Second Workshop on Simulated Learners

held in conjunction with

Seventeenth International Conference on Artificial Intelligence in Education (AIED 2015)

Friday, June 26, 2015
Madrid, Spain

Workshop Co-Chairs:

John Champaign[1] and Gord McCalla[2]

[1] *University of Illinois at Springfield, Springfield, IL, 62703, USA*
[2] *University of Saskatchewan, Saskatoon, S7N 5C9, Canada*

https://sites.google.com/site/simulatedlearners/

Table of Contents

Preface

This workshop, a follow-up to the successful first Simulated Learners workshop held at AIED 2013, is intended to bring together researchers who are interested in simulated learners, whatever their role in the design, development, deployment, or evaluation of learning systems. Its novel aspect is that it isn't simply a workshop about pedagogical agents, but instead focuses on the other roles for simulated learners in helping system designers, teachers, instructional designers, etc.

As learning environments become increasingly complex and are used by growing numbers of learners (sometimes in the hundreds of thousands) and apply to a larger range of domains, the need for simulated learners (and simulation more generally) is compelling, not only to enhance these environments with artificial agents, but also to explore issues using simulation that would be otherwise be too expensive, too time consuming, or even impossible using human subjects. While some may feel that MOOCs provide ample data for experimental purposes, it is hard to test specific hypotheses about particular technological features with data gathered for another purpose. Moreover, privacy concerns, ethics approval, attrition rates and platform constraints can all be barriers to this approach. Finally, with thousands of learners at stake, it is wise to test a learning environment as thoroughly as possible before deployment.

Since this is a follow-up to the 2013 workshop, we build on some of the ideas that emerged there (see proceedings at: http://goo.gl/12ODji).

The workshop explores these and other issues with the goal of further understanding the roles that simulated learners may play in advanced learning technology research and development, and in deployed learning systems.

John Champaign and Gord McCalla
Workshop Co-Chairs

An Approach to Developing Instructional Planners for Dynamic Open-Ended Learning Environments

Stephanie Frost and Gord McCalla

ARIES Lab, Department of Computer Science, University of Saskatchewan, Canada
stephanie.frost@usask.ca, mccalla@cs.usask.ca

Abstract. *Instructional planning* (IP) technology has begun to reach large online environments. However, many approaches rely on having centralized metadata structures about the learning objects (LOs). For *dynamic open-ended* learning environments (DOELEs), an approach is needed that does not rely on centralized structures such as prerequisite graphs that would need to be continually rewired as the LOs change. A promising approach is collaborative filtering based on learning sequences (CFLS) using the ecological approach (EA) architecture. We developed a CFLS planner that compares a given learner's most recent path of LOs (of length b) to other learners to create a neighbourhood of similar learners. The future paths (of length f) of these neighbours are checked and the most successful path ahead is recommended to the target learner, who then follows that path for a certain length (called s). We were interested in how well a CFLS planner, with access only to pure behavioural information, compared to a traditional instructional planner that used explicit metadata about LO prerequisites. We explored this question through simulation. The results showed that the CFLS planner in many cases exceeded the performance of the simple prerequisite planner (SPP) in leading to better learning outcomes for the simulated learners. This suggests that IP can still be useful in DOELEs that often won't have explicit metadata about learners or LOs.

Keywords: instructional planning, collaborative filtering, dynamic open-ended learning environments, simulated learning environments, simulated learners, ecological approach

1 Introduction

Online courses need to be able to personalize their interactions with their many learners not only to help each learner overcome particular impasses but also to provide a path through the learning objects (LOs) that is appropriate to that particular individual. This is the role of *instructional planning* (IP), one of the core AIED sub-disciplines. IP is particularly needed in open-ended learning environments (OELEs), where learners choose their own goals, because it has been shown that sometimes learners require an outside push to move forward

[11]. An added challenge is what we call a *dynamic open-ended* learning environment (DOELE), where both the learners and LOs are constantly changing. Some learners might leave before finishing the course, while others may join long after other learners have already begun. New material (LOs) may need to be added in response to changes in the course or the material, or to learner demand. Sometimes new material will be provided by the course developers, but the big potential is for this to be crowd sourced to anybody, including learners themselves. Other material may fade away over time.

Note that a DOELE is similar to, but not the same as, a "traditional" open-eded learning environment [8, 11]. A traditional open-ended environment also gives students choice, but mostly in the problems they solve and how they solve them, with the course itself fixed in its content, order and goals. In a DOELE everything is open-ended and dynamic, including even what is to be learned, how deeply, when it needs to be learned, and in what order.

An impediment to IP in a DOELE is that there is no centralized representation of knowledge about the content or the learners. Work has been done to make IP possible in online environments, such as [7], where authors showed that by extending the LO metadata, instructional plans could be improved to adapt based on individual learning styles as well as a resource's scheduling availability. But for IP to work in DOELEs, an approach to IP is needed where centralized course structures would not need to be continually revamped (by instructional designers, say) as learners and LOs change.

We wish to explore how IP can be done in a DOELE. We model a DOELE in the ecological approach (EA) architecture [14]. In the EA there is no overall course design. Instead, courses are conceived as collections of learning objects each of which captures usage data as learners interact with it. Over time this usage data accumulates and can be used for many pedagogical purposes, including IP [2]. Drawing inspiration from work like [1, 5], we propose a new IP algorithm based on collaborative filtering of learning sequences (CFLS). For a given learner our planner finds other learners who have traversed a similar sequence of learning objects with similar outcomes (i.e. similar paths). Then it suggests paths to the learner that were successful for these similar learners (peers) going forward.

To evaluate IP techniques in such an environment, one could implement a real course with thousands of learners using the EA to capture learner interactions with the various LOs in the course. However, after doing this it would take several years for enough learners to build up enough interactions with each LO to provide useful data to be used by an instructional planner. Also, in a course with thousands of learners, there is risk of causing confusion or inconvenience to a vast multitude if there are problems while the planner is under development. Finally, there are unanswered design questions such as the criteria to use for identifying an appropriate peer, how many LOs should be recommended for a learner before re-planning occurs, and appropriate values for many other parameters that would be used by the planner. In order to overcome these challenges and gain insight into these questions immediately, we have thus turned to simulation.

2 Simulation Environment

Before describing the CFLS planner and experiment in detail, we describe the simulation environment. The simulation is low-fidelity, using very simple abstractions of learners and LOs, as in our earlier work [6]. Each of the 40 LOs has a difficulty level and possible prerequisite relationships with other LOs. Each simulated learner has an attribute, *aptitude-of-learner*, a number between (0,1) representing a learner's basic capability for the subject and allows learners to be divided into groups: low ($\leq .3$), medium ($.4 - .7$) and high aptitude ($\geq .8$).

