User Model User-Adap Inter (2011) 21:99–135
DOI 10.1007/s11257-010-9086-0

ORIGINAL PAPER

An analysis of students' gaming behaviors in an intelligent tutoring system: predictors and impacts

Kasia Muldner · Winslow Burleson · Brett Van de Sande · Kurt VanLehn

Received: 30 April 2010 / Accepted in revised form: 17 November 2010 /
Published online: 4 January 2011

Abstract Students who exploit properties of an instructional system to make progress while avoiding learning are said to be "gaming" the system. In order to investigate what causes gaming and how it impacts students, we analyzed log data from two Intelligent Tutoring Systems (ITS). The primary analyses focused on six college physics classes using the Andes ITS for homework and test preparation, starting with the research question: What is a better predictor of gaming, problem or student? To address this question, we developed a computational gaming detector for automatically labeling the Andes data, and applied several data mining techniques, including machine learning of Bayesian network parameters. Contrary to some prior findings, the analyses indicated that student was a better predictor of gaming than problem. This result was surprising, so we tested and confirmed it with log data from a second ITS (the Algebra Cognitive Tutor) and population (high school students). Given that student was more predictive of gaming than problem, subsequent analyses focused on how students gamed and in turn benefited (or not) from instructional features of the environment, as well as how gaming in general influenced problem solving and learning outcomes.

K. Muldner (✉)
Department of Psychology, Arizona State University, Tempe, AZ, USA
e-mail: katarzyna.muldner@asu.edu

W. Burleson · B. Van de Sande · K. VanLehn
School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, AZ, USA
e-mail: winslow.burleson@asu.edu

B. Van de Sande
e-mail: bvds@asu.edu

K. VanLehn
e-mail: kurt.vanlehn@asu.edu

Keywords Educational data mining · Gaming · Utility of hints · Bayesian network parameter learning

1 Introduction

Students have long found ways to avoid reasoning about instructional materials, e.g., by copying from examples to generate problem solutions (VanLehn 1998), by paraphrasing text instead of self-explaining it more deeply (Chi et al. 1989), or by passively listening to tutors' didactic explanations without providing substantial contributions, even though active participation is needed for effective learning (Chi et al. 2001). A name given to shallow reasoning in the context of an Intelligent Tutoring System (ITS) is gaming: "*attempting to succeed by exploiting properties of the system rather than by learning the material*" (Baker et al. 2009). Not surprisingly, gaming is associated with reduced learning (Baker et al. 2004b), and so there have been efforts to detect gaming (Baker et al. 2006b, 2008a), understand gaming (Baker et al. 2008b; Rodrigo et al. 2008; Baker et al. 2009) and reduce gaming (Murray and VanLehn 2005; Walonoski and Heffernan 2006b). While progress has been made on each front, challenges still remain.

A key challenge pertains to understanding the causes of gaming. Several projects have tackled this challenge by looking for statistical associations with gaming (Arroyo and Woolf 2005; Baker 2007; Rodrigo et al. 2008; Baker et al. 2009). Early work focused on student features and how they correlate with gaming, such as students' goal orientation (Baker et al. 2005), attitudes (Baker et al. 2008b), and affect (Rodrigo et al. 2008). Of the studies that reported statistical relationships between student features and gaming, the maximum variance accounted for by student features was about 9% (Arroyo and Woolf 2005). In contrast, Baker et al. (2009) found that 56% of the variance in gaming was associated with lesson features, such as confusing hints and poor interface design. Because the lesson features explained more of the gaming variance than prior work involving student features, the argument was made that instructional defects are a key predictor of gaming. These findings are logical: if the student is at an impasse that appears to be caused by poor system design rather than by a lack of knowledge, then the student will game in order to work around the impasse. On the other hand, non-ITS research has shown that even when students use exactly the same instructional materials, they vary dramatically in how they choose to process them (e.g., Chi et al. 1989; Renkl 1997; VanLehn 1998). These non-ITS studies suggest that some kind of student features (e.g., knowledge, motivation) might be more important than instructional features in determining gaming.

To investigate the predictors of gaming further, we conducted an in-depth analysis of log data corresponding to several years worth of students interacting with Andes, an ITS for introductory physics (VanLehn et al. 2005). To identify gaming episodes in this data, we applied a computational gaming detector that we calibrated with a hand-analysis of the data. This detector provided the input to a series of data mining analyses that we conducted to better understand gaming. The initial research question was: What is a better predictor of gaming, *problem* or *student?*

Contrary to some prior work (e.g., Baker et al. 2009), we found that gaming is best predicted by *student*, and that by accounting for both *student* and *problem* features our approach explains 61% of the variance in the Andes data set. Four separate analyses of the Andes data support this conclusion. To extend and validate these findings, we applied our approach to a different ITS (the Algebra Cognitive Tutor, Carnegie Learning Inc. 2010) and population (high school students); we found that *student* was again a better predictor than *problem* for the majority of the analyses.

Having established that frequency of gaming depends more on who rather than what is being gamed, we next sought to characterize where students were gaming, and how gaming behaviors and subsequent learning and problem-solving outcomes depended on students who gamed frequently (high gamers) versus infrequently (low gamers). While we found that low gamers did indeed game differently than high gamers, in general a frequently gamed feature was high-level hints. Thus, to see if poor hint usability was driving gaming, the subsequent analysis focused on how hints scaffolded problem solving and learning. The findings show that when students actually try to use high-level hints, they eventually obtain the correct solution, although this may require several attempts, particularly for high gamers. However, neither high-level nor bottom-out hints elicited as much learning as we anticipated. As follow-up analysis demonstrated, this result was likely mediated by gaming. In particular, high gamers did not appear to benefit from high-level hints and in general learned less from hints than low gamers. Altogether, these analyses highlight the need for further research to obtain a better understanding of the utility of various kinds of help in ITSs and how to best promote learning for different types of students from it.

One of the techniques we relied on corresponded to Dynamic Bayesian Network (DBN) parameter learning. Bayesian networks, of which DBNs are a specific class, are established in the user modeling community as a tool for representing and reasoning about user states of interest (e.g., Mayo and Mitrovic 2001; Conati et al. 2002; Reye 2004); Bayesian networks have also been used in a data mining context outside of the educational community (e.g., for understanding tuberculosis epidemiology (Getoor et al. 2004)). However, within educational data mining (EDM), the application of Bayesian networks is more recent (Beck et al. 2008), although it shows high promise: DBN parameter mining obtained comparable results to other, arguably more established, techniques (Zhang et al. 2008).

To summarize, this paper includes three key analyses: (1) whether *student* or *problem* better predicts gaming, (2) how and where students are gaming and (3) the utility of ITS help features, including the impact of gaming and various kinds of help on problem solving and learning. Each analysis is motivated by its predecessor. In particular, because we found that *student* was such a strong predictor of gaming (analysis 1), we wanted to better understand how and what students were gaming, and whether certain groups of students (e.g., low gamers) were gaming differently than other groups (analysis 2). Since we found that students in general and high gamers in particular had a very high rate of help abuse, this led us to explore how ITS help scaffolded students, and how gaming influenced this process (analysis 3).

Our work brings the following contributions: (1) Through a variety of EDM techniques, we show that *student* is a better predictor of gaming than *problem*; (2) We identify individual differences in terms of how students game and in turn benefit (or

not) from instructional features of the environment; (3) We extend the work in (Beck et al. 2008) on Bayesian network parameter learning, both through its application to a novel data set and by subsequent analysis exploring the impact of gaming and various types of help on problem solving and learning for low versus high gamers. We presented a subset of this work in (Muldner et al. 2010), namely the gaming detector and its application to the Andes data for exploring *problem* versus *student* predictors of gaming, as well as a basic analysis on the short-term impact of ITS help. Here, we extend this work by applying our approach to a second data set, namely from the Cognitive Tutor, as well as presenting substantial new results from Bayesian parameter learning not in (Muldner et al. 2010).

We begin with a survey of related work (Sect. 2). We then present the gaming detector (Sect. 3) and the log data analysis to investigate predictors of gaming (Sect. 4); we validate this analysis by applying it to data from a different tutor and population (Sect. 5). Next, we present students' gaming profiles (Sect. 6). Since we found that high-level hints were one of the most abused features, we performed an in-depth analysis of how students used hints (Sect. 7). We then present findings on the impact of gaming on hint utility, problem solving and learning (Sect. 8). We conclude with a discussion of the findings and future work.

2 Related work

Students who frequently game tend to score lower on a post-test than students who never game (Baker et al. 2004b; Walonoski and Heffernan 2006a); likewise, a specific form of gaming, namely help abuse corresponding to skipping hints, is negatively correlated with learning (Aleven et al. 2006). Research also shows that some students are less likely to be hurt by gaming, such as ones with high initial domain knowledge (Baker et al. 2004a)—this makes sense in that if a concept is already known, gaming is not likely to erode that knowledge.

Given that some work suggests gaming hurts learning, the next question is: How wide-spread is this behavior? One study found that only about 10% of students ever gamed (Baker et al. 2004b), while in another study, all students gamed, albeit only some were frequent gamers (i.e., 15% were high gamers and another 25% were above-average gamers) (Walonoski and Heffernan 2006a). This discrepancy in gaming frequency may be due to the type of data collection method employed in each study. Baker et al. (2004b) used human observers to identify and record gaming episodes in real-time as students interacted with an ITS; it is possible that these observers missed some episodes. In contrast, Walonoski and Heffernan (2006a) employed a post-hoc rule-based detection algorithm to identify gaming frequencies (more on gaming detection below).

For the remainder of this section, we first survey work on detecting and understanding gaming, using EDM and other techniques, respectively. To discern between EDM versus other work, we rely on the following definition: "*educational data mining . . . is defined as the area of scientific inquiry centered around the development of methods for making discoveries within the unique kinds of data that come from educational settings, and using those methods to better understand students and the settings which*

they learn in" (Baker 2010). We then provide an overview of interventions designed to discourage gaming, and wrap up with EDM research on ITS help functionality.

2.1 EDM work on detecting and understanding the causes of gaming

Some of the EDM gaming-related work has focused on building machine detectors for gaming identification (Baker et al. 2004a, 2006b; Walonoski and Heffernan 2006a; Baker et al. 2008a). For instance, Baker et al. (2004a) used a density estimator with a set of Latent Response Models to predict how often a student would game the system. The gold standard to evaluate this detector was classroom observation of students' gaming behaviors. In this study, only some students' learning was hurt by gaming, and the detector accurately identified these students, but it was not accurate in recognizing the "gamed-not hurt" students (i.e., ones whose learning was unaffected by gaming). Another detector relied on various algorithms in WEKA, a machine learning package, to identify gaming (Walonoski and Heffernan 2006a)—while this detector correctly identified lack of gaming (98%), it's ability to recognize when gaming was occurring was quite low (19%).

There is also EDM work on understanding why gaming occurs. Some researchers propose that gaming is due to features of the instructional materials, including poor ITS design (Baker et al. 2009). This conjecture is based on analysis that involved (1) identifying salient lesson features, such as the total number of skills in a lesson and/or clarity of lesson interface icons; (2) clustering the features using Principal Component Analysis, and (3) checking for significant correlations between the clusters and a students' frequency of gaming. The full model explained 56% of the data variance, which was much higher than prior attempts, suggesting that lesson features are a strong predictor of gaming; likewise, Baker (2007) reported that lesson features were a stronger predictor of gaming than student features.

As far as student characteristics that drive gaming are concerned, Baker et al. (2004a) use the Latent Response Model described above to show that students' domain knowledge influences gaming frequency. In that model, relevant features such as number of errors and/or prior knowledge were included only if they reduced prediction error. Thus, the features remaining provided insight into factors influencing gaming. One of the remaining features encapsulated that if a student knew a skill then s/he had a low probability of gaming. Furthermore, the model predicted that students who were most hurt by gaming gamed on steps they did not know—this lead to the conjecture that students game on difficult steps (i.e., an unknown step is a difficult step). Other work has explored the relationship between student affect and gaming, and found that boredom is the most frequent emotion to precede gaming (Rodrigo et al. 2008).

2.2 Other work on detecting and understanding the causes of gaming

Some work does not rely on EDM for gaming detection, using instead other methods for labeling the data as gamed or not, but does use EDM for subsequent analysis of the labeled data (Baker et al. 2009; Cohen and Beal 2009)—we describe a representative sample of such work here.

One gaming detection approach entails human observation (Baker et al. 2004b; Walonoski and Heffernan 2006a). For instance, in the seminal work on gaming, Baker et al. (2004b) used several human observers in the classroom, as students interacted with an ITS. The observers identified gaming by visual inspection. This approach is challenging for several reasons. First, to avoid making students uncomfortable, observers only watched students out of the corner of their eye, and so may have missed nuances in a fast-paced classroom environment. Second, it's challenging for a human observer to be consistent in terms of ensuring all students are observed equally, and so the results may be biased towards certain students. Given these challenges related to human observation, there has subsequently been work on post hoc gaming identification. One method entails hand labeling of log data by a human coder (Baker et al. 2009). This approach affords the coder time to consider all student actions and attend to context-specific details that may be hard for a machine algorithm to recognize. However, hand-coding of data is very labor intensive and human coders may be inconsistent, particularly given the copious amounts of data that typically need to be coded. Other approaches have focused on machine-based gaming detection. For instance, the Help Tutor (Roll et al. 2006) relied on *if-then* rules that used latency between actions to recognize when students were gaming hints by skipping them. In contrast to the EDM approaches described above, these rules were designed by hand, using an expert-centric approach of relying on existing psychology work to guide the rule design. Likewise, latency between successive attempts and/or hint requests was used to identify gaming in the Wayang Outpost tutor (Cohen and Beal 2009).

In addition to detection, there is also work outside of the EDM community on understanding how student characteristics impact gaming. For instance, surveys were administered to students to study how attitudes, e.g., towards computers, influence gaming (Baker et al. 2008b). Survey methods were also used to understand how goal orientation influences gaming frequency (Baker et al. 2005), and in particular whether students with performance goals game more than those without. Contrary to expectations, no link was found between gaming and performance goal orientation; in fact, students with this orientation solved problems more slowly than other students.

