized by a *scaffolding penalty* α. The higher the penalty, the more discounted the scaffolded concepts are; setting $\alpha = 0$ corresponds to the complexity measures that do not account for the scaffolding while $\alpha = 1$ results in no contribution of the scaffolded concepts to the complexity.

For binary aggregation, the non-scaffolded concepts have weight 1, while the scaffolded ones $1-\alpha$. For the counts aggregation, the weights are $n_{sol}-\alpha n_{sc}$, where n_{sol} is the number of occurrences of the concept in the solution, and n_{sc} is the number of occurrences in the scaffolding. Similarly, the log-counts aggregation use weights $\ln(1 + n_{sol}-\alpha n_{sc})$. For example, the scaffolding-aware log-counts complexity of the Compass item (Fig. 3), using $\alpha = 0.5$, is $\ln(1 + n_{sol}(for)-\alpha n_{sc}(for)) + \ln(1 + n_{sol}(commands)-\alpha n_{sc}(commands)) \approx 2.3$.

The accounting for the scaffolding interacts with the choice of the concepts, including their granularity and weights. In the Turtle exercises, for instance, scaffolding is often used to free the students from guessing the size of the drawing or an atypical angle. If the scaffolding in the Compass item were `right(90)` instead of `right(15)`, it would not be much helpful. It suggests that accounting for the

scaffolding could be more effective when used with sufficiently granular and weighted concepts.

Results

Fig. 9 shows that even such a simple accounting for the scaffolding can be helpful. Across all exercises, reducing the weight of the scaffolded concepts increased the Spearman correlation with the median solving time. The concepts in the scaffolding should not, however, be disregarded completely: for $\alpha = 1$, the correlation is in most cases lower than when the information about scaffolding is unused ($\alpha = 0$). In the examined exercises, α between 0.25 and 0.5 typically results in the highest correlation.

While this straightforward accounting for the scaffolding increased the correlation with median solving time, it decreased the correlation with the failure rate. This effect might be partially attributed to the following bias: first items in each level usually provide an extensive scaffolding, so they appear simple when scaffolding is accounted for; at the same time, these items have a relatively high failure rate. As they introduce new concepts, they might be difficult to solve even with scaffolding. Moreover, they are often visited by users who just briefly explore the content and do not seriously attempt to solve it.

Biases

Biases are the inherent product of imperfect, i.e., not uniformly random, data collection. Anything from structured content to personalized interactions, while useful traits from the pedagogical point, can cause biases in collected data. The data being collected simply does not cover the whole reality, but only part of it. This missing information is the reason why it is hard to combat and compensate for biases. It is worth noting that biases only affect difficulty; complexity, independent of performance data, is not affected.

In this section, we focus only on the ordering bias (Čechák & Pelánek, 2019). There are also other biases; these include attrition (Eagle & Barnes, 2014), mastery (Nixon et al., 2013; Murray et al., 2013; Pelánek, 2018), and self-selection biases (Brooks et al., 2015).

Ordering bias occurs when students are solving problems in a similar order. The causes are typically fixed problem order in the system or students being incentivized to solve problems in a predefined order. Under these conditions, the observed difficulty of

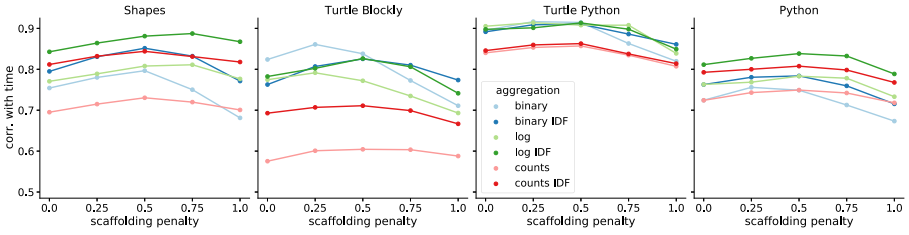


Fig. 9 Impact of accounting for scaffolding in the complexity measures on the Spearman correlation with median solving time. Each line corresponds to one possible aggregation of the concepts into a single number. The concepts that occur in scaffolding are discounted by the *scaffolding penalty*

initial program:
```
def multiples(n):
    print("Two times", n)
```

solution:
```
def multiples(n):
    print("Two times", 2*n)
    print("Three times", 3*n)
    print("Ten times", 10*n)
```

initial program:
```
def nonzero_product(numbers):
    return 0
```

solution:
```
def nonzero_product(numbers):
    product = 1
    for n in numbers:
        if n != 0:
            product *= n
    return product
```

Fig. 10 Initial program (scaffolding) and the author's solution to Multiples and Nonzero product problems from Python exercise

problems at the beginning of the solving sequence is overestimated, and the difficulty of problems later in the solving sequence is underestimated. This is caused by at least two underlying effects: student learning and student attrition.