A number called *P[learned]* is used to represent the learning that occurred when a learner visits a LO, or the probability that the learner learned the LO. P[learned] is generated by an *evaluation function*, a weighted sum: 20% of the learner's score on a LO is attributed to *aptitude-of-learner*, 50% attributed to whether the learner has mastered all of the prerequisite LOs, 20% attributed to whether the learner had seen that LO previously, and 10% attributed to the difficulty level of the LO. We feel this roughly captures the actual influences on how likely it is that real learners would master a learning object.

The simulated learners move through the course by interacting with the LOs, one after another. After each LO is encountered by a simulated learner, the above evaluation function is applied to determine the learner's performance on the LO, the P[learned] for that learner on that LO. In the EA architecture, everything that is known about a learner at the time of an interaction with a LO (in this case, including P[learned]) is captured and associated with that LO. The order of the LOs visited can be set to random, or it can be determined by a planner such as the CFLS planner. To allow for the comparison of different planning approaches without advantaging one approach, each simulated learner halts after its 140th LO regardless of the type of planner being used.

3 Experiment

By default, the simulation starts with an empty history - no simulated learners have yet viewed any LOs. However, because the CFLS planner relies on having previous interaction data, it is necessary to initialize the environment. Thus, a simple prerequisite planner (SPP) was used to initialize the case base with a population of simulated learners. The SPP is privy to the underlying prerequisite structure and simply delivers LOs to learners in prerequisite order. As Table 1 shows, the SPP works much better than a random planner. The data from the 65 simulated learners who used the SPP thus was used to initialize the environment before the CFLS planner took over. This interaction data generated by the SPP also provides a baseline for comparison with the CFLS planner. Our simulation experiment was aimed at seeing if, with appropriate choices of b and f (described below) the CFLS planner could work as well or better than the SPP.

We emphasize that the CFLS planner has no knowledge about the underlying prerequisite structure of the learning objects. This is critical for CFLS planning to work in a DOELE. However, there are two places where clarification

Table 1. Baseline results for each group of simulated learners (high, medium and low aptitude) when visiting LOs randomly and following a simple prerequisite planner.

Planning Type / Aptitude	low	medium	high
Random	N=21	N=26	N=18
Average Score on Final Exam (P[learned])	0.107	0.160	0.235
Simple Prerequisite Planner (SPP)	N=21	N=26	N=18
Average Score on Final Exam (P[learned])	0.619	0.639	0.714

is required. First, while the SPP is running, the evaluation function will be used by the simulation to calculate P[learned] values for each LO visited. This usage data will contain implicit evidence of the prerequisite relationships. So, at a later time when the CFLS planner is given access to the same usage data, the CFLS planner could implicitly discover prerequisite relationships from the interaction data. Second, during the CFLS planner execution, the underlying prerequisite structure is still being consulted by the evaluation function. However, the CFLS planner knows nothing about such prerequisites, only the P[learned] outcome provided by the evaluation function. When simulated learners are replaced with real learners, the evaluation function would disappear and be replaced with a real world alternative, such as quizzes or other evidence to provide a value for P[learned]. Similarly, the CFLS planner does not require knowledge of the difficulty level of each LO, nor does it require knowledge of the aptitude of each learner; these are just stand-in values for real world attributes used by the simulation and would disappear when the planner is applied in a real world setting.

Different studies can use simulated student data in varying ways. In some cases, low fidelity modelling is not adequate. For example, in [4] it was found that the low fidelity method of generating simulated student data failed to adequately capture the characteristics of real data. As a result, when the simulated student dataset was used for training the cognitive diagnosis model, its predictive power was worse than when the cognitive diagnosis model was trained with a simulated student dataset that had been generated with a higher fidelity method. In our study, using a low fidelity model is still informative. We are less concerned with the exactness of P[learned] and are more interested in observing possible relative changes of P[learned] for certain groups of students, as different variations of the planner are tried on identical populations of simulated students.

The CFLS planner works as follows. For a given target learner the CFLS planner looks backward at the b most recent learning objects traversed. Then, it finds other learners who have traversed the same b learning objects with similar P[learned] values. These b LOs can be in any order, a simplification necessary to create a critical mass of similar learners. These are learners in the target learner's "neighbourhood". The planner then looks forward at the f next LOs traversed by each neighbour and picks the highest value path, where value is defined as the average P[learned] achieved on those f LOs ahead. This path is then recommended to the learner, who must follow it for at least s (for "sticky") LOs before replanning occurs. Of course, s is always less than f. In our research

we explored various values of b and f to find which leads to the best results (we set $f = s$ for this experiment). "Best results" can be defined many ways, but we focused on two measurements that were taken for each learner at the end of each simulation: the percentage of LOs mastered, and the score on a final exam. A LO is considered to be mastered when a score of P[learned] = 0.6 or greater is achieved. The score on the final exam is taken as the average P[learned] on the LOs that are the leafs of the prerequisite graph (interpreted as the ultimate target concept, which in the real world might well be final exams).

There is still a cold start problem even after the simulation has been initialized with the interaction data from the SPP. This is because the simulated learners who are to follow the CFLS planner have not yet viewed any LOs themselves as they begin the course, so there is no history to match the b LOs to create the plan. In this situation, the CFLS planner matches the learner with another arbitrary learner (from the interaction data from the SPP), and recommends whatever initial path that the other learner took when they first arrived in the course. While another solution to the cold start problem could be to start the new learner with the SPP, we did this to avoid any reliance whatsoever on knowing the underlying prerequisite structure.

The most computationally expensive part of the CFLS planner is finding the learners in the neighbourhood, which is at worst linear on the number of learners and linear on the amount of LO interaction history created by each learner. Each learner's LO interaction history must be searched to check for a match with b, with most learners being removed from the list during this process. The forward searching of f is then executed using only the small resulting dataset.

4 Results

We ran the CFLS planner 25 different times with all pairings of the values of b and s ranging from 1 to 5, using a population of 65 simulated learners. This population had the same distribution of aptitudes as the population used to generate the baseline interaction data described above. The heat maps in Figs. 1 and 2 show the measurements for each of the 25 simulations, for each aptitude group, with the highest relative scores coloured red, mid-range scores coloured white, and the lowest scores coloured blue. In general, simulated learners achieved higher scores when following the CFLS planner than when given LOs randomly. The CFLS planner even exceeded the SPP in many cases.