2.3 Interventions for discouraging gaming

Various interventions have been developed to discourage gaming and so foster learning—the design of these interventions is often informed by EDM findings, and so for the sake of completeness, here we review a sample of this work. One strategy involved the introduction of a mandatory delay before a student could ask for a hint, to discourage students from rushing through hints (Aleven 2001; Murray and VanLehn 2005). Murray and VanLehn (2005) found that this design significantly reduced help requests, but overall did not impact learning (although students with low initial expertise did marginally learn more when given the intervention). Interestingly, the introduction of a mandatory delay resulted in a new form of gaming, where students would repeatedly cancel a step until they received proactive help from the tutor.

Other strategies for discouraging gaming include the incorporation of (1) supplementary exercises and (2) an animated agent that shows disapproval upon detection of

inappropriate student behaviors (Baker et al. 2006a). These interventions had a marginal impact on reducing overall gaming, as compared to a control condition. However, while fewer students gamed, those students who did game did not game less. Furthermore, the interventions did not affect overall student learning. Yet another approach to discourage gaming relies on visualizations of various student behaviors (Walonoski and Heffernan 2006b; Arroyo et al. 2007). Walonoski and Heffernan (2006b) found that a visualization of student behaviors did significantly reduce gaming in an ITS; impact on learning was not assessed. This was done by Arroyo et al. (2007), who found that visualizations of student progress fostered learning, while reducing some types of gaming. There is also work on reducing help abuse, a form of gaming, through written messages delivered upon its detection (Roll et al. 2006). This intervention significantly reduced help abuse, but did not impact learning. In summary, while progress has been made in discouraging gaming, more work is needed to understand how do so for all students and how to improve learning outcomes in the process.

2.4 EDM work on understanding the utility of its help-related functionality

As we mentioned above, our analysis of gaming prompted us to investigate the utility of tutor help. Therefore, here we review a representative sample of EDM work related to understanding and/or modeling ITS help functionality.

Typically, skipping through high-level hints to reach the bottom-out hint that essentially provides the solution step is labeled as gaming. In contrast, Shih et al. (2008) conjecture that bottom-out hints could be considered analogous to worked-out examples, and so viewing only the bottom-out hint might not be harmful to learning. To test this conjecture, a model was used that took into account the time a student thinks about the bottom-out hint. The model embedded the assumption that the time before an answer is produced corresponds to a student thinking about applying the hint, while the time after an answer is produced corresponds to a student reflecting about the hint. Analysis of log file data shows that bottom-out hint usage is indeed correlated with learning, as are the model time-related parameters. While this study suggests there is utility to bottom-out hints, it does not tell us whether skipping high-level hints is harmful, or what added value (if any) there is to those kinds of hints.

In general, high-level hints are intended to *elicit* information from students, while bottom-out hints *tell* information. Human tutors use a variety of elicit versus tell strategies during one-on-one tutoring sessions; these strategies are much more flexible than the ITS standard of including several high-level hints always followed by a bottom-out hint. Some researchers have investigated whether Reinforcement Learning (RL) could be applied to data from a tutoring corpus to learn elicit versus tell rules that would enable an ITS to more flexibly provide help (Chi et al. 2010a,b). While an earlier study did not show the RL approach to be significantly better than a random strategy, a subsequent model did generate better learning gains over a random approach. In this latter model, a larger set of features was used than in the original model, which included student, domain and system features. Since these features were not always ones predicted by current theories of learning, this work highlights how EDM

can extend existing theories by providing fine-grained insights about the pedagogical process.

Yet another example of EDM work investigating instructional interventions related to ITS help provision used learning decomposition (Feng et al. 2009). Three pedagogical approaches, embedded in an ITS, were compared, including: (1) scaffolding questions that appeared automatically when students generated an incorrect response and that students had to answer correctly before proceeding; (2) scaffolding questions that were instead available on demand, and that students were not required to answer; (3) delayed feedback that was provided in the form of worked-out examples after all problem solving was done. While the results are only marginally reliable, they suggest that providing delayed feedback might be better than providing scaffolding questions that students must answer. There was no difference, however, between scaffolding questions and delayed feedback when students were not forced to answer the scaffolding questions.

The work described above relied on pre and post test data to help train EDM models. Such data, however is not always available, and so alternatively, students' actions during their interaction with an ITS may be used to infer variables of interest. For instance, Beck et al. (Beck et al. 2008) relied on the BNT-SM toolkit (Chang et al. 2006) to learn the parameters of a Bayesian network designed to assess the impact of tutor help on students' problem solving and learning. The resulting parameters suggest that help is slightly more "helpful" in terms of fostering learning than no help (we provide more details on this work in Sect. 7.3). There is also work on biasing the parameter learning process through Dirichlet wavelets, initialized to represent the domain's parameters (Rai et al. 2009)—to date, however, this method has not been shown to result in better model prediction. Alternatively to using Dirichlet priors, others propose using a k-means clustering approach to find parameters for clusters of skills rather than individual skills, thereby reducing the space of parameters that need to be learned (Ritter et al. 2008).

3 The primary data and gaming detector

We now describe the primary data source and gaming detector.

3.1 The primary data

Our primary data, obtained from the Pittsburgh Learning Center DataShop, corresponds to logs of students using the Andes ITS (VanLehn et al. 2005) for assigned class homework and test preparation. This data was collected from a total of six different college physics classes over the span of about three years.

Andes tutors Newtonian physics by providing corrective feedback and hints on the target domain. Students solve problems in the Andes interface by drawing diagrams via the provided tools and typing equations—we refer to such interface actions as *entries*. Andes does not constrain students' solution generation, in that students are free to produce entries in any order they wish and/or to skip entries. Andes provides immediate feedback for correctness on students' entries, by coloring the entry red

Table 1 *Tutor-Student* turn pairs

	(a) Student: hint request		(b) Student: entry	
	Fast	Slow	Fast	Slow
(i) Tutor: bottom-out hint	*(S)Skip hint*	–	*(C)Copy hint*	–
(ii) Tutor: high-level hint	*(S)Skip hint*	–	–	–
(iii) Tutor: incorrect (red)	–	–	*(G)Guess (red only)*	– –
(iv) Tutor: correct (green)	*(P)No planning*	–	–	–

gamed cells *italicized*

(incorrect) or green (correct). As students solve problems, they can ask Andes for hints. The Andes hint sequence starts out with general information (e.g., "*Why don't you continue with the solution by working on setting the pressure at a point open to the atmosphere")* and ends with a bottom-out hint that indicates the step to enter (e.g., *"Write the equation Pa = Pr0*"). To discourage students from always clicking through to the bottom-out hint, Andes assigns a score to each problem, which is decremented slightly every time a bottom-out hint is requested. Full details on the system may be found in (VanLehn et al. 2005).

3.2 The gaming detector

We now describe the gaming detector (Muldner et al. 2010). After irrelevant actions are removed from the log data, a log consists of a time-stamped sequence of *tutor-student* turn pairs (e.g., tutor indicates an entry is incorrect, student responds by asking for a hint). To address our research questions, we needed to know which of these turn pairs corresponded to gaming. Given that the data comprised about 900,000 pairs, manual analysis was not feasible. Therefore, we first hand-analyzed a fragment of the log data to identify patterns indicative of gamed turn pairs (described below). Next, we encoded these patterns into a computational rule-based gaming detector that could automatically label the data. We then applied the detector to the data, hand-checking its output on a new data fragment, and revising as necessary.

For purposes of the analyses reported here, we classified the *tutor* turns as follows: (i) coloring an entry red (incorrect), (ii) coloring an entry green (correct), (iii) providing a bottom-out hint, or (iv) providing a high-level hint (we did not further subdivide the high-level hints since the number and characteristics of such hints varied considerably). We classified a *student's* turn as either (a) asking for a hint or (b) generating an entry. Thus, there are $4 \times 2 = 8$ types of turn pairs (see Table 1).

Each turn pair has a time duration associated with it, which is how long the student paused between seeing the tutor's turn and starting to take action. We assume that turn pairs with long durations are *not* gaming. Of the eight possible turn pairs with short durations (see Table 1), we consider the following five to be gaming:

(1–2) *Skipping a hint:* the tutor presents a hint and the student skips the hint by quickly asking for another hint (see '*S*′cells in Table 1). This suggests the student is trying to reach the bottom-out hint without even reading the preceding hints.

(3) *Copying a hint:* the tutor presents a bottom-out hint and the student quickly generates a solution entry, suggesting a shallow copy of the hint instead of learning of the underlying domain principle[1] (see 'C' cell, Table 1).
(4) *Guessing:* the tutor signals an incorrect entry, and the student quickly generates another *incorrect* entry, suggesting s/he is guessing instead of reasoning about why the entry is incorrect (see 'G' cell in Table 1). This is the only student entry for which we take into account correctness, as not doing so might incorrectly classify fixing slips as gaming (i.e., the other half of the *fast* cell next to the 'G' cell, Table 1).
(5) *Lack of planning:* the tutor signals a correct entry and the student quickly asks for a hint, suggesting reliance on hints for planning the solution (see 'P' cell, Table 1).

Other gaming detectors also consider hint abuse (i.e., skipping, lack of planning) and guessing as gaming (e.g., Walonoski and Heffernan 2006a; Cohen and Beal 2009). We also labeled copying hints as a gaming behavior because we wanted to capture instances when students were not reasoning about the bottom-out hint content (although as Sect. 6 describes, the *copy* category was gamed very infrequently). Note that *copying a hint* does not take into account the possibility that a student may copy the hint and *then* reason about it. This was explored in (Shih et al. 2008), by analyzing time spent after a hint was copied. However, a time lag after a hint could also correspond to a student reasoning about the next solution entry. How students reason after copying a bottom-out hint could be verified by obtaining verbal protocols of students interacting with an ITS. Since at this point we do not have such data, for the time being we only consider time before an entry is generated, as we felt this was more likely to correspond to reasoning about the entry.

Setting the time thresholds. Gaming detection relies on having reasonable time thresholds, one for each of the five gamed turn pairs. To set the thresholds, we obtained a value for each turn pair through a visual inspection of (1) the log file data and (2) the time distribution for each turn-pair under consideration (e.g., see Fig. 1). We were conservative when setting the thresholds. For instance, we set the *skipping hint* threshold $T < 3\,\text{s}$ [2]. The Andes log files only afford precision rounded to a full second, so skipping a hint means a student spent zero, one or two seconds on the hint before asking for another hint. While this range likely does not afford sufficient time to read all of a hint, the threshold captures instances when students are skipping most of the hint.

Unless otherwise stated, the results reported throughout this paper are based on applying the above-described gaming detector to the full Andes data set, corresponding to a set of 318 unique problems and 286 students.

[1] A high-level hint followed by a fast entry is not gaming since you can't copy high-level hints.

[2] The other thresholds were as follows: *copying* <4 s; *guessing* related to equations that were typed <4 s; *guessing* related to free-body diagrams that were drawn with interface tools <6 s. *lack of planning*<4 s.

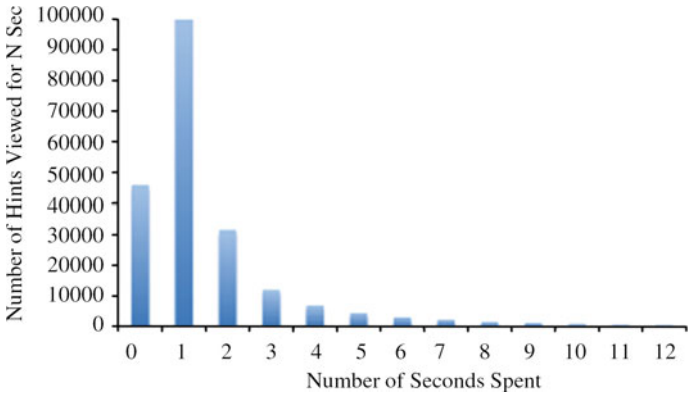


Fig. 1 Viewing time distribution for turn pairs corresponding to cells (i–ii)-(a) in Table 1

4 What is a better predictor of gaming: student or problem?

A primary research question we wanted to explore is whether student or problem features better predict gaming. To do so, we first obtained measures on gaming, as follows:

$$\text{PerGaming}_{sp} \quad \textit{percentage of gaming by a student } s \textit{ on a problem } p \tag{1}$$

$$\sum_{p=1}^{p=N} \text{perGaming}_p / N \quad \textit{average gaming by a student } s \textit{ across all } N \textit{ problems } p \textit{ solved by that student} \tag{2}$$

$$\sum_{s=1}^{s=M} \text{perGaming}_p / M \quad \textit{average gaming on a problem } p \textit{ across all } M \textit{ students } s \tag{3}$$

We use *problem* as the unit of analysis (see Eq. 1; Eq. 2 and 3 rely on it). Some research has used lesson as the primary unit of analysis (Baker et al. 2009). In fact, the ideal unit would correspond to tutor-student turn pairs, as these are where a student makes a game versus no-game decision. However, we need a unit of analysis that can be compared across students, so that we can determine whether all students tend to game at "the same" place. It would be difficult to determine if turn-pairs from one student are "the same" as turn-pairs from another student. The smallest unit of analysis that allows simple equivalence across students is the problem. Thus, we chose problem as the unit of analysis instead of lesson (too large) or tutor-student turn pairs (not equatable; too small). In these calculations, we use percentage of gaming (see Eq. 1) instead of raw values to avoid biasing the analysis towards, for instance, short problems.

We did not consider a series of rapid hint requests within a given problem as a single gamed action, for several reasons. First, lumping together a series of hint requests abstracts information on how many hints and which hints are skipped, and so fails to provide information on students' hint usage. Second, it does not make sense to lump certain gamed actions if other actions are not lumped (e.g., a series of non-gamed correct entries, or a series of slow hint requests), since this could skew the results.