To illustrate this point, we compare two problems on opposite ends of ordering in Python exercise, *Multiples* and *Nonzero product*. Initial scaffolding and example solution codes of both problems are in Fig. 10. Arguably, *Nonzero product* should be more difficult than *Multiples* as it requires knowledge of multiple programming constructs and the use of the idea of accumulating intermediate results. However, both problems have similar median solving times of 144 and 140 seconds for *Multiples* and *Nonzero product* respectively. The majority of students visited *Multiples* as their first problem, while the majority of students visiting *Nonzero product* have already seen at least 17 other problems. Such difference in student population in terms of their prior experience is what causes observed difficulty to be similar even for problems with arguably dissimilar true difficulty.

We present one more example of ordering bias found in the Python exercise. The problems in this exercise are grouped into levels and presented visually as such. This creates multiple natural entry points at the start of each level for incoming users. Some of these users are just exploring the system, trying out a few problems, and not solving problems seriously. Non-serious attempts artificially increase the failure rate despite difficulty being comparable to difficulties at the given level. Non-serious attempts are not distributed evenly across problems; they tend to occur mainly at the beginnings of levels. Hence, the failure rate of problems is biased based on the order in the level.

Fig 11 (left) demonstrates that problems at the same level are solved by students with widely different degrees of experience in the system. It shows distributions of solving sequence ordering, i.e., how many problems a student has visited, for the fifth level of Python exercise. Most students visiting the first problem at the level have very

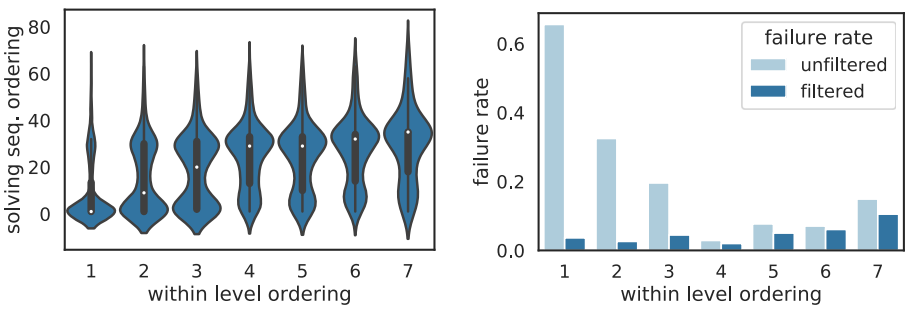


Fig. 11 Left: Violin plot showing the distribution of problem orders in individual solving sequences. It shows only problems from a single level of Python exercise. The shape and width of the violin are given by kernel density estimation of the underlying distribution of solving sequence ordering. Inside each violin, there is also a box plot with a white dot marking the median. Right: Bar chart showing unfiltered and filtered failure rates for the same problems used in the left plot

little or no prior experience in the system. In contrast, students have typically visited at least 30 other problems prior to visiting the last four problems of the level.

Fig 11 (right) shows the largest change in the failure rate for the first problem in the level when non-serious attempts are accounted for. It shows differences in failure rates of problems in the fifth level of Python exercise after filtering. We filtered out attempts of students with less than three visited problems in total and kept only students with at least three visited problems. The figure shows a drop in failure rates after filtering, especially for the first three problems of the level that are most affected by ordering bias. To further prove the reasonability of filtering, the filtered failure rate for Python exercise has higher correlations with concepts by 0.3–0.4.

Note that the level and exercise used for Fig. 11 have been deliberately chosen to demonstrate the bias. Other levels of Python exercise show lower sensitivity to filtering—differences between filtered and unfiltered failure rates are around 0.1. Similarly, filtering has a small effect on the failure rate in other programming exercises that we explored. One factor leading to the high effect of filtering is the topic of the used level. It focuses on editing existing code and finding errors in nearly correct solutions. Other levels require students mostly to write new code from scratch. Although filtering is not always necessary, it is highly useful in some cases.

Discussion

To conclude, we summarise the main points of our review and outline directions for future work.