A success triangle is visible in the lower left of each aptitude group. The success triangles can be interpreted to mean that if a path is going to be recommended, never send the learner any further ahead (s) than you have matched them in the past (b). For example if a learner's neighbourhood was created using their $b = 2$ most recent LOs, then never make the learner follow in a neighbour's steps further than $s = 2$ LOs. One reason for the eventual drop at high values of b is that no neighbour could be found and a random match is used instead. However, the abrupt drop at $b > s$ was unexpected. To be sure the pattern was real, an extended series of simulations was run. We ran $b = 6$ and $s = 5$ to see

if there would be a drastic drop in performance, and indeed this was the case. We also ran another row varying b with a fixed $s = 6$, and again found a drop at $b = 7$.

LOW				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
100	21.9	32.5	31	37.7
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
89.6	86	36.9	36.9	40.4
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
72.1	68.6	62	43.21	40.1
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
77.3	74.4	72.1	66.1	49.3
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
68.1	70.7	67.5	67.4	63.5

MEDIUM				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
100	50	45.8	42.2	44.1
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
100	100	42.9	40.3	38
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
100	99.4	98.6	42.4	43
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
100	99.3	99.5	99.4	50.8
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
100	100	100	100	100

HIGH				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
100	75	39.3	39.7	41.1
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
100	100	41.7	34.9	40
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
100	100	100	42.5	50
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
100	100	100	100	61
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
100	100	100	100	100

Fig. 1. Average % Learning Objects Mastered by aptitude group

LOW				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
0.6587	0.1036	0.1314	0.1283	0.146
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
0.5178	0.4387	0.1398	0.1248	0.1363
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
0.4051	0.266	0.2256	0.1586	0.132
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
0.4138	0.2984	0.3016	0.2755	0.176
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
0.357	0.2884	0.2859	0.2679	0.2249

MEDIUM				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
0.6894	0.1851	0.2105	0.2099	0.2425
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
0.7004	0.698	0.2058	0.22	0.1972
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
0.6942	0.6761	0.6715	0.1944	0.2152
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
0.6931	0.6867	0.6874	0.6856	0.2292
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
0.6912	0.6884	0.6924	0.6965	0.6899

HIGH				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
0.7641	0.2514	0.2805	0.2866	0.2702
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
0.77	0.7694	0.2673	0.2738	0.2748
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
0.7653	0.7638	0.7727	0.3019	0.3097
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
0.768	0.7697	0.7633	0.7697	0.3431
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
0.7601	0.7612	0.7591	0.7644	0.7636

Fig. 2. Average Score on Final Exam (P[learned]) by aptitude group

A hot spot of successful combinations of b and s appeared for each aptitude group. For low aptitude learners, it was best to only match on the $b = 1$ most recent learning objects, and to follow the selected neighbour for only $s = 1$ LOs ahead before replanning. This combination of b and s is the only time when the CFLS planner outperformed the SPP for the low aptitude group. However, for the medium and high aptitude groups, the CFLS planner outperformed the SPP in all cases within the success triangle. Looking at final exam scores (Fig. 2), medium aptitude learners responded well to being matched with neighbours using $b = 1$ or 2 and sticking with the chosen neighbour for the same distance ahead. The high aptitude group responded very well to using neighbourhoods created with $b = 3$ and recommending paths of $s = 3$.

Within the success triangles, the rows and columns of Fig. 2 were checked to see if there existed an ideal b for a given s, and vice versa. Wherever there appeared to be a large difference, Student's t-test was used to check for statistical significance. We are able to use paired t-tests because the simulated learners have exactly the same characteristics in all the simulation runs, the only difference being the order in which LOs were interacted with. For example, learner #3 always has *aptitude-of-learner* = .4, so, there is no difference in that learner

between simulation runs. We used a two-tailed t-test because it was not certain whether one distribution was going to be higher or lower than the other.

Looking along the rows, when s is held the same, there are some cases where one value of b is better than another. For the low aptitude group, for the most part the lower the b, the better. For the medium aptitude group, there were no significant advantages to changing b. For the high aptitude group, when $s = 3$, the t-test was used to check if $b = 3$ was significantly more advantageous than using $b = 2$. The measurements for Score on the Final Exam for the high aptitude learners were compared between both simulation results, ($b = 2$ and $s = 3$) and ($b = 3$ and $s = 3$). With N=19 learners in this group, the calculated p-value was 0.009, indeed a statistically significant difference.

Looking along the columns, when b is held the same there was a case where increasing s, i.e. sticking to a longer plan ahead, was statistically advantageous. In the medium aptitude group, when $b = 1$ it was statistically better to use $s = 2$ than to use $s = 1$ with a p-value of 0.011. None of the increases of s with the same b were significant for the high aptitude group, and there were no increases for the low aptitude group.

5 Analysis and Future Work

Through simulation, we have shown that a CFLS planner can be "launched" from an environment that has been conditioned with interaction data from another planner, such as an SPP, and operate successfully using only learner usage data kept by the EA and not needing centralized metadata such as a prerequisite graph. This is one of the key requirements for DOELEs. Like biological evolution, the EA is harsh in that it observes how learners succeed or fail as various paths are tried. Successful paths for particular types of learners, regardless of whether they follow standard prerequisites, is the only criterion of success. New learners or new learning objects will find their niche - some paths will work for some learners but not for others, and this is discovered automatically through usage.

More experiments are needed to explore the many possibilities of the simulation environment. While this experiment was not a true test of a DOELE because new learners and LOs were not inserted, this can be readily explored in future work. New additions could be matched randomly a few times in order to build enough data in the EA, and then automatically incorporated into neighbourhood matches or into future plans.

Given the evaluation function that was selected, we found that planning ahead and sticking to the plan worked best for high aptitude learners and a reactive approach (planning ahead but sticking to the plan for only a short time) worked best for the low aptitude learners. Would a different pattern emerge if a different evaluation function were chosen? Would a different threshold for mastery than P[learned] > 0.6 make any difference? In future work, would it be worthwhile to break down the aptitude groups into six: very-high, high, medium-high, medium-low, low, and very-low? This may assist with more easily tuning the weights of the evaluation function, as there was not much difference in our

results between the high and medium aptitude groups. In addition, more experiments where $s < f$ are needed to answer the question of whether the drop along the edge of each success triangle was because of s or f. Also, in this work we did not look at the many different types of pedagogical interactions (ex. asking the student a question, giving a hint etc.) and focused on very abstract representations. More work is needed to explore this approach on systems later in the design process, when more detail about the content and the desired interactions with learners is known.

Future work could also investigate the usage of a differential planner, where different settings are tuned for different situations. For example, when creating a neighbourhood for a low aptitude learner, medium aptitude learners could be allowed into the neighbourhood if they have a matching b. Results could reveal situations where for example a low aptitude learner is helped by following in the steps of a medium aptitude learner. A differential planner could also dynamically choose the values of b and s for a given individual instead of using the same values for everyone at all times. For example, in a real world setting a CFLS planner may try to create a plan using a neighbourhood of $b = 3$, knowing it is optimal, but if for the specific case there is not enough data, it could change to $b = 2$ on the fly. Other aspects that could be changed are the criteria for creating the neighbourhood: rather than filtering by aptitude, another attribute could be chosen such as click behaviour or learning goals.