An alternative approach includes using a *step* as the base measure within a problem instead of individual actions, where a step includes all the actions corresponding to a

single entry (which includes asking for hints about that entry, incorrect attempts, and the final correct entry). In this scheme, a step would be labeled as gamed or not, and the overall percentage of gamed steps within a problem would be used in the various models. This approach suffers from its own drawbacks—a prominent one being that it is not clear when a step should be labeled as gamed (e.g., when a student games a single action while generating a step? Multiple actions—if so how many?). Given these considerations, for the analysis reported here we label each action as gamed or not, and then obtain the overall percentage of gamed actions for a given problem, although we may revisit the *step* analysis in the future.

To investigate predictors of gaming we conducted four different analyses, enabling us to triangulate results across them. The first three analyses were presented in (Muldner et al. 2010), extended here with analysis on mined Bayesian parameters in Sect. 4.4.

4.1 Linear regression analysis

One way of investigating predictors of gaming is through a linear regression analysis. In the linear regression, we used $PerGaming_{sp}$ (see Eq. 1 above) as the dependent variable, and two independent variables: (1) *student*, the average gaming by a student s across all N problems p solved by that student (see Eq. 2 above), and (2) *problem*, the average gaming on a problem p across all M students s who solved that problem (see Eq. 3 above). The model we obtained is significant (F=16915, $p < 0.001$), and accounts for 60.8% of the variance ($R^2 = .608$). In this model, both *student* and *problem* yield a significant correlation with the dependent variable, but student is more strongly associated with gaming (*student*: standardized coefficient = .658, $t = 152.7$, $p < 0.001$; *problem*: standardized coefficient = .325, $t = 74.23$, $p < 0.001$).

Thus, the overall regression equation is as follows:

$$\text{perGamed}_{s,p} = \text{const} + .658\overline{\text{student}_s} + .325\overline{\text{problem}_p}$$

If we enter the independent variables separately to analyze the variance explained by each, the *student* variable accounts for 49.6% of the variance (standardized coefficient = .713), while the *problem* variable accounts for 18.6% of the variance (standardized coefficient = .434).

To explore the impact of a given data set (i.e., class or course section), we re-ran the regression analysis with a third independent variable, namely *data set id*. This variable explained only an additional 1% of the variance, showing that data set had at best a weak effect on gaming, and so we did not consider it in subsequent analysis.

4.2 Self-correlation analysis

Another way to verify whether students are more consistently gaming across problems or if instead problems are more consistent across students is to randomly sub-divide students (or problems) into buckets and then check for correlation between the buckets. To this end, to check how consistent students were in terms of gaming across problems (*student self-correlation*), we created two buckets by (1) randomly splitting problems

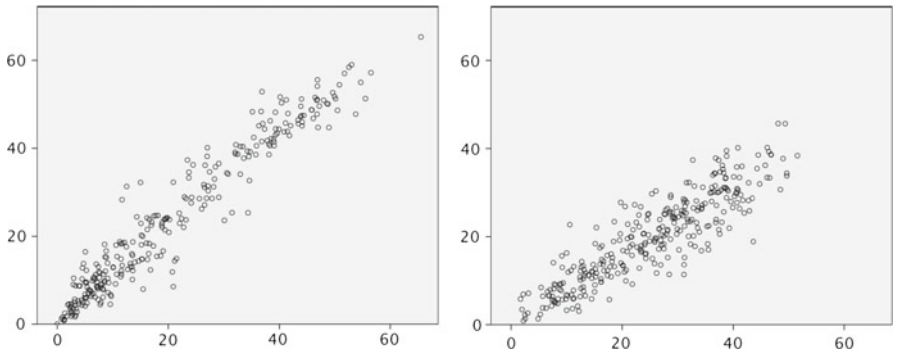


Fig. 2 Scatter plot for student ($left$) and problem ($right$) self correlations from the Andes data (percentage of gaming shown on X and Y axes)

solved by a given student across the two buckets and (2) storing in each bucket the average gaming across that bucket's problem subset, obtained using Eq. 2 above but applied to the bucket subset. A correlation analysis yielded a high degree of association between the two buckets ($r = .963$, $p < 0.001$). That is, if a student tended to game problems in the A bucket, then that student also tended to game problems in bucket B.

We used an analogous technique to check how consistent problems were across students (*problem self-correlation*). Specifically, we created two buckets by (1) randomly splitting students who solved a given problem across the two buckets and (2) storing in each bucket the average gaming across all students for that problem, obtained using Eq. 3 above. We also found a high degree of association between the two buckets ($r = .89$, $p < 0.001$). That is, if a problem was often gamed by students in bucket A, then it was also often gamed by students in bucket B (Fig. 2 shows the scatter plots for the two analyses). However, the correlation coefficient for the latter analysis was lower than for the former, suggesting that students are more consistent than problems. Specifically, if a student is a high gamer on half of the problems, then the student is also likely to be a high gamer on the other half. In contrast, if a problem is a high-gaming problem for half the students, then it is less likely to be a high-gaming problem for the other half. We verified the difference between the two correlation coefficients was reliable using Z scores (Lowry 2010); this was indeed the case ($Z = 6.9$, $p < 0.05$).

4.3 Gaming frequency distributions

Yet another way to investigate predictors of gaming is to examine histograms of gaming frequency. That is, we can look at how many students are high frequency gamers versus middle versus low frequency gamers. If differences among students are completely unimportant, and all students tend to solve roughly the same set of problems, then gaming frequency should be normally distributed (i.e., most of the gaming frequencies should cluster around the average). In fact, the student distribution is significantly different from the normal (Shapiro-Wilks test of normality $W = .89$, $p = 0.02$), and

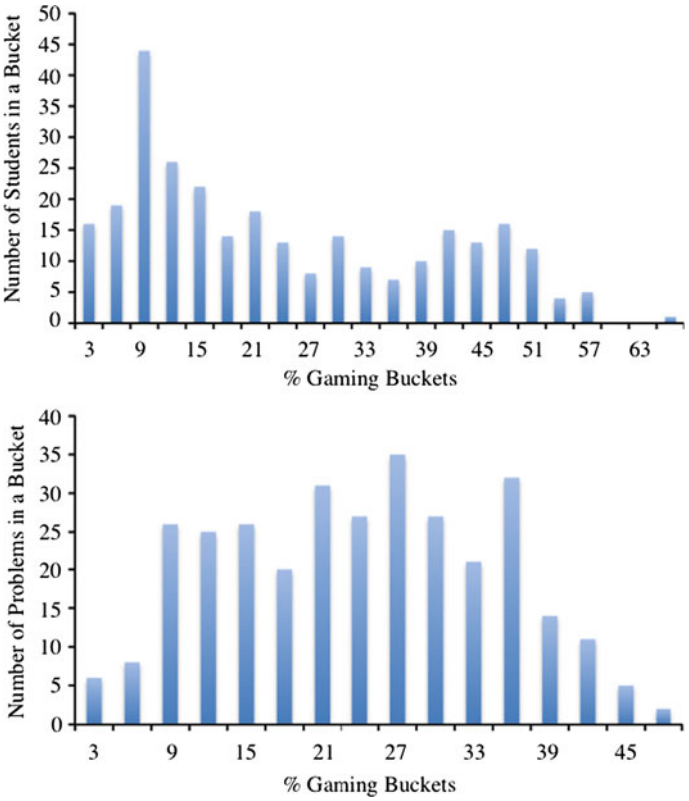


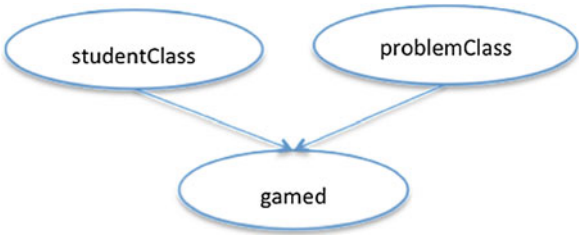
Fig. 3 Andes data on student (*top*) and problem (*bottom*) gaming distributions. Each bucket contains students (or problems) with a 3% gaming range (e.g., bucket 6 has 3% <gaming <6%)

appears bimodal (see Fig. 3, top). As the graph shows, there is one group of students who frequently game, and another group who seldom game. This again suggests that differences between students play an important role in gaming frequency.

Likewise, if the characteristics of problems are completely unimportant, then a histogram of the number of problems (y-axis) gamed at a certain range of frequencies (x-axis) should be normally distributed (Fig. 3, bottom). This is the case: the Shapiro-Wilks test of normality showed that the problem distribution is not significantly different from normal ($W = .92$, $p > 0.05$). Thus, it appears once again that characteristics of students are more important than characteristics of problems in predicting gaming.

4.4 Parameter learning for predicting gaming

We also rely on Bayesian network parameter learning to investigate how *student* versus *problem* predicts gaming, using the Bayesian network shown in Fig. 4 (top). In this network, all nodes are observable and binary, and have the following semantics:



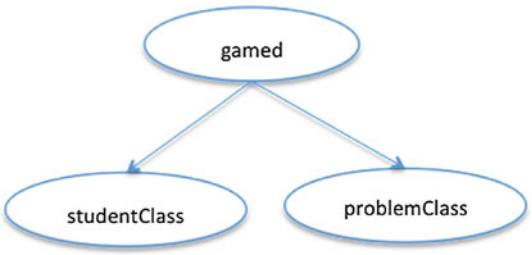
P (gamed = *true* | problemClass = *low*, studentClass = *low*) = .10
P (gamed = *true* | problemClass = *low*, studentClass = *high*) = .37
P (gamed = *true* | problemClass = *high*, studentClass = *low*) = .18
P (gamed = *true* | problemClass = *high*, studentClass = *high*) = .54

Fig. 4 Predictive Bayesian network (*top*) and corresponding parameters obtained from the Andes log data via the Netica counting algorithm (*bottom*). The parameters for P(gamed = false |. . .), not shown here, are simply 1 − P (gamed = true |. . .)

- 'studentClass': whether a student is a *low* or *high* gamer
- 'problemClass': whether a problem is a *low* or *high* gamed problem
- 'gamed': *true* if a tutor-student turn pair was gamed and *false* otherwise (see Table 1 for a listing of the turn pairs)

Thus, the network models how *student* and *problem* predict gaming. To obtain data for setting the values of the student and problem class nodes, we classified: (i) each student as a low or high gamer, based on a median split of average gaming obtained using Eq. 2 presented above and (ii) each problem as a low or high gamed problem based on a median split of average gaming obtained using Eq. 3. The data for setting the value of the 'gamed' node was provided by the gaming detector described in Sect. 3.2.

Since all nodes in the network are observable, we used Netica (Netica Reference Manual 2010), a Bayesian network toolkit, and specifically its *counting* algorithm to learn the network parameters from the Andes data. The counting algorithm uses a traditional one-pass method to determine the probabilities, which essentially amounts to counting the number of times a node takes on a certain value given each configuration of the parents (for details, see Russell and Norvig 2009). The resulting parameters are shown in Fig. 4 (bottom) and support the above findings that student is a stronger predictor of gaming than problem. In particular, the probability of gaming is higher when the 'studentClass' node is *high* and the 'problemClass' node is *low,* as compared to when the 'problemClass' node is *high* and the 'studentClass' node is *low*. Another way of looking at this is to consider the fact that when studentClass is *low*, indicating a student is not a gamer, then the probability of gaming is below 20%, regardless of whether a *low* or *high* gaming problem is solved by that student. In contrast, when studentClass is *high*, indicating that a student is a gamer, the probability of gaming is much higher, both when a *low* and a *high* gamed problem is solved. Conversely, when problem class is *low*, then the probability of gaming more depends on the type of student solving that problem (alternating between about 10% if the student is a low gamer and 37% if the student is a high gamer); likewise, when the problemClass is *high*. All together, this suggests that while *problem* does have an impact, it is less than the type of *student* solving that problem.



P (studentClass =*high* | gamed = *true*) = .86
P (studentClass =*high* | gamed = *false*) = .52
P (problemClass =*high* | gamed = *true*) = .77
P (problemClass =*high* | gamed = *false*) = .58

Fig. 5 Naïve Bayes classifier (*top*) and corresponding parameters obtained from the Andes log data via the Netica counting algorithm (*bottom*). The parameters for P([problem,student]Class = low |. . .), not shown here, are simply 1 − P ([problem, student]Class = high |. . .)

An alternative network structure to the one in Fig. 4 is a naïve Bayes classifier, such as the one shown in Fig. 5. The disadvantage of this network, however, is that by definition its structure assumes that student class and problem class are independent given evidence of gaming, which is not necessarily the case (in fact, the predictive network in Fig. 4 suggests that the two variables are related). For the sake of completeness, we used Netica to learn the parameters for this network. As Fig. 5 (bottom) shows, the results confirm the above analysis, in that the probability of gaming is higher if 'studentClass' is *high* than when 'problemClass' is *high* (see, Fig. 5, bottom; in fact, if we set the value of 'studentClass' = *high* and 'problemClass' = *low*, the probability of gaming is 0.48, higher than the $p = 0.28$ obtained when we set the value of 'studentClass' = *low* and 'problemClass' = *high*).

5 How do Gaming predictors in other tutors compare to those in Andes?

Past research has shown that both student and instructional aspects influence gaming, but to date there does not exist agreement as to which is the stronger predictor. Work suggesting that instructional features drive gaming (Baker et al. 2009) relied on data from the Algebra Cognitive Tutor (Koedinger et al. 1995; Carnegie Learning Inc. 2010). To see whether the particular data we used influenced the finding that student was the strongest predictor of gaming, we applied the gaming detector and analyses to the Cognitive Tutor data used in (Baker et al. 2009).

Before presenting the results, we provide a brief overview of the Cognitive Tutor. This tutor provides problems for students to solve, which they do by typing and/or drawing objects in the interface, much like in Andes. Also as in Andes, the Cognitive Tutor allows students to ask for hints, which start out general and end in a bottom-out hint that informs students of the step needed to generate the solution. Given this, the general gaming detection framework presented in Sect. 3.2 is also appropriate for gaming detection in the Cognitive Tutor data. Prior to applying it to the new data set we refined the gaming thresholds to make them appropriate in the context of the Cognitive tutor, using the method described in Sect. 3.2 to set the thresholds.