The Use of Complexity and Difficulty in AIED

One of our points in this paper is that it is useful to distinguish between complexity and difficulty. However, we do not argue for a search for some correct, universal measures of these concepts. Measures of complexity and difficulty can be used in several ways in the development of adaptive learning systems. When using these notions, it is necessary to take the purpose of their application into account and to choose specific measures accordingly. Let us discuss several typical AIED applications and their requirements on measures.

Adaptive choice of items. In computerized adaptive testing and adaptive practice, one of the aims is to present students with items of appropriate difficulty, i.e., neither too easy nor too difficult (Pelánek et al., 2017). In these applications, the difficulty measures are thus the main focus. Complexity measures here serve as a proxy that helps with the "cold start" problem with difficulty measures—before we have (sufficiently large) student data, difficulty measures cannot be computed. In this case, we are thus interested in a complexity measure that correlates with difficulty as much as possible (Benedetto et al., 2020). The interpretability of such complexity measures is not fundamental.

Preparation of items. Another case where we seek a high correlation between complexity and difficulty measures is the preparation of items, particularly automated content generation (Kurdi et al., 2020). One approach to content generation is to generate many items and then use the difficulty prediction to filter suitable ones. Good

predictions of difficulty are also valuable for human content authors. Using such measures, the authors can get instant feedback while preparing items and can tune the item to a better fit for the expected use of items. Note, however, that this application requires care. In the case of text complexity, this approach has been described as "writing to the formula" (Benjamin, 2012); mechanical use of this approach can lead to pedagogically unsuitable material.

Sequencing of items. Learning systems that are not adaptive provide to students items in a fixed sequence. In this case, it is usually a good practice to order the items from easier to more difficult, i.e., with respect to difficulty (Scheiter & Gerjets, 2002). However, it makes sense to take into account also complexity measures, e.g., it may not be a good idea to start with an item that is easy (can be answered correctly by most students), but otherwise rather complex (and thus potentially discouraging to new students). If we want to automatize the construction of such ordering, we may need a suitable compromise measure.

Item management and feedback for content authors. Difficulty and complexity measures are useful for the management of items (e.g., the removal of unsuitable items, updates of items) and as feedback for content authors (e.g., which items work as expected). In this case, we are interested in measures that are easily interpretable, and we do not require a single number. For this use case, it is useful to have measures that consider different aspects of items, and we do not necessarily seek measures that highly correlate. In fact, outliers and misalignment of measures can give us useful insight useful for the management and further development of items.

The use case significantly influences the choice of exact metrics and the interpretation of results. Consider as a specific example the case of items for which two measures do not correlate. In the adaptive choice use case, this situation indicates a "problem with measures"—we want to find measures that correlate well for all items, i.e., the action taken should be "try to find better measures." In the item management use case, the result may indicate a "(potential) problem with items"—if other items correlate, but a few do not, it may mean that these items are in some way problematic and should be removed or updated.

Measures and Their Relations

Details of complexity and difficulty measures depend on specific applications. Nevertheless, there are many common elements. Particularly difficulty measures are mostly universal since they depend on data on student performance, not on specifics of items. The common difficulty measures are the failure rate, median response time, and their group invariant (de-biased) versions computed by models (e.g., item response theory models or the Elo rating system).

Complexity measures depend on the specific format of items. Nevertheless, even for them, it is possible to find several common approaches:

- *size*, i.e., how long is the item statement (e.g., word, sentence, or text length, the length of an expression or equation, the number of lines of code of a program, the size of a grid of a logic puzzle),
- the *length* of solution (e.g., the number of steps in a sample solution),

– *concepts* (e.g., part-of-speech tags or types of subordinate clauses, arithmetic operations, control structures in code), which are quantified as a simple count or a weighted sum, often using weights based on a variation of TF-IDF (term frequency—inverse document frequency).

Whatever specific application of complexity and difficulty measure we are aiming for, it is useful to understand relations among candidate measures. A primary tool for this analysis is a simple pairwise correlation analysis. When we have a large number of measures, it may be useful to use unsupervised machine learning techniques to find clusters of related measures. For example, several studies have used principal component analysis to group complexity measures (Sheehan et al., 2014; Graesser et al., 2004). In our analysis, we have illustrated the use of the canonical correlation analysis—a technique used to study relations between two sets of variables. This technique naturally fits the setting of complexity and difficulty measures and can provide useful insights.