6 Conclusion

In this paper, we have described the need for instructional planning in DOELEs with many LOs aimed at large numbers of learners. Instructional planners such as [13] use AI planning technology that is based on states, actions and events, which are difficult to infer from an unstructured online environment. In recent years, instructional planning has been replaced by instructional design approaches such as [3]. Advanced instructional planners from the 1990s, such as PEPE and TOBIE [16] can blend different teaching strategies to appropriate situations. We have shown that instructional planning can still be done in the less rigid courses envisioned by the EA architecture and likely to be commonplace in the future, using only learner usage data kept by the EA and not needing centralized metadata about the course.

We have shown a specific planning technique, the CFLS planner, that is appropriate for DOELEs, and how to experiment in this domain. The simulation experiment revealed the number of LOs from a target learner's recent browsing history should be used for creating a neighbourhood (b), a question that has also been investigated by other researchers, such as in [18]. We have also found recommendations for settings for how far ahead to plan (s and f) for different groups of learners, and identified questions for future work. As is the case with collaborative filtering and case-based approaches, the quality of the plans created is limited to the quality of LOs within the repository and the quality

of interactions that have previously occurred between learners and sequences of LOs.

The bottom-up discovery of prerequisite relationships has been investigated by others, such as [17]. When the need for centralized metadata about a course is discarded, and when the further step is taken that different paths can be found to work better for different learners, then a shift in thinking occurs. Each individual learner could effectively have a unique ideal (implicit) prerequisite graph. Whether or not a prerequisite relationship even exists between two LOs could vary from learner to learner. The notion of prerequisite can thus be viewed not only as a function of the content relationships, but also as a function of the individual learner.

Making recommendations of sequences has also been identified as a task in the recommender systems domain [9]. An approach such as a CFLS planner is a step in the direction of building recommender systems that can use sequence information to recommend sequences. This has also been accomplished with standards approaches such as [15]. Simulation with the EA provides another method for developing and testing such approaches.

Overall, the research we have done to date and the questions it raises, shows the value of exploring these complex issues using simulation. We were able to essentially generate some 25 different experiments exploring some issues in instructional planning, in a very short time when compared to what it would have taken to explore these same issues with real learners. Others have also used simulation for developing an educational planner, such as [10] for social assessment games. To be sure our simulation model was of low fidelity, but we suspect that there are some properties of the CFLS planner that we have uncovered that apply in the real world (the lower triangles seem to be very strong and consistent patterns). And, there are some very real issues that we can explore fairly quickly going forward that might reveal other strong patterns, as discussed. We believe that it isn't always necessary to have simulations with high cognitive fidelity (as in SimStudent [12]) to find out interesting things. Low fidelity simulations such as the ones we have used in this and our earlier work [6] (and those of [2]) have a role to play in AIED. Especially as we move into the huge questions of dynamic open-ended learning environments with thousands of learners and big privacy issues, the sharp minimalist modelling possible with low fidelity simulation should allow quick and safe experimentation without putting too many real learners at risk and without taking years to gain insights.

Acknowledgements
We would like to thank the Natural Sciences and Engineering Research Council of Canada for funding some aspects of this research.

References

[1] Cazella, S., Reategui, E., and Behar, P.: Recommendation of Learning Objects Applying Collaborative Filtering and Competencies. IFIP Advances in Information and Communication Technology, 324, pp 35-43 (2010)

[2] Champaign, J.: Peer-Based Intelligent Tutoring Systems: A Corpus-Oriented Approach. Ph.D. Thesis, University of Waterloo, Waterloo, Canada (2012)
[3] Drachsler, H., Hummel, H. and Koper, R.: Using Simulations to Evaluate the Effects of Recommender Systems for Learners in Informal Learning Networks. SIRTEL Workshop (Social Information Retrieval for Technology Enhanced Learning) at the 3rd EC-TEL (European Conf. on Technology Enhanced Learning) Maastricht, The Netherlands: CEUR-WS.org, online CEUR-WS.org/Vol-382/paper2.pdf (2008)
[4] Desmarais, M., and Pelczer, I.: On the Faithfulness of Simulated Student Performance Data. In de Baker, R.S.J. et al. (Eds.), Proc. of the 3rd Int. Conf. on Educ. Data Mining, pp 21-30. Pittsburg USA (2010)
[5] Elorriaga, J. and Fernández-Castro, I.: Using Case-Based Reasoning in Instructional Planning: Towards a Hybrid Self-improving Instructional Planner. Int. Journal of Artificial Intelligence in Educ., 11(4), pp 416-449 (2000)
[6] Erickson, G., Frost, S., Bateman, S., and McCalla, G.: Using the Ecological Approach to Create Simulations of Learning Environments. In Lane, H.C. et al. (Eds), Proc. of the 16th Int. Con. on AIED, pp 411-420. Memphis USA: Springer (2013)
[7] Garrido, A. and Onaindia, E.: Assembling Learning Objects for Personalized Learning: An AI Planning Perspective. Intelligent Systems, IEEE, 28(2), pp 64-73 March/April (2013)
[8] Hannafin, M.J.: Learning in Open-Ended Environments: Assumptions, Methods and Implications. Educational Technology, 34(8), pp 48-55 (1994)
[9] Herlocker, J., Konstan, J., Terveen, L., and Riedl, J.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems (TOIS) 22(1), pp 5-53 (2004)
[10] Laberge, S., Lenihan, T., Shabani, S., and Lin, F.: Multiagent Coordination for Planning and Enacting an Assessment Game. Workshop on MultiAgent System Based Learning Environments of Int. Tutoring Systems (ITS) Honolulu, USA (2014)
[11] Land, S.: Cognitive Requirements for Learning with Open-Ended Learning Environments. Deuce. Technology Research and Development, 48(3), pp 61-78 (2000)
[12] Matsuda, N., Cohen, W. and Koedinger, K.: Teaching the Teacher: Tutoring SimStudent Leads to More Effective Cognitive Tutor Authoring. Int. Journal of Artificial Intelligence in Educ., 25(1), pp 1-34 (2014)
[13] Matsuda, N., and VanLehn, K.: Decision Theoretic Instructional Planner for Intelligent Tutoring Systems. In B. du Boulay (Ed.), Workshop Proc. on Modelling Human Teaching Tactics and Strategies, ITS 2000 pp 72-83. (2000)
[14] McCalla, G.: The Ecological Approach to the Design of e-Learning Environments: Purpose-based Capture and Use of Information about Learners. Journal of Interactive Media in Educ., http://jime.open.ac.uk/jime/article/view/2004-7-mccalla (2004)
[15] Shen, L. and Shen, R.: Learning Content Recommendation Service Based on Simple Sequencing Specification. In Liu W et al. (Eds.) Advances in Web-Based Learning - ICWL 2004 3rd Int. Conf. Web-based Learning, LNCS 3143, pp 363-370. Beijing, China:Springer (2004)
[16] Vassileva, J. and Wasson, B.: Instructional Planning Approaches: from Tutoring Towards Free Learning. Proceedings of Euro-AIED'96, Lisbon, Portugal (1996)
[17] Vuong, A., Nixon, T., and Towle, B.: A Method for Finding Prerequisites Within a Curriculum. In Pechenizkiy, M. et al. (Eds.) , Proc. of the 4th Int. Con. on Educ. Data Mining, pp 211-216. Eindhoven, the Netherlands (2011)
[18] Zhang, Y., and Cao, J.: Personalized Recommendation Based on Behavior Sequence Similarity Measures. In Cao, L. et al. (Eds.) Int. Workshop on Behaviour and Social Informatics / Behaviour and Social Informatics and Computing (BSI/BSIC 2013), Gold Coast QLD Australia / Beijing China, LNCS 8178, pp 165-177 (2013)