Once the gaming detector labeled the Cognitive Tutor data, we obtained information for *PerGaming, problem* and *student* using Eqs. 1, 2, 3 (i.e., as for the Andes data set, see Sect. 4), and analyzed the relationship between these variables using the methods presented in Sect. 4. The results, presented below, are based on data from 53 individual students 775 unique problems.

Linear Regression Analysis. A linear regression with $PerGaming_{sp}$ (Eq. 1, Sect. 4) as the dependent variable, and as the independent variables, *student* (Eq. 2, Sect. 4) and *problem* (Eq. 3, Sect. 4) produced a significant model ($F = 4588$, $p < 0.001$) that accounted for 42% of the variance ($R^2 = .420$). In this model, both *student* and *problem* are correlated with the dependent variable, although *student* yields a higher correlation than *problem* (*student*: standardized coefficient = .52, $t = 76$, $p < 0.001$; *problem*: standardized coefficient = .35, $t = 51$, $p < 0.001$).

If we enter the independent variables separately to analyze the variance explained by each, the *student* variable accounts for 30% of the variance, while the *problem* variable accounts for 15% of the variance.

Self-correlation Analysis. We used the method in Sect. 4.2 to analyze whether students were consistently gaming on problems, i.e., by randomly assigning problems a given student solved to two buckets and obtaining an overall problem average. We found a high degree of association ($r. = .973$, $p < 0.001$). In contrast, when we analyzed that problems were consistently gamed on by randomly assigning students to two buckets and obtaining an overall average, the analysis yielded a lower degree of association ($r = .55$, $p < 0.001$); the difference between the two correlation coefficients is reliable ($Z = 10.5$, $p < 0.05$). Thus, if a student is a high gamer on half of the problems, then the student is also likely to be a high gamer on the other half. In contrast, if a problem is a high-gaming problem for half the students, then it is less likely to be a high-gaming problem for the other half. This analysis confirms the Andes findings that students are more consistent and thus better predictors than problems when it comes to gaming.

Gaming frequency distributions. As we proposed in Sect. 4.3, distributions of gaming frequency across students (or problems) should be normally distributed if differences between students (or problems) do not impact gaming. The histograms for the student and problem gaming distributions from the Cognitive Tutor data are shown in Fig. 6.[3] Based on a visual inspection, the student gaming distribution is not as cleanly separated into low and high gamers as the Andes data. This distribution is not normally distributed according to the Shapiro-Wilks test of normality ($W = .64$, $p < 0.05$). However, if we remove the outlier (see bucket 38, Fig. 6, top), then the Shapiro-Wilks test reports a normal distribution ($W = .912$, $p > 0.05$).

The problem gaming distribution appears normally distributed in the middle of the graph, but not on the extreme ends. Specifically, there is a large number of problems in the Cognitive Tutor that receive very little gaming (bucket 1, Fig. 6, bottom), and there is a long tail in the upper end of the distribution, suggesting that a few Cognitive

[3] The distributions are divided into smaller buckets in Fig. 6 than in Fig. 2, because the Cognitive Tutor data had an overall lower gaming frequency, and so bigger buckets would have collapsed the data into very few buckets. The previously reported Andes results hold if we subdivide the Andes buckets into the Cognitive Tutor-sized buckets.

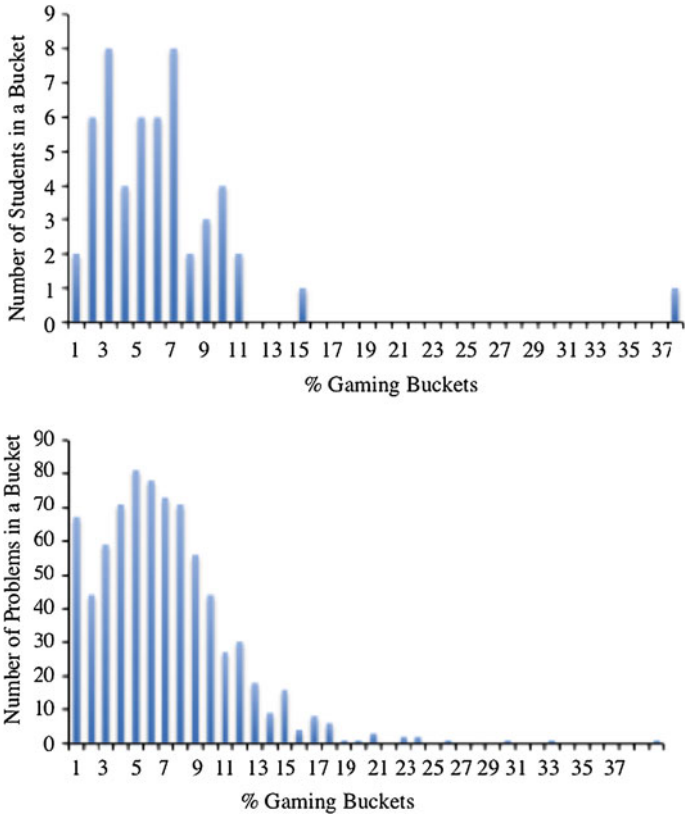


Fig. 6 Cognitive Tutor data on student (*top*) and problem (*bottom*) gaming distributions. Each bucket contains students (or problems) with a 1% gaming range (e.g., bucket 1 has 0% <gaming <1%)

P (gamed = *true* | problemClass = *low*, studentClass = *low*) = .03
P (gamed = *true* | problemClass = *low*, studentClass = *high*) = .10
P (gamed = *true* | problemClass = *high*, studentClass = *low*) = .07
P (gamed = *true* | problemClass = *high*, studentClass = *high*) = .17

Fig. 7 Parameters for the Bayesian network shown in Fig. 4 obtained with the Netica counting algorithm for the Cognitive Tutor data. The parameters for P(gamed = false | . . .), not shown here, are simply 1 − P (gamed = true | . . .)

Tutor problems are highly gamed. These two factors likely contribute to the result that this distribution is not normally distributed according to the Shapiro-Wilks test ($W = 0.71$, $p < 0.05$), which is not consistent with our hypothesis.

Parameter learning for predicting gaming. We used the Cognitive Tutor data to learn the parameters for the Bayesian network shown in Fig. 4—the results are shown in Fig. 7 (see Sect. 4.4 for a complete description of this technique). As was the case with the Andes log data, in the Cognitive Tutor the probability of gaming is higher

Table 2 Gaming opportunities for each *Tutor–student* turn pair

	(a) Student: Hint Request		(b) Student: Entry	
	fast	slow	fast	slow
(1) Tutor: B-O Hint	*S: 0.02 (.3)*	.2 (2.1)	*C: 1.8 (23.6)*	5.7 (73.9)
(2) Tutor: H-L Hint	*S:18.4 (58.6)*	5.8 (18.5)	.7 (*2.3*)	6.4 (20.6)
(3) Tutor: Incorrect	3.0 (12.4)	2.5 (10.3)	*G: 5.4 (22.1) (RED)* | 4.8 (20) (GREEN)	8.7 (35.8)
(4) Tutor: Correct	*P: 5.3 (14.5)*	3.6 (10.0)	14.1 (38.9)	13.3 (36.6)

Shown in each cell: (1) the mean % of a student responses overall (i.e., the 17 numbers in the table sum to 100%), (2) in parentheses: mean % of a student response for that row's tutor action
Fast: student action <gaming threshold; *slow:* student action >gaming threshold; *B-O:* Bottom-out, *H-L:* High-level

when *student* is true/*problem* is false than vice versa, again suggesting that *student* is a stronger predictor of gaming than *problem.*

In summary, the majority of the analyses with the Cognitive Tutor data confirms the Andes-related findings in Sect. 4 that *student* is a stronger predictor of gaming than *problem.* The '*gaming distribution*' analysis was the only one that did not support this conjecture. This latter analysis suggested that compared to the Andes problems, some problems in the Cognitive Tutor appeared to be less consistently gamed; likewise that differences between students were not as pronounced as in the Andes data set.

6 Gaming profiles: how much and where are student gaming?

The analyses above established that frequency of gaming depends more on who rather than what is being gamed, but do not provide any insight into how gaming is occurring, nor its consequences. Thus, for the remainder of the paper, we will focus on obtaining details on students' gaming and related behaviors and how these impact outcomes of interest, including problem solving and learning. For these analyses, we will use the Andes data (while it would be interesting to compare whether differences exist between the Andes and Cognitive Tutors, we leave doing so for future work). We begin with a descriptive analysis of the Andes data, which will set the stage for guiding subsequent analysis.

On average, 22.5% of tutor-student turn pairs were gamed. We first analyzed where the gaming was occurring—the results are shown in Table 2, which extends Table 1 with numeric data (as in Table 1, the italicized cells indicate gamed turn pairs). For this analysis, because we were interested in general in how students gamed, we collapsed across problems (i.e., obtained for each type of turn pair the total number of corresponding actions across all problems for a given student); Table 2 reports the average percentage for each type of turn pair.

As shown in Table 2, students most frequently took advantage of the opportunity to game when the tutor presented a high-level hint: on average, 18.4% of all student actions corresponded to gaming on these hints; when given such a hint, students gamed on it 58.6% of the time. In contrast, bottom-out hint turn-pairs were rarely

Table 3 Gaming opportunities for each tutor–student turn pair for low and high gamers

		(a) Student: Hint Request		(b) Student: Entry		
		fast	slow	fast		slow
(1) Tutor: B-O Hint	LG	*S: 0.02*	0.07	*C: 0.34*		3.2
	HG	*S: .03*	.25	*C: 3.3*		8.3
(2) Tutor: H-L Hint	LG	*S: 7.1*	6.4	0.53		7.3
	HG	*S: 29.7*	5.1	0.92		5.7
(3) Tutor: Incorrect	LG	3.2	3.2	*G: 5.5 (RED)*	5.3 (GREEN)	11.4
	HG	2.8	1.8	*G: 5.2 (RED)*	4.2 (GREEN)	6.0
(4) Tutor: Correct	LG	*P: 1.5*	3.4	20.3		21.1
	HG	*P: 9.1*	3.9	8.1		5.5

Shown in each cell is the mean % of a student response given a tutor action over all 17 possible combinations for the low gamers (LG upper half cell) and high gamers (HG lower half cell)
Fast: student action <gaming threshold; *slow:* student action >gaming threshold; *B-O:* Bottom-out, *H-L:* High-level

gamed. After skipping of high-level hints, guessing and lack of planning (see 'G' and 'P' cells, Table 2) were the next most frequently gamed turn-pairs (5.4% and 5.2%, respectively). Gaming by copying bottom-out hints was rare (see 'C' cells, Table 2).

In order to compare the gaming patterns of students who frequently gamed with those who infrequently gamed, we divided students into low gamers and high gamers based on a median split. The gaming profiles for each group are shown in Table 3. As the table highlights, high-level hints are a highly-gamed feature by both low and high gamers. Furthermore, low gamers are more likely to game by guessing than high gamers; this is the only gaming turn pair that low gamers abuse more than high-gamers. Another way to investigate differences between low and high gamers is to subdivide the data slightly differently: instead of considering at all the possible tutor-student turn pairs, as we do in Tables 2 and 3, we can instead analyze how much students game on a certain turn-pair in proportion only to the total five gaming opportunities (e.g., skipping high-level hints/total gamed events). The results of doing so are shown graphically in Fig. 8. On average, low gamers were significantly more likely than high gamers to game by guessing (46% vs. 13.2%; $F(1, 283) = 126$, $p < .01$). On the other hand, in contrast to low gamers, high gamers had a significantly higher proportion of skipped high-level hints (61.6% vs. 43.4%; $F(1, 283) = 64$, $p < .01$), lack of planning (18.5% vs. 8.9%; $F(1, 283) = 159$, $p < .01$) and bottom-out hint copying (6.5% vs. 1.7%; $F(1, 283) = 215$, $p < .01$).

7 Exploring the utility of hints for supporting problem solving and learning

Over all students' gaming opportunities, as well as proportion of gaming for high gamers, high-level hints elicited the most gaming. Some investigators propose that gaming is in part due to the fact that reading hints does not influence solution entry success, i.e., that hints are not helpful (Baker et al. 2009). We wanted to see if this was

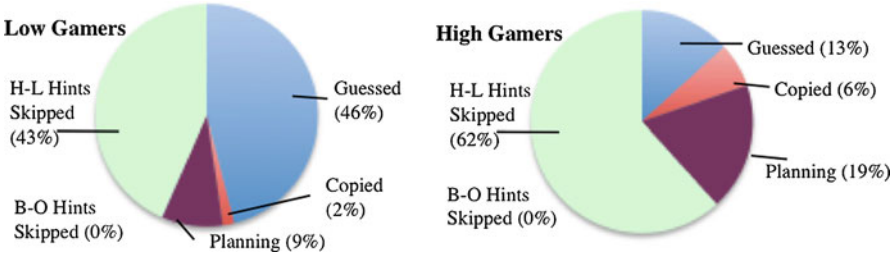


Fig. 8 Proportion of gamed-turn pairs for low and high gamers

the case for the Andes data—if we found evidence that hints were not helpful, then this would provide insight into why the high-level hints were gamed as much as they were.

7.1 Hint viewing

The most basic analysis we started with was to calculate the time students spent on hints. To do so, we obtained the latency between the provision of a hint and the next student action. On average, students spent 9.2 s vs. 5.7 s. on bottom-out versus high-level hints. High gamers spent significantly less time on hints than low gamers, both on bottom-out hints (7.5 s vs. 10.9 s; $F(1, 277) = 71$, $p < .01$) and high-level hints (3.2 s vs. 8.1 s; $F(1, 286) = 246$, $p < .01$). Thus, both low and high gamers devoted more time to bottom-out than high-level hints. High gamers' average viewing time for high-level hints was quite low, suggesting that in contrast to low gamers, these students did not pay much attention to high-level hints.