As our overview of results from several domains clearly shows, relations between complexity and difficulty measures vary quite a lot. In some cases, we can get a high correlation between complexity and difficulty even when using relatively simple measures. This is the case for some logic puzzles (e.g., *Rush hour* where the length of the solution path is predictive of difficulty) or introductory programming (where the number of concepts used in the sample solution is predictive of difficulty).

In other cases, it is challenging to capture complexity using measures that can be easily computed, and it may be nearly impossible to obtain a high correlation between complexity and difficulty using reasonably simple measures. This is the case particularly for items that combine natural language text and some underlying domain principles—a typical case is word problems in mathematics, which we discussed in some detail. Consequently, for some domains, some of the above-outlined applications may not be achievable. However, even simple measures can be useful; we just have to focus on applications other than prediction. For example, even simple complexity measures can be useful for sequencing or for feedback to content authors.

Distinguishing Complexity and Difficulty

We believe that it is useful to carefully distinguish the notions of complexity and difficulty. The measures that we have discussed could be, of course, used even without such distinction. We can analyze and employ measures like *the length of item statement* or *median response time* without giving them labels *complexity* and *difficulty* measures (or giving them all the same label). However, distinguishing the two notions leads to greater clarity, easier communication, and inspiration for both research and development.

The clear treatment of complexity and difficulty is helpful particularly for highlighting similarities between studies in different educational domains. There are many studies that focus on the prediction of a difficulty measure based on complexity measures (see, e.g., our discussion in Section 4). However, when discussing the work directly in terms of specific measures, these studies may look quite unrelated. Once we have clear terminology, transfer across domains is easier. For example, text complexity measures can be very useful in the development of systems for learning mathematics or

programming (Daroczy et al., 2015; Sheard et al., 2013). A well-used terminology can lead to easier communication and better presentation of research results—a message of a study is clearer when presented as a relationship between complexity and difficulty measures instead of a relationship among a group of specific measures.

A clear terminology also facilitates generalization and provides inspiration relevant for both practice and research. In the introductory programming setting, we may start with a question "Should we sequence problems with respect to success rate or the number of used programming concepts?" Once we have the clear terminology of complexity and difficulty at hand, this naturally leads to a more general question "Should we sequence problems with respect to difficulty or complexity?" And this, in turn, suggests other useful questions: "Has this question been studied in other domains? What other complexity and difficulty measures may be relevant?"

Limitations and Future Work

An important topic concerning complexity and difficulty measures is the *role of context*. In this paper, we focus mainly on basic measures that do not take the context into account and provide only a brief discussion and illustrations of the role of context and potential biases. This topic requires further clarification and research into methods for dealing with it.

Another topic that requires more attention is how to incorporate *scaffoldings* into measures of difficulty and complexity. The use of scaffoldings is a crucial pedagogical principle in learning systems (Jumaat & Tasir, 2014). However, with basic measures of complexity, scaffoldings tend to lead to paradoxical results: increasing complexity (item statements are longer) but decreasing difficulty (items are easier to solve). We provide an analysis of the role of scaffoldings for the case of programming exercises, and the analysis suggests a suitable way of incorporating scaffoldings into measures. The topic, however, deserves more attention in further work.

A similar but more challenging topic is the use of *hints*. Hints can be seen as an inherent part of an item, which influences both its complexity and difficulty. The role of hints is very dependent on their specific use in a particular learning system, particularly on the chosen approach to the availability of hints. When are hints available? On-demand (whenever students want), after a time limit, or based on a decision of an adaptive algorithm based on a student model? The presence of hints can also lead to "gaming the system" behavior, which can have a nontrivial impact on difficulty measures (Baker et al., 2008). For these reasons, it is challenging to study difficulty and complexity measures in the presence of hints. Nevertheless, due to the common presence of hints in intelligent tutoring systems, this challenge should be addressed in future research.

References

Aleven, V., McLaughlin, E. A., Glenn, R. A., & Koedinger, K. R. (2016). Instruction based on adaptive learning technologies. In *Handbook of research on learning and instruction*. Routledge.

Alvarez, A., & Scott, T. A. (2010). Using student surveys in determining the difficulty of programming assignments. *Journal of Computing Sciences in Colleges, 26*(2), 157–163.

Amendum, S. J., Conradi, K., & Hiebert, E. (2018). Does text complexity matter in the elementary grades? A research synthesis of text difficulty and elementary students' reading fluency and comprehension. *Educational Psychology Review, 30*(1), 121–151.

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths J., & Wittrock, M. C. (2000). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives.*, abridged edition. Pearson.