Exploring the Issues in Simulating a Semi-Structured Learning Environment: the SimGrad Doctoral Program Design

David Edgar K. Lelei and Gordon McCalla

ARIES Laboratory, Department of Computer Science, University of Saskatchewan
davidedgar.lelei@usask.ca and mccalla@cs.usask.ca

Abstract. The help seeking and social integration needs of learners in a semi-structured learning environment require specific support. The design and use of educational technology has the potential to meet these needs. One difficulty in the development of such support systems is in their validation because of the length of time required for adequate testing. This paper explores the use of a simulated learning environment and simulated learners as a way of studying design validation issues of such support systems. The semi-structured learning environment we are investigating is a graduate school, with a focus on the doctoral program. We present a description of the steps we have taken in developing a simulation of a doctoral program. In the process, we illustrate some of the challenges in the design and development of simulated learning environments. Lastly, the expected contributions and our research plans going forward are described.

Keywords: Simulated learners, Simulated learning environment, Agent-based simulation, Help seeking, Doctoral learners, Multi-agent system.

1 Introduction

Artificial Intelligence in Education (AIED) is one of the research fields whose focus is the use of technology to support learners of all ages and across all domains[1]. Although, one shortcoming of AIED research is the limited research attention that very dynamic and semi-structured domains, such as a graduate school, have received. There is little research that investigates how technology can be used to help connect learners (help seeker and potential help givers) in the graduate school domain. Consequently, there is a gap in our understanding of how such technology may mitigate graduate learners' attrition rates and time-to-degree. We have suggested the use of reciprocal recommender technology to assist in the identification of a suitable helper [1]. However, the nature of graduate school means that validation of any education system designed to be used in a semi-structured environment would take a long time (measured in years). This paper aims to address this challenge by exploring the use of

[1] http://iaied.org/about/

simulated learning environment and simulated learners as a potential way of validating educational technologies designed to support doctoral learners.

In this paper, we first describe the nature and the metrics used by interested stakeholders to measure the success or lack therefore of a doctoral program. Following this, we briefly discuss the uses of simulation as it relates to learning environment. We then introduce the research questions we are interested in answering using simulation. We go on to describe the architectural design of our simulation model. Further, we show how data about the 'real world' target domain is used to inform the parameters and initial conditions for the simulation model. This provides the model with a degree of fidelity. Throughout this model development process, we illustrate some of the challenges in the design and development of simulated learning environments. We conclude the paper with a discussion of the expected contributions and our research plans going forward.

2 Understanding Doctoral Program

Graduate school is a very dynamic and complex social learning environment. A doctoral program in particular is a dynamic, semi-structured, and complex learning environment. Most doctoral programs have some structure in the sense that there are three distinct stages that doctoral learners must go through: admission stage, coursework stage, and dissertation stage. While coursework stage is fairly structured, the dissertation stage is not. Further, the dissertation stage have various milestones that include: comprehensive exam, thesis proposal, research, writing, and dissertation defense. As time passes, learners move from one stage to the next and their academic and social goals change. There is need for self-directed learning and individual doctoral learners are responsible for their own learning pace and choice of what to learn especially in the dissertation stage.

The dynamic nature of the program ensures that there is constant change; there are new learners joining the program, other learners leaving the program either through graduation or deciding to drop out, and still other learners proceeding from one stage to the next. There are two key aspects that influences learners to decide whether to persist or drop out of a learning institution: academic and social integration [2], [3] which are impacted by learner's initial characteristics and experiences during their duration in the program. The various stages of the doctoral program (e.g., coursework) and learning resources can be seen as factors that directly influence the academic integration of a doctoral learner. Peers and instructors/supervisors can be viewed as supporting the social aspects of the doctoral program and hence, directly impact the social integration of doctoral learners. As time passes, doctoral learners continually interact with both the academic and social facets of the doctoral program. As a result, there is constant change in learners' commitment to their academic goal and the social sides of the learning institution

Time-to-degree, completion rates, and attrition rates are important factors influencing the perception and experience of graduate education by interested stakeholders [4], [5]. Research on doctoral attrition and time-to-completion indicates that on aver-

age, the attrition rate is between 30% and 60% [5]–[8]. Long times to completion and a high attrition rate are costly in terms of money to the funding institution and the learning institution; and in terms of time and effort to the graduate student(s) and supervisor(s) [8]. Lack of both academic and social integration (isolation) have been shown to affect graduate learners decision to persist [2], [3], [9]. Learners facing academic and social integration challenges should be enabled to engage in a community of peers to foster interaction and hence, encourage peer help and personalized collaboration [10]. Understanding the nature of learner-institution interactions that foster doctoral learners' persistence to degree is important to both the learning institution and its learners. We use simulation to achieve this feat.

Simulation is an established third way of exploring research questions in addition to qualitative and quantitative methods [11], [12]. VanLehn [13] has identified three main uses of simulation in learning environments: 1) to provide an environment for human teachers to practise their teaching approaches; 2) to provide an environment for testing different pedagogical instructional design efforts; 3) to provide simulated learners who can act as companions for human learners. Our research is mainly focused on the first and the second uses – to enable deep insight into the complex interaction of the factors affecting doctoral learners' attrition and time-to-degree leading to a better design of an educational system. Therefore, our research questions are formulated around investigations of how various factors influence time-to-degree, completion rates, and dropout rates of doctoral students. We are interested in answering the following research questions:

1. How does the number of classes (as a platform for social integration with peers – potential helpers) offered by a program(s) or taken by a learner, influence learners' time-to-degree and their propensity to persist or drop out?
2. How does the average class size (as basis of learners' social integration) attended by learners, impact learners' time-to-degree and their inclination to persist or drop out? What is the optimum class size?
3. How does the overall population size of the learners (a few learners vs many learners) influence learners' time-to-degree and their likelihood to persist or drop out?
4. Does timely help affects doctoral learners' time-to-degree and their decision to persist or drop out? If so, how?
5. How does the level of reciprocation influence the formation of a 'helpful community' of learners and adaptive help seeking behavior of the learners?