7.2 Hint utility: basic analyses

In this section, we report Andes hints' impact on short-term performance, i.e., does the hint scaffold the student to generate the entry after seeing a hint (Muldner et al. 2010). In the subsequent section, we describe hints' impact on long-term outcomes, namely learning.

If for a moment we don't consider entry correctness, high gamers tried to generate an entry only about 16% of the time after receiving a high-level hint, asking for another hint the other 84% of the time. Low gamers, on the other hand, responded to a high-level hint with an entry about 36% of the time. This is in contrast to bottom-out hints, when *both* low and high gamers responded to the hint with an entry about 97% of the time. The latter is probably due to the fact that asking for a hint after the bottom-out hint merely repeats the hint, so students only do so by accident.

For each student, we obtained the percentage of time s/he was successful at (eventually) generating the *correct* entry after receiving each type of hint (bottom out, high level). Note that (1) students may require several attempts in order to generate a correct entry and (2) if hint B is requested after hint A but prior to generating a *correct* entry, then hint A is not counted as "successful" for helping the student. We acknowledge that in some situations, hint information might be additive, i.e., seeing hint A might

add information to hint B. If this is the case, then high-level hints are at a disadvantage in this analysis.

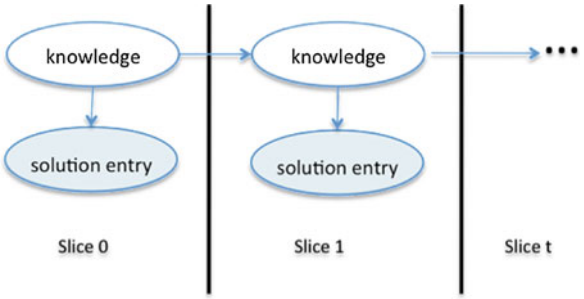
If students did generate a correct entry (or entries) after seeing a bottom-out hint, on average, they were successful in 90% of instances (i.e., the entry was correct, albeit possibly after several tries, as we show below). There was little difference between low and high gamers for this analysis (89% vs. 92%, respectively, NS difference). After high-level hints, students (eventually) generated a correct entry 73% of the time (again, this may have required several tires, as we analyze below). There was also little difference between low and high gamers (72% vs. 73%, respectively, NS difference). This suggests that high-level hints scaffolded students to generate the solution entry in about three out of four instances, albeit doing so may have taken a number of tries, as we now show.

After bottom-out hints, students required 1.1 attempts on average (1.23 for low gamers versus 1.19 for high gamers, NS), and took 29 sec. to answer correctly (34 s for low gamers versus 23 s for high gamers, $F(284, 1) = 4$, $p = .052$). This makes sense, assuming that most students copied entries from the bottom out hints, but the entries were complex enough that it took about a half-minute to do the copying, and occasionally generated typos and other minor errors. After high-level hints, on average students required 1.83 attempts to generate the correct entry; here low gamers needed significantly fewer attempts than high gamers (1.66 vs. 2.01, $F(1, 284) = 17$, $p < 0.001$), suggesting that perhaps the low gamers were more diligent about applying high-level hints. This conjecture is supported by the fact that low-gamers spent significantly longer than high-gamers to generate a correct entry after seeing a high-level hint (37 s vs. 28 s; $F(1, 286) = 9$, $p < 0.01$).

7.3 Hint utility: machine learning analyses

Above, we analyzed the short-term impact of hints with a relatively simple technique, i.e., by counting the number of times a student was able to (eventually) generate a correct response after seeing a hint. A more complex alternative is to rely on Bayesian network parameter learning to explore the impact of help on both short-term performance and long-term learning. This was originally implemented in (Beck et al. 2008), where it was applied to data from the Reading Tutor, an ITS designed to help children learn to read (Beck et al. 2004). Here, we apply this method to the Andes data. In this section, we explore the impact of hints in general for all students, while in the subsequent section we tease apart hints' influence on low versus high gamers.

For parameter learning, we relied on the BNT-SM Bayesian network parameter learning toolkit (Chang et al. 2006), which was also used in (Beck et al. 2008) and is based on the established MatLab Bayesian toolkit BNT (Murphy 2004). Learning in BNT-SM is accomplished using Expectation Maximization (Dempster et al. 1977), since as we shall see below, some of the network variables are latent, i.e., not observable. The toolkit is especially useful for parameter learning of dynamic Bayesian networks, DBNs (Dean and Kanazawa 1989). DBNs model the evolution of states of interest over time, by including so called time slices to represent particular points in time (e.g., a basic dynamic Bayesian network for modeling the evolution of knowledge



guess: P(solution entry $_t$ = *correct* | knowledge $_t$ = *unmastered*) = 0.33
slip: P(solution entry $_t$ = *incorrect* | knowledge $_t$ = *mastered*) = 0.15
learn: P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ = *unmastered*) = 0.25
forget: P(knowledge $_t$ = *unmastered* | knowledge $_{t-1}$ = *mastered*) = 0.06

Fig. 9 Basic knowledge-tracking DBN model (*top*) & corresponding parameters mined from Andes data (*bottom*)

is shown in Fig. 9, top). BNT-SM implements parameter tying for dynamic Bayesian networks (Murphy 2002), meaning that expected statistics are pooled for all nodes that share the same parameters (e.g., 'knowledge' nodes in Fig. 9). Consequently, when machine learning is applied, the same parameter for a given node class is obtained, which is the desirable outcome (although not all Bayesian toolkits yet implement parameter tying, e.g., Netica does not).

7.3.1 The basic knowledge-tracing model

We begin by presenting a simple knowledge-tracing model, one that does not take into account the impact of hints at all, shown in Fig. 9 (referred to as the *Basic model* below). This model is advocated as one appropriate for inferring student learning from problem-solving actions (Corbett and Anderson 1995; Reye 2004); Beck et al. (2008) use it as the baseline model to compare against a "Help" model, something we also do here. The Basic model, a dynamic Bayesian network, encodes two variables: (1) 'knowledge', i.e., the probability that a student knows a particular domain principle, and (2) 'solution entry', i.e., the probability that a student will generate a correct problem-solving entry as a result of applying the corresponding domain principle. In this model, all nodes have binary values, namely *correct/incorrect* for solution entries and *mastered/unmastered* for knowledge. Note that 'knowledge' is not observable, but 'solution entry' is observable (i.e., its value can be set given evidence corresponding to a student's *correct* or *incorrect* solution entry).

The Basic model captures four key variables relevant to student modeling of problem solving and learning in instructional contexts (Conati et al. 2002; Reye 2004; Beck et al. 2008):

– *guess*: the probability the student will generate a correct entry (captured by the 'solution entry' node in the network in Fig. 9) by guessing when the corresponding knowledge is in the *unmastered* state;

– *slip*: the probability the student will not generate a correct entry when the corresponding knowledge is in the *mastered* state;
– *learn*: the probability the student will learn the domain principle, which was *unmastered* in the past and which is needed to generate the corresponding entry;
– *forget*: the probability the student will forget a domain principle that was *mastered* in the past, needed to generate the corresponding entry.

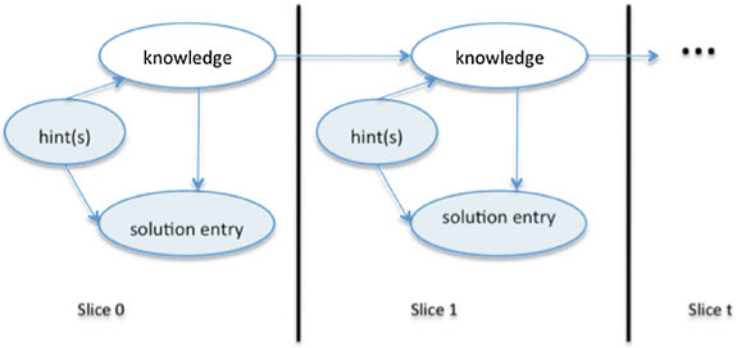
Figure 9, bottom, operationalizes these four variables in terms of Bayesian network parameters (probabilities shown are learned from the Andes data, as we describe below). BNT-SM requires that initial conditional parameter table (CPT) values are provided: to do so, we rely on the ones specified in (Beck et al. 2008), taking into account parameters advocated for Andes-specific Bayesian networks (Conati et al. 2002). These, however, are only starting values, as they are subsequently refined by the learning algorithm.

When we used BNT-SM to learn the four parameters (*guess, slip, learn, forget*) for the Andes data in the Basic model shown in Fig. 9, top, we obtained the values shown in Fig. 9, bottom. Note that BNT-SM learns parameters for individual skills—in (Beck et al. 2008), a single value was reported, suggesting that an average over the whole set of parameters learned was obtained; that is what we report here. The parameters learned from the data for the Basic model suggest that in Andes, there is a moderate probability of guessing and learning (interestingly, the probability of learning is quite a bit higher than that reported in the Reading Tutor network, i.e., Andes *learn* = 0.25 versus Reading Tutor *learn* = .08).

7.3.2 The help model

The network in Fig. 9 does not represent the influence of hints on problem solving or learning. Thus, as the next step we used BNT-SM to learn parameters for a DBN that does account for hints, referred to as the *Help model* below. The Help model was proposed in (Beck et al. 2008), and is shown in Fig. 10, top. This model extends the one in Fig. 9 with an additional observable node, 'hint(s)'. The Help model in (Beck et al. 2008) used a binary 'hint(s)' node, with values *true/false* to indicate the presence/absence of tutor help, respectively. While we are also interested in investigating the impact of various kinds of hints, something not captured with a binary-valued node, we begin with this binary 'hint(s)' node network.

The 'hint(s)' node has the following semantics: If a student asked for one or more hints *related to* (described below) a given entry, then *hint(s)=true*, and hint(s)=*false* otherwise. We say a hint could be related to an entry because when Andes displays a hint, it is shown and remains visible in a side panel, so a student could, for instance, ask for a hint, generate an incorrect entry, look back at the hint, and generate a correct entry. Thus, once a student asks for a hint, the value of 'hint(s)' node will be *true* until that entry concludes (i.e., student generates a correct entry for the current solution step or gives up and moves on). The 'hint(s)' node includes links to both the 'knowledge' and 'solution entry' nodes in the DBN, which model several hint-related influences, shown in Fig. 10, bottom; key ones include:



scaffold: P(solution entry $_t$ = *correct* | knowledge $_t$ = *unmastered*, hint(s) $_t$ = *true*) = 0.38
guess: P(solution entry $_t$ = *correct* | knowledge $_t$ = *unmastered*, hint(s) $_t$ = *false*) = 0.31
slipHint(s): P(solution entry $_t$ = *incorrect* | knowledge $_t$ = *mastered,* hint(s) $_t$ = *true*) = 0.17
slipNoHint(s): P(solution entry $_t$ = *incorrect* | knowledge $_t$ = *mastered* , hint(s) $_t$ = *false*) = 0.16
learnHint(s): P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ = *unmastered* , hint(s) $_t$ = *true*) = 0.21
learnNoHint(s): P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ = *unmastered,* hint(s) $_t$ = *false*) = 0.23
forgetHint(s): P(knowledge $_t$ = *unmastered* | knowledge $_{t-1}$ = *mastered,* hint(s) $_t$ = *true*) = 0.05
forgetNoHint(s): P(knowledge $_t$ = *unmastered* | knowledge $_{t-1}$ = *mastered,* hint(s) $_t$ = *false*) = 0.04

Fig. 10 Help DBN model (*top*) and corresponding parameters mined from Andes data (*bottom*)

- *scaffolding*, the short-term influence of a hint(s): the probability of a correct entry given that a hint(s) was requested and the relevant knowledge was *unmastered*;
- *learnWithHint(s)*: learning of a domain principle via a requested hint(s);
- *learnNoHint(s):* learning of a domain principle without a hint request.

Beck et al. (Beck et al. 2008) used BNT-SM to learn the parameters for the Help network using data from the Reading Tutor, and reported that tutor help was useful both in terms of scaffolding problem solving and learning—albeit its influence on learning was very minor: learning with a hint obtained a *learnHint* parameter of 0.088, while the *learnNoHint* parameter was only very slightly lower at 0.083.

To explore the impact of hints in Andes, we used BNT-SM to learn the Help model parameters with the Andes log data; the results are shown in Fig. 10, bottom. We found, as did (Beck et al. 2008), that when students did not posses the necessary domain knowledge, a hint(s) increased the probability of a correct entry, i.e., scaffolded problem solving, as compared to episodes where students did not ask for a hint(s) (and thus probably guessed, see *scaffold* versus *guess*, $p = .38$ vs. $p = .31$, Fig. 10, bottom), confirming the analysis in Sect. 7.2. In contrast to the results in (Beck et al. 2008), we found that overall, students learned slightly *worse* when they received hint(s), as compared to whey they did not receive hints (i.e., $p = .21$ vs. $p = 23$, *learnWithHint* versus *learnNoHint*, Fig. 10, bottom). We will return to why this may have been the case shortly.