Aponte, M.-V., Levieux, G., & Natkin, S. (2011). Measuring the level of difficulty in single player video games. *Entertainment Computing, 2*(4), 205–213.

Ascalon, M. E., Meyers, L. S., Davis, B. W., & Smits, N. (2007). Distractor similarity and item-stem structure: Effects on item difficulty. *Applied Measurement in Education, 20*(2), 153–170.

Ayako Hoshino, H. N. (2010). Predicting the difficulty of multiple-choice close questions for computer-adaptive testing. *Research in Computing Science, 47*(2), 279–292.

Bailin, A., & Grafstein, A. (2001). The linguistic assumptions underlying readability formulae: A critique. *Language & Communication, 21*(3), 285–301.

Baker, F. B. (2001). *The basics of item response theory*. ERIC.

Baker, R. S. (2016). Stupid tutoring systems, intelligent humans. *International Journal of Artificial Intelligence in Education, 26*(2), 600–614.

Baker, R., Walonoski, J., Heffernan, N., Roll, I., Corbett, A., & Koedinger, K. (2008). Why students engage in gaming the system behavior in interactive learning environments. *Journal of Interactive Learning Research, 19*(2), 185–224.

Baldwin, P., Yaneva, V., Mee, J., Clauser, B. E., & Ha, L. A. (2020). Using natural language processing to predict item response times and improve test construction. *Journal of Educational Measurement*.

Barbu, O. C., & Beal, C. R. (2010). Effects of linguistic complexity and math difficulty on word problem solving by english learners. *International Journal of Education, 2*(2), 1–19.

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the ACM, 60*(6), 72–80.

Beckmann, J. F., & Goode, N. (2017). Missing the wood for the wrong trees: On the difficulty of defining the complexity of complex problem solving scenarios. *Journal of Intelligence, 5*(2), 15.

Beckmann, J. F., Birney, D. P., & Goode, N. (2017). Beyond psychometrics: The difference between difficult problem solving and complex problem solving. *Frontiers in Psychology, 8*, 1739.

Benedetto, L., Cappelli, A., Turrin, R., & Cremonesi, P. (2020). R2de: A NLP approach to estimating IRT parameters of newly generated questions. In *Proceedings of learning analytics & knowledge*.

Benjamin, R. G. (2012). Reconstructing readability: Recent developments and recommendations in the analysis of text difficulty. *Educational Psychology Review, 24*(1), 63–88.

Biggs, J. B., & Collis, K. F. (1981). *Evaluating the quality of learning: The SOLO taxonomy (structure of the observed learning outcome)*. Academic Press.

Bloom, B. S., Engelhart, M. B., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). Taxonomy of educational objectives. In *The classification of educational goals. Handbook 1: Cognitive domain*. Longmans Green.

Bouvier, D., Lovellette, E., Matta, J., Alshaigy, B., Becker, B. A., Craig, M., Jackova, J., McCartney, R., Sanders, K., & Zarb, M. Novice programmers and the problem description effect. In *Proceedings of the 2016 ITiCSE working group reports, ITiCSE '16* (pp. 103–118). ACM.

Brooks, C., Chavez, O., Tritz, J., & Teasley, S. (2015). Reducing selection bias in quasi-experimental educational studies. In *Proceedings of learning analytics & knowledge* (pp. 295–299). ACM.

Brusilovsky, P. L. (1992). A framework for intelligent knowledge sequencing and task sequencing. In *Proceedings of intelligent tutoring systems* (pp. 499–506). Springer.

Campbell, D. J. (1988). Task complexity: A review and analysis. *Academy of Management Review, 13*(1), 40–52.

Čechák, J., & Pelánek, R. (2019). Item ordering biases in educational data. In S. Isotani, E. Millán, A. Ogan, P. Hastings, B. McLaren, & R. Luckin (Eds.), *Proceedings of artificial intelligence in education* (pp. 48–58). Springer.

Cen, H., Koedinger, K., & Junker, B. (2006). Learning factors analysis–a general method for cognitive model evaluation and improvement. In *Proceedings of intelligent tutoring systems* (pp. 164–175). Springer.

Chen, C.-M., Liu, C.-Y., & Chang, M.-H. (2006). Personalized curriculum sequencing utilizing modified item response theory for web-based instruction. *Expert Systems with Applications, 30*(2), 378–396.

Craig, M., Smith, J., & Petersen, A. (2017). Familiar contexts and the difficulty of programming problems. In *Proceedings of computing education research* (pp. 123–127). ACM.