Use of simulation enables us to explore the aforementioned issues in a fine-grained controlled environment. For example, it would be almost impossible in the 'real world' setting to examine the impact of different number of course to take or class size to attend. Two cohorts of learners will have different attributes. Simulation allows us to tweak the number of courses or class size without touching the other characteristics of learners. Hence, we are able to see the real impact of one variable at a time. Before any exploration and insight can be gained on these issues, there is need to design and implement the simulation model.

3 Building an Initial Prototype of *SimGrad*

In this section we demonstrate the steps we have taken in the development of our initial prototype of our simulated doctoral learning environment: *SimGrad*. We show how a designer of an educational technology can develop a model of their target learning environment and inform its initial condition with available 'real world' data.

3.1 *SimGrad* Design

We need to design a simulation model by addressing two key challenges. First, we need to consider issues related to the modeling of the learning environment: how do we design conceptual and computational models of a doctoral program and what stakeholders should be included in these models? The second concern is about modeling of simulated learners: what doctoral learners' features affect persistence and time-to-degree, what factors do we model, and can we inform these features with available 'real world' data?

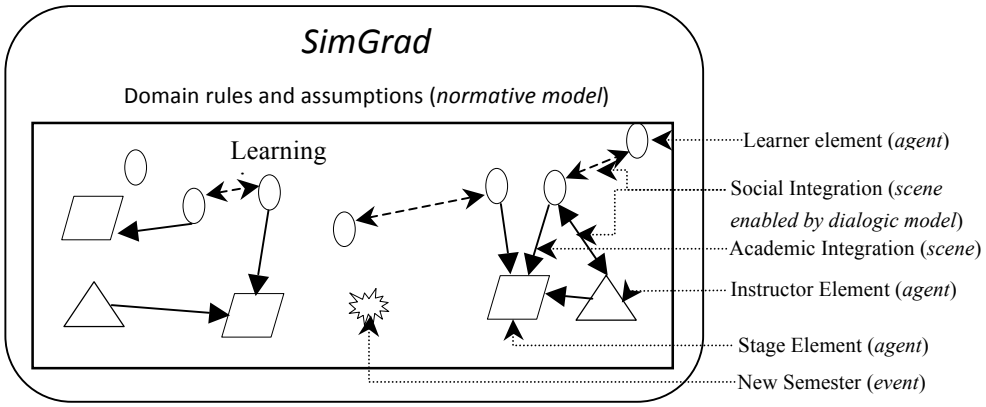

Fig. 1. SimGrad conceptual framework, its three elements, and the possible interaction between the elements

We have designed our conceptual model of the different aspects of simulated doctoral learners and doctoral learning environment based on the simulated learning environment specifications suggested by Koper et al. in [14], and features for building an electronic institution proposed by Esteva et al. [15]. We name our conceptual framework, *SimGrad*. Its core elements include: *normative model* - specifies requirements and constraints to guide agent actions and behavior; *dialogic model* – deals with interaction strategies and communication mechanism; *events* – refers to happenings in the model that trigger (re)action by agents; *scene* – description of an interaction between elements; *elements (agents)* – represent key stakeholders of the target domain that are modeled. Elements are modeled as agents. Each of the agents has attributes and behavior which are informed by our assumptions guided by our research questions and factors that influence learners' decision to persist. Every element of interest is to be

modeled within the learning environment and all possible interactions and operations within the learning setting is guided by domain rules represented by the normative model. See **Fig. 1**.

In our simulation model, we have chosen to model three types of elements: class, instructor, and learner. In this paper, in keeping with model simplicity, both the class and the instructor agents are passive while the learner agent is modeled to be active and reactive to its environment. Also, the only instructor's attributes we are interested in are related to classes (see **Table 1**). We modeled only one type of instructor agent. Another instructor type agent that can be modeled is the supervisor.

Each learner agent has the following properties: autonomy, social ability, reactivity, proactivity, and a degree of intentionality. We have also identified the following key attributes for our agent learner model: state – (busy, available), program, stages, course taken, peer interactions (pertaining challenges), academic integration, social integration, and motivation (see **Table 2**). In our model, peer interaction and state contribute to a learners' social integration, while research area, stage, course taken impact to their academic integration. Motivation combines both the social and academic integration and hence, is the main factor that determines whether an agent continues to persist or chooses to drop out of the program.

Table 1. Comparison of computed attributes of the three agent types

Attribute – data (value range)	*Agent learner*	*Agent instructor*	*Agent class*
Total number of classes take, taught, or frequency of offering within 10 years – *numeric (0-20)*	X	X	X
Grade obtained, average awarded, or average obtained by learners – *numeric (0,12)*	X	X	X
Take classes from, teach classes in, or class offered in various programs – *textual (program id)*	X	X	X
Instructors teaching a class – *array list (instructor id)*	X	-	X
What is the class size – *numeric (1-5)*	X		X
Number of classes taken or taught per year - *numeric (0,4)*	X	X	-
Which classes are taken or taught – *textual (class id)*	X	X	-

The main intentions of each agent is to persist through doctoral requirements to graduation and to do so in a timely manner. However, each of these agents reacts to the different challenges at various stages of graduate school in divergent and autonomous ways. At the coursework stage, agents have the goal of taking courses that are relevant to their field and that they will perform well. When facing a course choice challenge or any other particular challenge, we have modeled our agents to proactively associate with peers to seek help. Each peer makes individual choice on whether to or not to respond to a request for help from others. The dialogic model

handles the agent to agent interaction and communication through a message passing mechanism [16].

Table 2. Attributes and parameters considered for an agent learner model for learners, their description and how each of them changes.