To explore the utility of different types of hints we refined the domain of the 'hint(s)' node to include three values: *high-level (HL), bottom-out (BO), none*. To set the value of this node, we followed the approach above by looking for hints related to an entry,

scaffold-HL: P(solution entry $_t$ =*correct* | knowledge $_t$ = *unmastered*, hint(s) $_t$ =*high-level*) = 0.21
scaffold-BO: P(solution entry $_t$ =*correct* | knowledge $_t$ = *unmastered*, hint(s) $_t$ =*bottom-out*) = 0.49
guess: P(solution entry $_t$ = *correct* | knowledge $_t$ = *unmastered* , hint(s) $_t$ =*none*) = 0.25
slipWith-HL-Hint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s) $_t$ =*high-level*) = 0.32
slipWith-BO-Hint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s) $_t$ =*bottom-out*) = 0.13
slipNoHint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s) $_t$ =*none*) = 0.17
learnWith-HL-Hint(s): P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered,* hint(s) $_t$ =*high-level*) = 0.25
learnWith-BO-Hint(s): P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered,* hint(s) $_t$ =*bottom-out*) = 0.28
learnNoHint(s): P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered* , hint(s) $_t$ =*none*) = 0.23
forgetWith-HL-Hint(s): P(knowledge $_t$ =*unmastered* | knowledge $_{t-1}$ =*mastered,* hint(s) $_t$ =*high-level*) = 0.08
forgetWith-BO-Hint(s): P(knowledge $_t$ =*unmastered* | knowledge $_{t-1}$ =*mastered,* hint(s) $_t$ =*bottom-out*) = 0.05
forgetNoHint(s): P(knowledge $_t$ = *unmastered* | knowledge $_{t-1}$ = *mastered* , hint(s) $_t$ =*none*) = 0.03

Fig. 11 Parameters for the Help DBN model shown in Fig. 10 but revised with a three-valued hint node: *high-level* (*HL,* high-level hint obtained), *bottom-out* (*BO,* bottom-out hint obtained), *none* (step generated without requesting a hint)

but set the value of 'hint(s)' to: (1) *bottom-out* if a student requested a bottom-out hint related to the current entry, (2) *high-level* if a student requested a high-level hint, and no bottom-out hint(s), related to the current entry, and (3) *none* otherwise. When we applied BNT-SM to this revised network, we obtained the parameters shown in Fig. 11. The results highlight that bottom-out hints provide more short-term scaffolding for entry generation than high-level hints, something we also found with the basic analysis in Sect. 7.2. In fact, the *guess* parameter is slightly higher than the *scaffold-HL* parameter (see Fig. 11, $p = .25$ vs. $p = .21$), suggesting that high-level hints are not very helpful for immediate correct entry generation. This doesn't contradict the finding obtained via the basic analysis in Sect. 7.2, because in the latter, we analyzed whether a student would *eventually* generate a correct entry after seeing a high-level hint (possibly requiring several attempts), while here, the network assesses hints' immediate impact.

As was the case for the *scaffold* parameter, the *learn* parameter is also higher for bottom-out hint(s), as compared to high-level hint(s). However, when the network includes a three-valued 'hint(s)' node, learning without help obtains the lowest score, i.e., learning with hint(s) is better than without (see Fig. 11, bottom). Thus, one possibility for why the original Help network with a two-valued 'hint(s)' node did not suggest hints fostered learning is that it did not consider the impact of different types of hints. There are, however, other possibilities that we explore in the next section.

8 Impact of gaming on hint utility and learning

The initial analysis on the impact of help did not show that hint(s) fostered learning. While the subsequent analysis that teased apart the impact of high-level and bottom-out hints did provide indications of learning with both types of hints, learning was not as high as we hoped, particularly for high-level hints. One possibility for this finding is related to gaming. Past research suggests that gaming in general can be harmful to learning (Baker et al. 2004b; Walonoski and Heffernan 2006a), as well as help abuse in particular (Aleven et al. 2006). Furthermore, help abuse has been associated with poor learning gains in cognitive science research. For instance, VanLehn (1998) found

that when students copy from examples instead of trying to generate the solution on their own, a form of help abuse, their learning is diminished. Altogether these findings suggest that at least some of the Andes students may be learning less precisely because they obtain hints without trying to generate the solution on their own and/or trying to learn from the hints provided.

A way to test this hypothesis is to re-run the parameter learning with the original Help network, but with a refined 'hint(s)' node that includes information on gaming, as follows: 'hint(s)'=*gamed*, 'hint(s)'=*not-gamed* and 'hint(s)'=*none*. That is, the 'hint(s)'=*true* category would be replaced by 'hint(s)'=*gamed* if one or more hints related to an entry were gamed (e.g., skipped, see "fast" in Table 2; latency below threshold), and 'hint(s)'=*not-gamed* if all hints were not gamed ("slow" in Table 2; latency above threshold). Unfortunately, when we applied this method, there were very few values for the 'hint(s)'=*not-gamed* parameter: when students asked for a hint (or series of hints), they tended to game on at least one of these. Note that the data in Table 2 shows that overall, 18.1% of all student hint requests were slow (not gamed) after receiving a high-level hint—this does not contradict the previous statement on lack of data, because in Table 2, only single tutor-student pairs were considered (e.g., tutor provides a hint, student asks for another hint). In contrast, for the Help model, we consider *all* the hints related to an entry, and also consider a hint as requested if it is *related* to a entry (but not necessarily requested directly prior to generating the entry, since as we mention above, hints remain visible once requested). While students don't always game on a single turn pair related to hints, if they ask for hints then they tend to game on at least one of these. An alternative is to consider only the last hint request prior to a solution entry attempt; unfortunately, this also results in sparse data, since as shown in Table 2, students rarely copy in a shallow manner from hints (and this is the only type of gaming possible from the *hint-provided/solution-entry* turn pair).

An alternative is to investigate the utility of hints for two groups of students: the low and the high gamers. As the analysis in Sect. 7.2 suggests, low gamers are more diligent than high gamers in terms of using hints, both in terms of their willingness to use high-level hints and time spent on all types of hints. To see if the BNT-SM parameter learning also showed differences between low and high gamers and hint impact, we re-ran the learning algorithms separately for the *low* and the *high* gamers, for each of the Help networks (original 2-valued 'hint(s)' node and subsequent 3-valued 'hint(s)' node that differentiated between the two types of hints). The results, shown in Figs. 12 and 13, suggest that low gamers benefit from hints in terms of learning, while high-gamers' learning is hurt by at least some of the hints (for the sake of brevity, the *forget* parameter is not shown in the figures). Specifically, for the low gamers, the mined *learnWithHint(s)* parameter is higher than *learnNoHint(s)*, i.e., $p = .26$ vs. $p = .22$ (see Fig. 12, bolded items, top). The opposite is true for the high gamers (*learnWithHint(s)* = .17 versus *learnNoHint(s)* = .19, see Fig. 12, bolded items, bottom). If the impact the two types of hints (bottom out, high-level) is considered, then the *learn* parameter for both types of hints is higher for low gamers than high gamers (see Fig. 13, bolded items). In fact, according to this network's parameters, low gamers' learning is not affected by high-level hints.

The above analysis suggests that gaming can be harmful to learning, since it shows that high gaming students who abused hints more also learned less from them. It

Low Gamers (2-valued 'hint(s)' node):

scaffold: P(solution entry $_t$ =*correct* | knowledge $_t$ = *unmastered*, hint(s) $_t$ =*true*) = 0.37

guess: P(solution entry $_t$ = *correct* | knowledge $_t$ = *unmastered* , hint(s) $_t$ =*false*) = 0.27

slipWithHint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s) $_t$ =*true*) = 0.22

slipNoHint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s) $_t$ =*false*) = 0.15

***learnWithHint(s):* P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered,* hint(s)=*true*) = 0.26**

***learnNoHint(s):* P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered* , hint(s)=*false*) = 0.22**

High Gamers (2-valued 'hint(s)' node):

scaffold: P(solution entry $_t$ =*correct* | knowledge $_t$ = *unmastered*, hint(s) $_t$ =*true*) = 0.39

guess: P(solution entry $_t$ = *correct* | knowledge $_t$ = *unmastered* , hint(s) $_t$ =*false*) = 0.32

slipWithHint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s) $_t$ =*true*) = 0.16

slipNoHint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s) $_t$ =*false*) = 0.18

***learnWithHint(s):* P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered,* hint(s)=*true*) = 0.17**

***learnNoHint(s):* P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered* , hint(s)=*false*) = 0.19**

Fig. 12 Parameters for the DBN Help network shown in Fig. 10 with a 2-valued 'hint(s)' node for *low* and *high* gamers

Low Gamers (3-valued 'hint(s)' node):

scaffold-HL: P(solution entry $_t$ =*correct* | knowledge $_t$ = *unmastered*, hint(s) $_t$ =*high-level*) = 0.27

scaffold-BO: P(solution entry $_t$ =*correct* | knowledge $_t$ = *unmastered*, hint(s) $_t$ =*bottom-out*) = 0.43

guess: P(solution entry $_t$ = *correct* | knowledge $_t$ = *unmastered* , hint(s) $_t$ =*none*) = 0.25

slipWith-HL-Hint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s) $_t$ =*high-level*) = 0.26

slipWith-BO-Hint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s) $_t$ =*bottom-out*) = 0.14

slipNoHint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s) $_t$ =*none*) = 0.15

***learnWith-HL-Hint(s):* P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered,* hint(s)=*high-level*) = 0.31**

***learnWith-BO-Hint(s):* P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered,* hint(s)=*bottom-out*) = 0.41**

***learnNoHint(s):* P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered* , hint(s)=*none*) = 0.22**

High Gamers (3-valued 'hint(s)' node):

scaffold-HL: P(solution entry $_t$ entry =*correct* | knowledge $_t$ = *unmastered*, hint(s)=*high-level*) = 0.22

scaffold-BO: P(solution entry $_t$ =*correct* | knowledge $_t$ = *unmastered*, hint(s)=*bottom-out*) = 0.46

guess: P(solution entry $_t$ = *correct* | knowledge $_t$ = *unmastered* , hint(s)=*none*) = 0.26

slipWith-HL-Hint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s)=*high-level*) = 0.37

slipWith-BO-Hint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s)=*bottom-out*) = 0.14

slipNoHint(s): P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* hint(s)=*none*) = 0.21

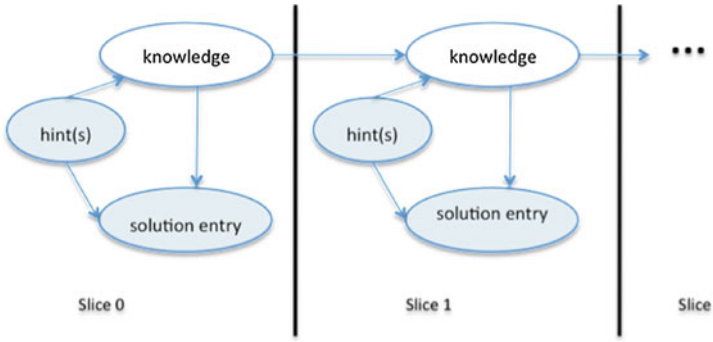
***learnWith-HL-Hint(s):* P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered,* hint(s)=*high-level*) = 0.20**

***learnWith-BO-Hint(s):* P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered,* hint(s)=*bottom-out*) = 0.25**

***learnNoHint(s):* P(knowledge $_t$ = *mastered* | knowledge $_{t-1}$ =*unmastered* , hint(s)=*none*) = 0.20**

Fig. 13 Parameters for the DBN Help network shown in Fig. 10 revised with a 3-valued 'hint(s)' node for *low* and *high* gamers

does not, however, directly tell us about the impact of gaming on learning. To explore this, we applied BNT-SM parameter learning to the network shown in Fig. 14. This network is designed to capture the impact of gaming on both problem solving and learning, by including a 'gamed' node that represents gaming behaviors. In this network, all nodes have binary values, and all nodes except knowledge are observable. The 'gamed' node is set to *true* if the gaming detector indicates that gaming occurred



guessWithGaming: P(solution entry $_t$ =*correct* | knowledge $_t$ = *unmastered*, gaming $_t$ =*true*) = 0.47
guessNoGaming: P(solution entry $_t$ =*correct* | knowledge $_t$ = *unmastered*, gaming $_t$ =*false*) = 0.47
slipWithGaming: P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* gaming $_t$ =*true*) = 0.26
slipNoGaming: P(solution entry $_t$ =*incorrect* | knowledge $_t$ = *mastered,* gaming $_t$ =*false*) = 0.15
***learnWithGaming:* P(knowledge $_t$ =*mastered* | knowledge $_{t-1}$ =*unmastered,* gaming=*true*) = 0.19**
***learnNoGaming:* P(knowledge $_t$=*mastered* | knowledge $_{t-1}$=*unmastered,* gaming=*false*) = 0.33**
forgetWithGaming: P(knowledge $_t$ =*unmastered* | knowledge $_{t-1}$ =*mastered,* gaming=*true*) = 0.06
forgetNoGaming: P(knowledge $_t$ = *unmastered* | knowledge $_{t-1}$ =*mastered* , gaming=*false*) = 0.06

Fig. 14 Gaming DBN model (*top*) and corresponding parameters mined from the Andes data (*bottom*)

for the corresponding solution entry, either (1) because students abused hints prior to generating the entry (here, only hints between two successive entries are considered, since gamed hints corresponding to previous entries are accounted for by the corresponding 'gamed' node in a previous DBN time slice), or (2) because they generated the entry by guessing or with a lack of planning. Note that in contrast to the earlier attempt to learn parameters for gamed help requests that we did not have sufficient data to do, here there is sufficient data, because in the former, we only considered gamed/ungamed help requests, while in the current Game model, all types of actions are included (solution entries, help requests, copying etc).

As the parameters obtained from BNT-SM show (Fig. 14, bottom), the probability of learning is indeed higher in the absence of gaming (the mined *learnNoGaming* parameter is higher than *learnWithGaming*, $p = .33$ vs. $p = .19$, see Fig. 14, bottom). This network also suggests that the probability of making a slip is higher when gaming occurs than without gaming (the *slipWithGaming* parameter is higher than *slipNoGaming*, $p = .26$ vs. $p = .15$, see Fig. 14, bottom). That is, when students has gamed, for instance by speeding through the hints, their next entry is more likely to be incorrect than if they had processed the hints slowly.

9 Discussion and future work

A prerequisite for reducing gaming in ITSs is understanding when and why it occurs. Past research has shown that both student and instructional aspects influence gaming,

but to date there does not exist agreement as to which is the stronger predictor, making it difficult to understand where to focus research efforts. Some researchers argue that it is the latter, i.e., instructional aspects, that drive gaming (Baker et al. 2009). This argument stems from data mining related to instructional differences between ITS lessons that resulted in a model explaining 56% of the gaming variance (we refer to the model as the *Baker model* below). Since 56% was considerably higher than other models at that time considering student characteristics, the conclusion was made that instructional features are the better gaming predictors.