Csikszentmihalyi, M., & Csikszentmihalyi, I. S. (1992). *Optimal experience: Psychological studies of flow in consciousness*. Cambridge University Press.

Daroczy, G., Wolska, M., Meurers, W. D., & Nuerk, H.-C. (2015). Word problems: A review of linguistic and numerical factors contributing to their difficulty. *Frontiers in Psychology, 6*, 348.

De Ayala, R. (2008). *The theory and practice of item response theory*. The Guilford Press.

Eagle, M., & Barnes, T. (2014). Survival analysis on duration data in intelligent tutors. In *Proceedings of intelligent tutoring systems* (pp. 178–187). Springer.

Effenberger, T., Čechák, J., & Pelánek, R. (2019). Measuring difficulty of introductory programming tasks. In *Proceedings learning at scale*, pp. 1–4.

Finch, H. (2008). Estimation of item response theory parameters in the presence of missing data. *Journal of Educational Measurement, 45*(3), 225–245.

Gierl, M. J., Bulut, O., Guo, Q., & Zhang, X. (2017). Developing, analyzing, and using distractors for multiple-choice tests in education: A comprehensive review. *Review of Educational Research, 87*(6), 1082–1116.

Gluga, R., Kay, J., Lister, R., Kleitman, S., & Lever, T. (2012). Coming to terms with Bloom: An online tutorial for teachers of programming fundamentals. In *Proceedings of Australasian computing education conference* (pp. 147–156). Australian Computer Society, Inc.

Goutte, C., Durand, G., & Léger, S. (2018). On the learning curve attrition bias in additive factor modeling. In *Proceedings of artificial intelligence in education* (pp. 109–113). Springer.

Graesser, A. C., McNamara, D. S., Louwerse, M. M., & Cai, Z. (2004). Coh-metrix: Analysis of text on cohesion and language. *Behavior Research Methods, Instruments, & Computers, 36*(2), 193–202.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika, 28*(3–4), 321–377.

Huang, Y., Aleven, V., McLaughlin, E., & Koedinger, K. (2020). A general multi-method approach to design-loop adaptivity in intelligent tutoring systems. In *Proceedings of artificial intelligence in education* (pp. 124–129). Springer.

Hufkens, L. V., & Browne, C. (2019). A functional taxonomy of logic puzzles. In *IEEE conference on games (CoG)* (Vol. 2019, pp. 1–4). IEEE.

Ihantola, P., & Petersen, A. (2019). Code complexity in introductory programming courses. In *Proceedings of international conference on system sciences*.

Jarušek, P., & Pelánek, R. (2011). What determines difficulty of transport puzzles? In *Proceedings of Florida artificial intelligence research society conference* (pp. 428–433). AAAI Press.

Jumaat, N. F., & Tasir, Z. (2014). Instructional scaffolding in online learning environment: A meta-analysis. In *Proceedings of teaching and learning in computing and engineering* (pp. 74–77). IEEE.

Kelleher, C., & Hnin, W. (2019). Predicting cognitive load in future code puzzles. In *Proceedings of conference on human factors in computing systems* (pp. 1–12).

Keller, L. A., Swaminathan, H., & Sireci, S. G. (2003). Evaluating scoring procedures for context-dependent item sets. *Applied Measurement in Education, 16*(3), 207–222.

Khodeir, N. A., Elazhary, H., & Wanas, N. (2018). *Generating story problems via controlled parameters in a web-based intelligent tutoring system*. The International Journal of Information and Learning Technology.

Kiili, K., De Freitas, S., Arnab, S., & Lainema, T. (2012). The design principles for flow experience in educational games. *Procedia Computer Science, 15*, 78–91.

Koedinger, K. R., & Nathan, M. J. (2004). The real story behind story problems: Effects of representations on quantitative reasoning. *The Journal of the Learning Sciences, 13*(2), 129–164.

Koedinger, K. R., Corbett, A. T., & Perfetti, C. (2012). The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science, 36*(5), 757–798.

Kotovsky, K., Hayes, J. R., & Simon, H. A. (1985). Why are some problems hard? Evidence from tower of Hanoi. *Cognitive Psychology, 17*(2), 248–294.

Kuperman, V., Stadthagen-Gonzalez, H., & Brysbaert, M. (2012). Age-of-acquisition ratings for 30,000 english words. *Behavior Research Methods, 44*(4), 978–990.

Kurdi, G., Leo, J., Parsia, B., Sattler, U., & Al-Emari, S. (2020). A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education, 30*(1), 121–204.