Attribute	*Value - description*	*How it changes*
Enrolment	Date (MM/YYYY) Indicate the month a year an agent enrolled in the program	Does not change
Graduation	Date (MM/YYYY) Target graduation date	Evaluated whenever an agent completes a milestone
State	Textual (busy, available) Indicates an agent availability to help others, assigned based on the smallest time unit model	Changes whenever an agent experiences a challenge
Program	Textual (program id) Identify an agent's closer community within the larger community of learners	Does not change during a simulation run
Stage	Textual (admission, coursework, dissertation, timeout, dropout)	Admission stage is like an event. Learner move to the coursework immediately after admission. They more to dissertation after completing their course load.
Courses taken	Array [course, mark, instructor id](0-6) Record courses taken by an agent and the marks obtain in each course	Every end of semester that the student took classes, this array is updated
Peer interaction	Array [learner id, challenge, result], Keep track of an agent interactions with others and the outcome of the interaction	Changes whenever two agents interact
Academic integration	Numeric (-1,1) Measures the academic satisfaction	Changes whenever an agent learner interacts with agent stage (i.e., completes a milestone or experience a challenge)
Social integration	Numeric (-1,1) Measures a learners sense of belonging to the learning environment	Changes whenever an agent learner interacts with its peers or agent instructors
Motivation	Numeric (-1,1) Measures the propensity of an agent to still want to persist. A motivation value above 0.3 indicates persistence. A value between -0.3 and 0.3 indicate help seeking needed. A value below -0.3 means the agent drops out	Whenever there is a change in the social and academic integration values. Its value is the average of the integration values.

3.2 Informing *SimGrad* behavior and evaluation functions

Having identified the important agents and their key attributes, there are two sets of important functions for each element that need to be modelled: behaviour functions and evaluation functions [17]. Behaviour functions inform the decision making of the active elements and dictates the interaction patterns between them and the other modeled elements (e.g., how many classes a given agent takes). Evaluation functions indicate whether or not various interactions between the different agents in a simulation were successful (e.g., determine what grade a given agent attains in a class it took). Informing such functions with 'real world' data allows the simulation to behave in a way consistent with reality.

Simulation model fidelity is an issues that might arise when using simulation to study a target real world phenomenon. However, the most important issue to consider is the research question to be answered. While Champaign [18] used a very low fidelity model, Matsuda et al. [19] used a model with high cognitive fidelity to reach compelling conclusion. Further yet, Erickson et al. [17] also demonstrated that is possible to use a medium fidelity model and uncover interesting results. In some situations it might not be possible to have a high fidelity model because of lack of data. A case in point is our simulation scenario. Where possible, we inform our simulation functions with data received from the U of S on their doctoral program. An investigation into the U of S data showed that we will not be able to inform every aspect of our simulation model. It would be desirable to inform every initial aspects of our simulation model with 'real world' data but, we do not have data on the dissertation stage.

We are provided information on student id, years a student is registered, year of graduation (if graduated), student's program, classes taken and marks obtained, class instructor, and students instructional responsibilities. From this dataset we are able to inform the admission and coursework stages of our model (academic integration). However, there is no information concerning the dissertation stage and the social integration aspects. While it possible to inform various behaviour and evaluation functions for our simulation model, in this paper we focus on describing the steps we took to inform two functions of our simulation: learning environment admission behaviour function, and learners' class interactions behaviour function.

As already mentioned, admission is an important part of a doctoral program that contributes to it dynamic nature. The admission process is complex and involves a lot of stakeholders and processes, but we are concerned only with determining the year to year patterns in how many students are admitted. To provide some fidelity to our simulated learning environment admission, we analyzed data provided to us by the U of S University Data Warehouse[2]. The provided dataset contained information on doctoral learners registered in the 10 years 2005-2014. In this time there were 2291 doctoral learners with a total of 52850 data points on class registration. The 2005 registration included learners who had joined the program earlier than 2005. In order to get a clean admission pattern, we only considered learners who were registered from the year 2006 onwards. This reduced the population size to 1962.

[2] http://www.usask.ca/ict/services/ent-business-intelligence/university-data-warehouse.php

We were able to identify three admission periods, September, January, and May. We then obtained values for each of the admissions months for the years 2006-2014. This provided a distribution for each month that we used to generate a scatter plot of admission numbers. A sigmoidal pattern emerged. Next, we performed a non-linear curve fitting to the scatter plot so that the admission function can be represented in the form $Y = St*(c + x)$, where c is a constant, St is a variable dependent on the admission period, and x is the admission period. We then ran a regression to find values of each of these variables. This allowed us to model the admission patterns observed in the U of S dataset.

Next we derived the number of classes taken. To introduce some realism to the number classes taken behaviour, we had to further prune the data. We only considered data for students whose cohorts would have been registered for at least 3 years by the end of the year 2014 and hence, we considered class taking behaviour of 1466 U of S doctoral learners.

We obtained the number of classes each of the remaining learners we registered in and created a histogram. This histogram showed us the distribution of the number of students registered for a certain number of classes. Next, we transformed this distribution graph into a cumulative distribution function. We then took an inverse of the cumulative distribution function to achieve a quantile function. The quantile function, when run over many learners, assigns learners a class count that mimics the initial histogram. We use this quantile function to inform the number of classes a learner can take.

In this section we have described the importance of informing a simulation model with 'real world' data. We have described two functions that are informed with U of S dataset. Other examples of functions that can be informed using the U of S dataset include: class performance evaluation function, dropout behaviour function, time to degree behaviour function, and flow through behavior function (main as pertains to coursework stage). We have identified that missing data values is a major hindrance in this endeavor. There are possible ways of informing simulation attributes where there are no 'real world' data to derive from. A designer can either assign common sense values, generate and assign random values, or refer to the research literature to identify patterns that have been found by other researchers. Since we have the enrolment dates and the graduate dates for learners who graduate, we choose to derive common sense values with these two dates guiding the process and the value range.

4 Discussion, Expected Contributions, and Future Research Plans

Despite the growth in the use of simulation as a method for exploration and learning in many areas such as: engineering, nursing, medicine [20], and building design [21], research in the used of simulation within AIED is still at an early stage. There is need for more research to demonstrate that the outputs of simulation runs are desirable and informative to the AIED community. In this paper, we aim at contributing to this notion and by promoting the use of simulation in educational research and presenting

an agent based simulation conceptual framework for building simulated learning environment, with a focus on the semi-structured ones. Simulated learning environment and simulated learners are important in exploring and understanding a given learning domain. Further, it helps with the generation of system validation data.

The expected contributions to AIED include: providing a conceptual framework for simulated graduate school learning environment – an architecture that enables investigations into factors affecting doctoral learners progress through their program; shedding light on learner modeling issues in dynamic learning environments; and demonstrating the importance of simulation in exploring various AIED research domains, particularly semi-structured domains.

Current research work is focused on the implementation of the simulation model and the refinement of the various behaviour and evaluation functions. Once the implementation is done, we will validate our model against the dataset we have from the U of S before proceeding to explore the impact of various environmental factors. Since we are informing the simulation with both common sense assumptions and U of S dataset, the goal is to tweak the common sense assumptions such that when the model is run we get similar results as the U of S data in terms of class performance, dropout rate, and time-to-degree. Achieving this, would give us confidence that we have captured reality in some measurable way. We can then start exploring the various impact of measures we are interested in examining. As earlier indicated, we are interested in exploring the interactions of a number of variables: number of classes taken which will impact the availability of potential peer helpers, the effect of reciprocity on help seeking and help giving, and the effect of help seeking and other factors on doctoral learners' time-to-degree and attrition rates.