To analyze predictors of gaming, we took a slightly different approach than the one in (Baker et al. 2009). Rather than extracting fine-grained features, such as number of hints in a lesson, we obtained an overall gaming statistic for a given student or problem. While this means that we did not have fine-grained information on exactly which aspects of student or problem are driving gaming, our approach does very cleanly separate problem versus student (more on this below), as well as is lightweight to implement. One of the EDM techniques we used to analyze gaming predictors corresponded to a linear regression model that took into account the average percentage of gaming by a student over all the problems s/he solved, and the average percentage of gaming on a problem over all the students. We initially applied this model to data from the Andes tutor, and found that it explained 61% of the variance, and that in this model, *student* was a stronger predictor of gaming than *problem*. We re-analyzed the data using a range of techniques: each analysis confirmed that *student* was a stronger predictor of gaming.

Given the above-stated prior findings on gaming predictors, to shed more light on the issue, we applied the gaming detector to the same data set as that used in (Baker et al. 2009). The full model explained 43% of the variance, somewhat less than the 56% explained by the Baker model. In this model, *student* was again a better predictor than *problem*, confirmed by all but one analyses. One could argue that in the Baker model *only* instructional features (i.e., lesson) are considered, and so for a fair comparison, we should use a subset of the model, i.e., one that includes only *problem* (and excludes student). However, a closer analysis of the Cognitive Tutor features used in the Baker model, which are characterized as "lesson features", suggests that these features may sometimes blur the instructional/student boundary. For instance, the category '*Difficulty, Complexity of Material, and Time Consumingness'* includes aspects such as 'student already knows the skill' and 'average probability the student will learn the skill at each application'. The category of '*Help Features'* includes 'average amount that reading hints improves future performance'—this is in fact one of the features included in the final full model. These features, however, are not only lesson specific, since they clearly take into account student characteristics. For instance, the amount that reading hints improves performance should be influenced by a students' willingness and/or ability to learn from the hint. Thus, it seems that even in the Baker model, some aspects of *student* were included in the gaming predictor analysis.

As far as the cause of the difference between our findings and ones in (Baker et al. 2009), there are a number of possibilities. First and foremost, the argument in (Baker et al. 2009) that instructional features better predict gaming was based on the fact that the corresponding model predicted more of the variance in the data than prior work with models incorporating only student features. However, as we suggest above, even

the Baker model may have relied on some student features. This makes it more difficult to compare the output of the two models, since ours is the only one that explicitly divides student and problem features. Second, it is possible that the labeling of the data influenced the findings. While we relied on a computational gaming detector, in (Baker et al. 2009) a human observer coded the data, and only coded a representative sample. There are trade offs with either approach: while a computational detector might miss some nuances that a human could pick up, the former is also more consistent than a human who is tasked with the rather arduous job of labeling thousands of episodes.

In general, our model does quite well in explaining the variance in the Andes data, surpassing existing approaches. The amount of variance explained by the model does decrease somewhat when applied to the Cognitive Tutor data. There are a number of possible reasons for this. First, the Andes ITS might have less instructional variability than the Cognitive Tutor, as suggested by the 'gaming distribution analysis', and the gaming detector may need to take into account more fine-grained features than *problem* to capture that variability. Second, we found that in the Andes model *student* was a very strong predictor of gaming, more so than in the Cognitive Tutor model; this helped to increase the Andes model's ability to explain the gaming variance. The two sets of data came from very different contexts and populations: the Andes data corresponds to college students working at home while the Cognitive Tutor data came from high school students working in classrooms. When we recently did a preliminary analysis on a set of high school honors students using Andes mostly in their classroom, we found that their gaming levels were much lower (in the 5% range) than those of the college students reported here; thus it is possible that gaming behaviors differ between these two populations and contexts, and in particular are more predictable in older populations/non-classroom settings where students are not observed by researchers/teachers, something that warrants further analysis and validation.

In addition to exploring predictors of gaming, we also analyzed differences between students in terms of gaming behaviors. We found that when we looked at gaming opportunities over the tutor-student turn pairs, students most often seized the opportunity to game after the tutor presented them with a high-level hint. However, when we analyzed the proportion of each type of gaming over the total gaming events for each class of gamers, in contrast to high-gamers (who primarily skipped hints), the low gamers had a slightly higher incidence of guessing on entries. One possible explanation for this difference, supported by literature on individual differences in help seeking behaviors (Nelson-Le Gall 1985), is that the low gamers preferred to obtain the solution on their own, without the tutor's help. Another possibility relates to Andes' scoring system. Recall that students were penalized for asking for a bottom out hint but were not penalized for guessing, and so perhaps the low gamers were simply more concerned about their Andes score than high gamers. The analysis also showed, however, that low gamers spent more time with hints and took longer to generate a solution entry after seeing a hint. Since no points were awarded for taking time, this suggests that obtaining a higher score was not the only incentive for the high-gamers, indicating that perhaps these students were motivated and/or diligent in the problem-solving process.

Prior research suggests that poor hint usability is associated with gaming (Baker et al. 2009). To see if this was the case, we investigated how students used hints in the Andes data; doing so was particularly pertinent since certain hints were a highly-gamed

feature of the Andes tutor. The basic analysis consisted of obtaining statistics on students' hint usage and utility. This analysis showed that when Andes presented a high-level hint, high gamers were quite unlikely to even *try* generating a corresponding solution entry, as compared to low gamers. If students did try to generate a solution entry, both low and high gamers were moderately successful when given high-level hints. This conclusion was based on determining if students could (eventually) generate a correct entry after being presented with a hint, prior to seeing another hint. An alternative possibility, however, is that students gave up using the high-level hint and generated the solution entry on their own.

While the basic analysis provided some indication for the utility of hints for scaffolding short term performance, it did not provide information on whether hints fostered long-term shifts, i.e., learning. To investigate this aspect, we applied the data mining techniques advocated in (Beck et al. 2008), by learning Bayesian network parameters related to the utility of hints.

Bayesian networks have been long been used in the user modeling community for representing and inferring various user states of interest, such as knowledge (Mayo and Mitrovic 2001; Conati et al. 2002), affect (Conati and Maclaren 2009) and meta-cognitive tendencies (Bunt et al. 2004; Muldner and Conati 2005). Another application of Bayesian networks, however, pertains to data mining. Parameter mining has been used in the data mining community outside of educational applications, for instance to analyze tuberculosis epidemiology (Getoor et al. 2004) and/or understand gene variation (Rodin et al. 2005). As discussed in (Heckerman 1997), there are a number of advantages of the graphical model afforded by a Bayesian network for traditional data mining, including: (1) the ability to gracefully handle missing data points, (2) during exploratory analysis, support for insight into a particular domain, including causal relationships in that domain, (3) affordance for the integration of prior knowledge about a domain into the learning process and (4) Bayesian statistical methods afford a principled approach to avoid over fitting the data. Although EDM has some unique needs as compared to traditional mining (Baker 2010), these advantages clearly also apply to educational domains. Bayesian networks offer other advantages as well. First, they can be easily extended to provide support for decision tasks, enabling them to be used for risk analysis. While traditionally this approach has not been used in educational applications, one can imagine doing so could be highly informative; for instance, "risk" could correspond to analyzing when a student looses motivation and therefore disengages from the learning task. Second, Bayesian networks afford a clear semantic interpretation of the model parameters. This is especially beneficial for EDM applications since it means that potentially novel relationships encoded in the network are directly visible. Thus, for data mining tasks in general, and EDM tasks in particular, Bayesian networks are well suited in helping to identify and/or model relationships among a large variety of variables. One disadvantage of Bayesian learning is the computational expense, although this is becoming less of an issue with the increasingly-available processor resources.

Our results using Bayesian parameter learning confirmed the basic analysis that hints scaffolded short-term problem solving, both in the original DBN Help model that only considered whether a hint was requested or not, and in a refined version that took into account the type of hint (high-level versus bottom-out). However, the former

analysis also showed that students learned slightly better in the absence of hint(s) as compared to when given hint(s), while in prior work students did very slightly better when given help (Beck et al. 2008). When we dug deeper into this issue, we found that learning from hints was influenced by several factors. First, the type of hint had an impact on learning: the network parameters indicated that students learned the most from bottom-out hints, as opposed to high-level hints. Second, gaming also had an impact. In particular, low gamers appeared to benefit from both high-level and bottom-out hints, while the high gamers only (slightly) learned from bottom-out hints.

This discussion highlights how data mining can guide subsequent research by pointing to the need to more closely investigate (1) the utility of high-level hints and (2) how student characteristics interact to influence learning and problem-solving outcomes with various kinds of hints. One way this could be achieved is through talk-aloud protocol, where students are asked to verbally express their thoughts as they interact with an ITS. While this technique is established as a means of obtaining rich data on users' cognitive processes (e.g., Chi et al. 2008), surprisingly it is not often applied in the context of ITS evaluation (although exceptions exist, e.g., D'Mello et al. 2006; Muldner and Conati 2010). We should point out that there already is substantial work on the utility of help, both in the cognitive science and the ITS communities (e.g., Phye and Sanders 1994; Chi et al. 2008; Hays et al. 2009; Razzaq and Heffernan 2009). However, there is a lack of agreement across studies as to which type of feedback is best, as is pointed out by various meta-analyses (e.g., Shute 2008). Some researchers suggest that bottom-out hints can be useful for learning (Shih et al. 2008), but do not compare the utility of various types of hints. Work directly comparing various levels of feedback found that specific feedback, i.e., providing the answer, is better than general high-level advice (Phye and Sanders 1994). In contrast, however, there is also evidence that high-level scaffolding, which is at least somewhat comparable to general hints, fosters learning better than simply providing didactic explanations (e.g., the answer) (Chi et al. 2008; Chi 2010). Furthermore, student characteristics, such as learning orientation, may interact with the utility of various types of help. For instance, Davis et al. (Davis et al. 2005) found that students with low *and* high learning orientation perform better with more specific help, as compared to general help. We also found that low and high gamers, who could arguably be labeled as high and low achievers, respectively, had better problem-solving performance and slightly better learning with bottom-out, specific hints, but that only low gamers benefited in terms of learning from high-level hints.

In addition to using it for analyzing hint impact, we also relied on Bayesian network parameter learning to analyze impact of gaming on various outcomes. Our results confirm other findings that gaming in general can be harmful to learning (Baker et al. 2004b), but to the best of our knowledge, ours is the first work to show this in the absence of pre/post test data. The analysis also highlights another reason why gaming is not an effective strategy: in the Gaming model, the probability of making a slip when gaming is present is higher than in the absence of gaming. One way to apply this particular result is through the design of tutorial interventions. For instance, students are willing to shift to strategies that foster learning if they understand that it is more cost effective to do so, for instance in terms of time (Chi and VanLehn 2007b,a). Thus, perhaps we can convince students, for instance via visualizations extending existing

ones (Walonoski and Heffernan 2006b), that gaming is not effective strategy because it results in more errors (slips).

As the next steps, we plan to follow up on some of the avenues highlighted by the data mining analyses, including evaluating further the usage of high-level hints through verbal protocols. We are also in the process of designing tailored interventions to reduce gaming in Andes. As we described in Sect. 2, interventions for reducing gaming have been successful in reducing gaming overall, but only in the less-frequent gamers, and often without influencing learning outcomes. Since findings suggest that student features are a key predictor of gaming, we believe that to be successful on all these fronts related to discouraging gaming, the system needs to tailor its interventions according to a user model that takes into account user traits relevant to gaming. To this end, we are planning on extending the gaming model we proposed here to include user traits of interest, e.g., gaming tendency, and using it in Andes to inform the interventions to discourage gaming.

Acknowledgements The authors thank the anonymous reviewers for their helpful suggestions. This research was funded the National Science Foundation, including the following grants: (1) IIS/HCC *Affective Learning Companions: Modeling and supporting emotion during learning*(#0705883); (2) *Deeper Modeling via Affective Meta-tutoring*(DRL-0910221) and (3) *Pittsburgh Science of Learning Center*(SBE-0836012).