Leo, J., Kurdi, G., Matentzoglu, N., Parsia, B., Sattler, U., Forge, S., Donato, G., & Dowling, W. (2019). Ontology-based generation of medical, multi-term MCQS. *International Journal of Artificial Intelligence in Education, 29*(2), 145–188.

Lin, C., Liu, D., Pang, W., & Apeh, E. (2015). Automatically predicting quiz difficulty level using similarity measures. In *Proceedings of international conference on knowledge capture* (pp. 1–8).

Linehan, C., Bellord, G., Kirman, B., Morford, Z. H., & Roche, B. (2014). Learning curves: Analysing pace and challenge in four successful puzzle games. In *Proceedings of computer-human interaction in play* (pp. 181–190). ACM.

Liu, P., & Li, Z. (2012). Task complexity: A review and conceptualization framework. *International Journal of Industrial Ergonomics, 42*(6), 553–568.

Lovett, M. C., & Anderson, J. R. (1996). History of success and current context in problem solving: Combined influences on operator selection. *Cognitive Psychology, 31*(2), 168–217.

Luchins, A. S. (1942). Mechanization in problem solving: The effect of einstellung. *Psychological Monographs, 54*(6), i.

Mesmer, H. A., Cunningham, J. W., & Hiebert, E. H. (2012). Toward a theoretical model of text complexity for the early grades: Learning from the past, anticipating the future. *Reading Research Quarterly, 47*(3), 235–258.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Miller, G. A. (1998). *WordNet: An electronic lexical database*. MIT Press.

Milton, J. (2010). The development of vocabulary breadth across the CEFR levels. In *Communicative proficiency and linguistic development: Intersections between SLA and language testing research*, pp. 211–232.

Mitkov, R., Ha, L. A., Varga, A., & Rello, L. (2009). Semantic similarity of distractors in multiple-choice tests: Extrinsic evaluation. In *Proceedings of the workshop on geometrical models of natural language semantics* (pp. 49–56). Association for Computational Linguistics.

Murray R. C., Ritter S., Nixon T., Schwiebert R., Hausmann R. G., Towle B., Fancsali S. E., & Vuong A. (2013). Revealing the learning in learning curves. In Proceedings of Artificial Intelligence in Education, (pp. 473–482). Springer.

Nixon, T., Fancsali, S., & Ritter, S. (2013). The complex dynamics of aggregate learning curves. In *Proceedings of educational data mining* (pp. 338–339).

Nuthong, S., & Witosurapot, S. (2017). Enabling fine granularity of difficulty ranking measure for automatic quiz generation. In *Proceedings of information technology and electrical engineering* (pp. 1–6). IEEE.

Pandarova, I., Schmidt, T., Hartig, J., Boubekki, A., Jones, R. D., & Brefeld, U. (2019). Predicting the difficulty of exercise items for dynamic difficulty adaptation in adaptive language tutoring. *International Journal of Artificial Intelligence in Education*, 1–26.

Papasalouros, A., Kanaris, K., & Kotis, K. (2008). Automatic generation of multiple choice questions from domain ontologies. *e-Learning*, 427–434.

Pelánek, R. (2014). Difficulty rating of sudoku puzzles: An overview and evaluation. arXiv preprint arXiv: 1403.7373.

Pelánek, R. (2016). Applications of the elo rating system in adaptive educational systems. *Computers & Education, 98*, 169–179.

Pelánek, R. (2017). Bayesian knowledge tracing, logistic models, and beyond: An overview of learner modeling techniques. *User Modeling and User-Adapted Interaction, 27*(3), 313–350.

Pelánek, R. (2018). The details matter: Methodological nuances in the evaluation of student models. *User Modeling and User-Adapted Interaction, 28*, 207–235.

Pelánek, R., & Jarušek, P. (2015). Student modeling based on problem solving times. *International Journal of Artificial Intelligence in Education, 25*(4), 493–519.

Pelánek, R., Papoušek, J., Řihák, J., Stanislav, V., & Nižnan, J. (2017). Elo-based learner modeling for the adaptive practice of facts. *User Modeling and User-Adapted Interaction, 27*(1), 89–118.

Polozov, O., O'Rourke, E., Smith, A. M., Zettlemoyer, L., Gulwani, S., & Popović, Z. (2015). Personalized mathematical word problem generation. In *Proceedings of international joint conference on artificial intelligence*.

Primi, R. (2001). Complexity of geometric inductive reasoning tasks: Contribution to the understanding of fluid intelligence. *Intelligence, 30*(1), 41–70.