References

Aleven, V.: Helping students to become better help seekers: towards supporting metacognition in a cognitive tutor. In: Paper Presented at German-USA Early Career Research Exchange Program: Research on Learning Technologies and Technology-Supported Education (2001)

Aleven, V., McLaren, B., Roll, I., Koedinger, K.: Toward meta-cognitive tutoring: a model of help seeking with a cognitive tutor. Int. J. Artif. Intell. Educ. **16**, 101–128 (2006)

Arroyo I., Woolf B.: Inferring learning and attitudes from a Bayesian network of log file data. In: Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED'05), Amsterdam, Netherlands, pp. 33–40 (2005)

Arroyo, I., Ferguson, K., Johns, J., Dragon, T., Meheranian, H., Fisher, D., Barto, A., Mahadevan, S., Woolf, B.: Repairing disengagement with non-invasive interventions. In: Proceedings of the 13th International Conference on Artificial Intelligence in Education (AIED'07), Los Angeles, United States, pp. 195–202 (2007)

Baker, R.: Is gaming the system state-or-trait? educational data mining through the multi-contextual application of a validated behavioral model. In: Proceedings of the Workshop on Data Mining for User Modeling, Corfu, Greece, pp. 76–80 (2007)

Baker, R., Corbett, A., Koedinger, K.: Detecting student misuse of intelligent tutoring systems. In: Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS'04), Maceio, Brazil, pp. 531–540 (2004a)

Baker, R., Corbett, A., Koedinger, K., Wagner, A.: Off-task behavior in the cognitive tutor classroom: when students "game the system". In: Proceedings of the ACM CHI 2004: Computer-Human Interaction (CHI'04), Vienna, Austria, pp. 383–390 (2004b)

Baker, R.S., Roll, I., Corbett, A., Koedinger, K.: Do performance goals lead students to game the system. In: Proceedings of the 12th International Conference on Artificial Intelligence and Education (AIED2005), Amsterdam, Netherlands, pp. 57–64 (2005)

Baker, R., Corbett, A., Koedinger, K., Evenson, E., Roll, I., Wagner, A., Naim, M., Raspat, J., Baker, D., Beck, J.: Adapting to when students game an intelligent tutoring system. In: Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS'06), Jhongli, Taiwan, pp. 392–401 (2006a)

Baker, R.S.J.d., Corbett, A.T., Koedinger, K., Roll, I.: Generalizing detection of gaming the system across a tutoring curriculum. In: Proceedings of the 11th International Conference on Intelligent Tutoring Systems (ITS'06), Jhongli, Taiwan, pp. 402–411 (2006b)

Baker, R., Corbett, A., Roll, I., Koedinger, K.: Developing a generalizable detector of when students game the system. User Model. User-Adap. Inter. **18**(3), 287–314 (2008a)
Baker, R., Walonoski, J., Heffernan, N., Roll, I., Corbett, A., Koedinger, K.: Why students engage in "gaming the system". Behavior in interactive learning environments. J. Interact. Learn. Res. **19**(2), 185–224 (2008b)
Baker, R., Corbett, A., Koedinger, K.: Educational software features that encourage and discourage "gaming the system". In: Proceedings of the 14th international conference on artificial intelligence in education (AIED'09), Brighton, UK, pp. 475–482 (2009)
Baker, R: Data mining for education. In: McGaw, B., Peterson, P., Baker, E. (eds.) International Encyclopedia of Education, vol. 7, 3rd edn, pp. 112–118. Elsevier Oxford, UK (2010)
Beck, J.E., Jia, P., Mostow, J.: Using automated questions to assess reading comprehension, vocabulary, and effects of tutorial interventions. Technol. Instr. Cogn. Learn. (TICL) **2**, 97–134 (2004)
Beck, J., Chang, K., Mostow, J., Corbett, A.: Does help? Introducing the Bayesian evaluation and assessment methodology. In: Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS'08), Montreal, Canada, pp. 383–394 (2008)
Bunt, A., Conati, C., Muldner, K.: Scaffolding self-explanation to improve learning in exploratory learning environments. In: Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS'04), Maceio, Brazil, pp. 656–667 (2004)
Carnegie Learning Inc.: Retrieved April 1, 2010, from http://www.carnegielearning.com/ (2010)
Chang, K., Beck, J., Mostow, J., Corbett, A.: A Bayes net toolkit for student modeling in intelligent tutoring systems. In: Proceedings of the 11th International Conference on Intelligent Tutoring Systems (ITS'06), Jhongli, Taiwan, pp. 104–113 (2006)
Chi, M.T.H.: How adaptive is an expert human tutor? In: Proceedings of the 10th International Conference on Intelligent Tutoring Systems (ITS'10), Pittsburgh, United States, pp. 401–113 (2010)
Chi, M., VanLehn, K.: The impact of explicit strategy instruction on problem-solving behaviors across intelligent tutoring systems. In: Proceedings of the 29th annual conference of the cognitive science society, Nashville, Tennessee, pp. 167–172 (2007a)
Chi, M., VanLehn, K.: Accelerated future learning via explicit instruction of a problem solving strategy. In: Proceedings of the 13th international conference on artificial intelligence in education (AIED'07), Los Angeles, United States, pp. 409–416 (2007b)
Chi, M.T.H., Bassok, M., Lewis, M., Reimann, P., Glaser, R.: Self-explanations: how students study and use examples in learning to solve problems. Cogn. Sci. **15**, 145–182 (1989)
Chi, M.T.H., Siler, S.A., Jeong, H., Yamauchi, T., Hausmann, R.G.: Learning from human tutoring. Cogn. Sci. **25**, 471–533 (2001)
Chi, M.T.H., Roy, M., Hausmann, R.: Observing tutoring collaboratively: insights about tutoring effectiveness from vicarious learning. Cogn. Sci. **32**(2), 301–341 (2008)
Chi, M., VanLehn, K., Litman, D.: Do micro-level tutorial decisions matter: applying reinforcement learning to induce pedagogical tutorial tactics. In: Proceedings of the 10th International Conference on Intelligent Tutoring Systems (ITS'10), Pittsburgh, United States, pp. 184–193 (2010a)
Chi, M., VanLehn, K., Litman, D., Jordan, P.: Inducing effective pedagogical strategies using learning context features. In: Proceedings of the 18th International Conference on User Modeling, Adaptation and Personalization (UMAP'10), Kona, Hawaii, pp. 147–158 (2010b)
Chi, M., VanLehn, K., Litman, D., Jordan P.: Empirically evaluating the Application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. User model User-Adap. Inter. (to appear). Special issue on educational data mining for personalized educational systems
Cohen, P., Beal, C.: Temporal data mining for educational applications. Int. J. Softw. Inform. **3**(1), 31–46 (2009)
Conati, C., Gertner, A., VanLehn, K.: Using Bayesian networks to manage uncertainty in student modeling. User Model. User-Adap. Inter. **12**(4), 371–417 (2002)
Conati, C., Maclaren, H.: Empirically building and evaluating a probabilistic model of user affect. User Model. User-Adap. Inter. **19**(3), 267–303 (2009)
Corbett, A.T., Anderson, J.R.: Knowledge tracing: modeling the acquisition of procedural knowledge. User Model. User-Adap. Inter. **4**(4), 253–278 (1995)
D'Mello, S., Craig, S., Sullins, J., Graesser, A.: Predicting affective states expressed through an emote-aloud procedure from autotutor's mixed-initiative dialogue. Int. J. Artif. Intell. Educ. **16**(1), 3–28 (2006)
Davis, W., Carson, C., Ammeter, A., Treadway, D.: The interactive effects of goal orientation and feedback specificity on task performance. Human Perform. **18**, 409–426 (2005)

Dean, T., Kanazawa, K.: A model for reasoning about persistence and causation. Comput. Intell. **5**(3), 142–150 (1989)
Dempster, L., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. **39**(1), 1–38 (1977)
Feng, M., Beck, J., Heffernan, N.T.: Using learning decomposition and bootstrapping with randomization to compare the impact of different educational interventions on learning. In: Proceedings of the 2nd International Conference on Educational Data Mining (EDM'09), Cordoba, Spain, pp. 51–60 (2009)
Getoor, L., Rhee, J., Koller, D., Small, P.: Understanding tuberculosis epidemiology using probabilistic relational models. J. Artif. Intell. Med. **30**, 233–256 (2004)
Hays, M., Lane, C., Auerbach, D., Core, M., Gomboc, D., Rosenberg, M.: Feedback specificity and the learning of intercultural communication skills. In: Proceedings of the 14th International Conference on Artificial Intelligence in Education (AIED'09), Brighton, UK, pp. 391–398 (2009)
Heckerman, D.: Bayesian networks for data mining. Data Min. Knowl. Discov. **1**, 79–119 (1997)
Koedinger, K.R., Anderson, Hadley, W.H., Mark, M.A.: Intelligent tutoring goes to school in the big city. In: Proceedings of the 3rd International Conference on Artificial Intelligence and Education (AIED'95), Washington, DC, United States, pp. 421–428 (1995)
Mayo, M., Mitrovic, A.: Optimizing ITS behaviour with Bayesian networks and decision theory. Int. J. Artif. Intell. Educ. **12**, 124–153 (2001)
Muldner, K., Conati, C.: Using similarity to infer meta-cognitive behaviors during analogical problem solving. In: Proceedings of the 10th International Conference on User Modeling (UM'05), Edinborough, Scottland, pp. 134–143 (2005)
Muldner, K., Burleson, W., de Sande, B., VanLehn, K.: An analysis of gaming behaviors in an intelligent tutoring system. In: Proceedings of the 10th International Conference on Intelligent Tutoring Systems (ITS'10), Pittsburgh, United States, pp. 195–205 (2010)
Muldner, K., Conati, C.: Scaffolding meta-cognitive skills for effective analogical problem solving via tailored example selection. Int. J. Artif. Intell. Educ. **20**(2), 99–136 (2010)
Murphy, K.: Dynamic Bayesian Networks: Representation, Inference and Learning. Ph.D. Thesis, UC, Berkeley (2002)
Murphy, K.: Bayes Net Toolbox for Matlab. http://bnt.sourceforge.net. Accessed 2010 April 1 (2004)
Murray, R.C., VanLehn, K.: Effects of dissuading unnecessary help requests while providing proactive help. In: Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED'05), Amsterdam, Netherlands, pp. 887–889 (2005)
Nelson-Le Gall, S.: Help-seeking behavior in learning review of research in education. Rev. Res. Educ. **12**, 55–90 (1985)
Netica Reference Manual. Retrieved March 1, 2010, from www.norsys.com/netica-j/docs/NeticaJ_Man.pdf (2010)
Phye, G.D., Sanders, C.: Advice and feedback: elements of practice for problem solving. Contemp. Educ. Psychol. **19**(3), 286–301 (1994)
Rai, D., Gong, Y., Beck, J.: Using Dirichlet priors to improve model parameter plausibility. In: Proceedings of the international conference on educational data mining (EDM'09), pp. 141–150 (2009)
Razzaq, L., Heffernan, N.: To tutor or not to tutor: that is the question. In: Proceedings of th 2nd International Conference on Artificial Intelligence in Education (AIED'09), Cordoba, Spain, pp. 457–464 (2009)
Renkl, A.: Learning from worked-examples: a study on individual differences. Cogn. Sci. **21**(1), 1–30 (1997)
Reye, J.: Student modeling based on belief networks. Int. J. Artif. Intell. Educ. **14**, 1–33 (2004)
Ritter, S., Harris, T., Nixon, T., Dickison, D., Murray, R.C., Towle, B.: Reducing the knowledge tracing space. In: Proceedings of the 1st International Conference On Educational Data Mining (EDM'08), Montreal, Canada, pp. 151–160 (2008)
Rodin, A., Mosley, T.H., Clark, A.G., Sing, C.F., Boerwinkle, E.: Mining genetic epidemiology data with Bayesian networks application to APOE gene variation and plasma lipid levels. J. Comput. Biol. **12**(1), 1–11 (2005)
Rodrigo, M., Baker, R., d'Mello, S., Gonzalez, M.C.T., Lagud, M., Lim, S., Macapanpan, A., Pascua, S., Santillano, J., Sugay, J., Tep, S., Viehland, N.: Comparing learners affect while using an intelligent tutoring systems and a simulation problem solving game. In: Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS'08), Montreal, Canada, pp. 40–49 (2008)
Roll, I., Aleven, V., McLaren, B., Ryu, E., Baker, R.S.J.d., Koedinger, K.: The help tutor: does metacognitive feedback improve students' help-seeking actions, skills and learning? In: Proceedings

of the Eight International Conference on Intelligent Tutoring Systems (ITS'06), Jhongli, Taiwan, pp. 360–369 (2006)
Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Los Altos, CA, Morgan-Kaufman (2009)
Shih, B., Koedinger, K., Scheines, R.: A response time model for bottom-out hints as worked examples. In: Proceedings of the 1st International Conference on Educational Data Mining (EDM'08), Montreal Canada, pp. 117–126 (2008)
Shute, V.: Focus on formative feedback. Rev. Educ. Res. **78**(1), 153–189 (2008)
VanLehn, K.: Analogy events: how examples are used during problem solving. Cogn. Sci. **22**(3), 347–388 (1998)
VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., WintersgillM.TheAndesphysics tutoring, system: Lessons learned. Int. J. Artif. Intell. Educ. **15**(3), 1–47 (2005)
VassarStats: Website for Statistical Computation. Retrieved April 1, 2010, from http://faculty.vassar.edu/lowry/VassarStats.html
Walonoski, J.A., Heffernan, N.T.: Detection and analysis of off-task gaming behavior in intelligent tutoring systems. In: Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS'06), Jhongli, Taiwan, pp. 382–391 (2006a)
Walonoski, J.A., Heffernan, N.T.: Prevention of off-task gaming behavior in intelligent tutoring systems. In: Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS'06), Jhongli, Taiwan, pp. 722–724 (2006b)
Zhang, X., Mostow, J., Beck, J.: A case study empirical comparison of three methods to evaluate tutorial behaviors. In: Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS'08), Montral, Canada, pp. 122–131 (2008)

Author Biographies

Kasia Muldner is currently a Post-Doctoral Scholar in the department of Psychology at Arizona State University (ASU); prior to this appointment, she worked as a Post-Doctoral Scholar in the School of Computing, Informatics, and Decision Systems Engineering, also at ASU. She received her Ph.D. from the University of British Columbia, Canada, from the Computer Science Department at the Laboratory for Computational Intelligence. Her research interests span Artificial Intelligence, Cognitive Science and Human Computer Interaction, with a particular focus on pedagogical applications, including the design and evaluation of intelligent tutoring systems, educational data mining, and investigating ways to foster constructive learning, for instance during analogical problem solving and learning from observing.

Winslow Burleson is an Assistant Professor of Human-Computer Interaction in the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University, and directs the Motivational Environments research group (http://hci.asu.edu). Author of over 60 scientific publications, including "best paper" at AIED 2009, and 10 patents, he received a Ph.D. in Affective Computing from the MIT Media Lab, a BA from Rice, and MSE from Stanford. He worked with the Life Long Kindergarten group and Entrepreneurial Management Unit at Harvard Business School, IBM Research, NASA-SETI Institute, the Space Telescope Science Institute and UNICEF.

Brett Van de Sande is an Assistant Research Professional in the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University, working in the area of intelligent tutoring systems. He completed his Ph.D. at Ohio State University in theoretical physics, and has since taught physics and conducted research in physics education, including the development of adaptive physics tutors.

Kurt VanLehn is a Professor of Computer Science at Arizona State University. He completed his Ph.D. at MIT, did post-doctoral research at Xerox PARC, joined the faculty of Carnegie-Mellon University in 1985, moved to the University of Pittsburgh in 1990 and joined ASU in 2008. His work involves applications of artificial intelligence to education, including intelligent tutoring systems, cognitive modeling and educational data mining.