Robertson S. (2004). Understanding inverse document frequency: On theoretical arguments for IDF. *Journal of Documentation*.

Robinson, P. (2001). Task complexity, task difficulty, and task production: Exploring interactions in a componential framework. *Applied Linguistics, 22*(1), 27–57.

Rosa, K. D., & Eskenazi, M. (2011). Effect of word complexity on l2 vocabulary learning. In *Proceedings of workshop on innovative use of NLP for building educational applications* (pp. 76–80). Association for Computational Linguistics.

Sao Pedro, M., Baker, R., & Gobert, J. (2013). Incorporating scaffolding and tutor context into bayesian knowledge tracing to predict inquiry skill acquisition. *In Educational Data Mining, 2013*.

Scheiter, K., & Gerjets, P. (2002). The impact of problem order: Sequencing problems as a strategy for improving one's performance. *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 24.

Schwarz, N., & Sudman, S. (2012). *Context effects in social and psychological research*. Springer Science & Business Media.

Seyler, D., Yahya, M., & Berberich, K. (2017). Knowledge questions from knowledge graphs. *Proceedings of theory of information retrieval*, pp. 11–18.

Sheard, J., Carbone, A., Chinn, D., Clear, T., Corney, M., D'Souza, D., Fenwick, J., Harland, J., Laakso, M.-J., Teague, D., et al. (2013). How difficult are exams?: A framework for assessing the complexity of introductory programming exams. In *Proceedings of australasian computing education conference* (vol. 136, pp. 145–154). Australian Computer Society, Inc.

Sheehan, K. M., Kostin, I., & Futagi, Y. (2008). When do standard approaches for measuring vocabulary difficulty, syntactic complexity and referential cohesion yield biased estimates of text difficulty. In *Proceedings of annual conference of the cognitive science society*.

Sheehan, K. M., Kostin, I., Napolitano, D., & Flor, M. (2014). The textevaluator tool: Helping teachers and test developers select texts for use in instruction and assessment. *The Elementary School Journal, 115*(2), 184–209.

Sohsah, G. N., Ünal, M. E., & Güzey, O. (2015). Classification of word levels with usage frequency, expert opinions and machine learning. *British Journal of Educational Technology, 46*(5), 1097–1101.

Susanti, Y., Nishikawa, H., Tokunaga, T., & Obari, H. (2016). Item difficulty analysis of english vocabulary questions. In *Proceedings of conference on computer supported education* (pp. 267–274).

Taylor, K., & Rohrer, D. (2010). The effects of interleaved practice. *Applied Cognitive Psychology, 24*(6), 837–848.

Thompson, B. (1984). *Canonical correlation analysis: Uses and interpretation, number 47*. Sage.

Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., & Robbins, P. (2008). Bloom's taxonomy for cs assessment. In *Proceedings of Australasian computing education* (pp. 155–161). Australian Computer Society, Inc.

Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games, 3*(3), 172–186.

Uemura, T., & Ishikawa, S. (2004). Jacet 8000 and Asia TEFL vocabulary initiative. *Journal of Asia TEFL, 1*(1), 333–347.

Van Der Linden, W. J. (2009). Conceptual issues in response-time modeling. *Journal of Educational Measurement, 46*(3), 247–272.

Van Merrienboer, J. J., & Krammer, H. P. (1990). The "completion strategy" in programming instruction: Theoretical and empirical support. In *Research on instruction: Design and effects*, pp. 45–61.

Wang, K., & Su, Z. (2016). Dimensionally guided synthesis of mathematical word problems. In *Proceedings of international joint conference on artificial intelligence* (pp. 2661–2668).

Wauters, K., Desmet, P., & Van Den Noortgate, W. (2012). Item difficulty estimation: An auspicious collaboration between data and judgment. *Computers & Education, 58*(4), 1183–1193.

Webb, N. L. (1997). Criteria for alignment of expectations and assessments in mathematics and science education. In *Number 6 in research monograph*. Council of Chief State School Officers.

Whalley, J., & Kasto, N. (2014). How difficult are novice code writing tasks?: A software metrics approach. In *Proceedings of Australasian computing education conference* (pp. 105–112). Australian Computer Society, Inc.

Yaneva, V., Baldwin, P., Mee, J., et al. (2019). Predicting the difficulty of multiple choice questions in a high-stakes medical exam. In *Proceedings of workshop on innovative use of NLP for building educational applications*, pp. 11–20.