Acknowledgement

We would like to thank University of Saskatchewan University Data Warehouse team for giving us access to 10 year dataset. Specifically, we would like to thank Mathew Zip for processing and explaining to the first author the nature of the dataset. We also wish to acknowledge and thank the Natural Science and Engineering Research Council of Canada (NSERC) for funding our research.

References

[1] D. E. K. Lelei, "Supporting Lifelong Learning: Recommending Personalized Sources of Assistance to Graduate Students," in *Artificial Intelligence in Education*, 2013, pp. 912–915.

[2] V. Tinto, "Taking student success seriously: Rethinking the first year of college.," *Ninth Annu. Intersession Acad. Aff. Forum,* vol. 19, no. 2, pp. 1–8, 2005.

[3] V. Tinto, "Dropout from higher education: A theoretical synthesis of recent research," *Rev. Educ. Res.*, vol. 45, no. 1, pp. 89–125, 1975.

[4] H. Groenvynck, K. Vandevelde, and R. Van Rossem, "The Ph.D. Track: Who Succeeds, Who Drops Out?," *Res. Eval.*, vol. 22, no. 4, pp. 199–209, 2013.

[5] F. J. Elgar, “Phd Degree Completion in Canadian Universities,” Nova Scotia, Canada, 2003.
[6] M. Walpole, N. W. Burton, K. Kanyi, and A. Jackenthal, “Selecting Successful Graduate Students: In-Depth Interviews With GRE Users,” Princeton, NJ, 2002.
[7] L. Declou, “Linking Levels to Understand Graduate Student Attrition in Canada,” McMaster University, Hamilton, Ontario, Canada, 2014.
[8] B. E. Lovitts, *Leaving the Ivory Tower: The Causes and Consequences of Departure from Doctoral Study*, Illustrate., vol. 32. Rowman & Littlefield Publishers, 2001, p. 307.
[9] A. Ali and F. Kohun, “Dealing with isolation feelings in IS doctoral programs,” *Int. J. Dr. Stud.*, vol. 1, no. 1, pp. 21–33, 2006.
[10] Computing Research Association, “Grand research challenges in information systems,” Washington, D.C, 2003.
[11] R. Axelrod, “Advancing the Art of Simulation in the Social Sciences,” in *Proceedings of the 18th European Meeting on Cybernetics and Systems Research*, 2006, pp. 1–13.
[12] N. Gilbert and K. G. Troitzsch, “Simulation and Social Science,” in *Simulation for the Social Scientist*, McGraw-Hill International, 2005, pp. 1–14.
[13] K. VanLehn, S. Ohlsson, and R. Nason, “Applications of Simulated Students: An Exploration,” *J. Artif. Intell. Educ.*, vol. 5, no. 2, pp. 1–42, 1994.
[14] R. Koper and B. Olivier, “Representing the Learning Design of Units of Learning,” *Educ. Technol. Soc.*, vol. 7, no. 3, pp. 97–111, 2003.
[15] M. Esteva, J.-A. Rodriquez-Aguilar, C. Sierra, P. Garcia, and J. L. Arcos, “On the Formal Specification of Electronic Institutions,” in *Agent Mediated Eectronic Commerce*, 2001, pp. 126–147.
[16] H. J. C. Berendsen, D. Van Der Spoel, and R. Van Drunen, “GROMACS: A message-passing parallel molecular dynamics implementation,” *Comput. Phys. Commun.*, vol. 91, no. 1, pp. 43–56, 1995.
[17] G. Erickson, S. Frost, S. Bateman, and G. Mccalla, “Using the Ecological Approach to Create Simulations of Learning Environments,” in *In Artificial Intelligence in Education*, 2013, pp. 411–420.
[18] J. Champaign, “Peer-Based Intelligent Tutoring Systems: A Corpus-Oriented Approach,” University of Waterloo, 2012.
[19] N. Matsuda, W. W. Cohen, K. R. Koedinger, V. Keiser, R. Raizada, E. Yarzebinski, S. P. Watson, and G. Stylianides, “Studying the Effect of Tutor Learning Using a Teachable Agent that Asks the Student Tutor for Explanations,” in *2012 IEEE Fourth International Conference On Digital Game And Intelligent Toy Enhanced Learning*, 2012, pp. 25–32.
[20] A. L. Baylor and Y. Kim, “Simulating Instructional Roles Through Pedagogical Agents,” *Int. J. Artif. Intell. Educ.*, vol. 15, no. 2, pp. 95–115, 2005.
[21] G. Augenbroe, “Trends in Building Simulation,” *Build. Environ.*, vol. 37, no. 8, pp. 891–902, 2002.

Exploring the Role of Small Differences in Predictive Accuracy using Simulated Data

Juraj Nižnan, Jan Papoušek, and Radek Pelánek

Faculty of Informatics, Masaryk University Brno
{niznan,jan.papousek,xpelanek}@mail.muni.cz

Abstract. Research in student modeling often leads to only small improvements in predictive accuracy of models. The importance of such improvements is often hard to assess and has been a frequent subject of discussions in student modeling community. In this work we use simulated students to study the role of small differences in predictive accuracy. We study the impact of such differences on behavior of adaptive educational systems and relation to interpretation of model parameters. We also point out a feedback loop between student models and data used for their evaluation and show how this feedback loop may mask important differences between models.

1 Introduction

In student modeling we mostly evaluate models based on the quality of their predictions of student answers as expressed by some performance metric. Results of evaluation often lead to small differences in predictive accuracy, which leads some researchers to question the importance of model improvements and meaningfulness of such results [1]. Aim of this paper is to explore the impact and meaning of small differences in predictive accuracy with the use simulated data. For our discussion and experiments in this work we use a single performance metric – Root Mean Square Error (RMSE), which is a common choice (for rationale and overview of other possible metrics see [15]). The studied questions and overall approach are not specific to this metric.

Simulated students provide a good way to study methodological issues in student modeling. When we work with real data, we can use only proxy methods (e.g., metrics like RMSE) to evaluate quality of models. With simulated data we know the "ground truth" so we can study the link between metrics and the true quality of models. This enables us to obtain interesting insight which may be useful for interpretation of results over real data and for devising experiments. Similar issues are studied and explored using simulation in the field of recommender systems [7, 17].

We use a simple setting for simulated experiments, which is based on an abstraction of a real system for learning geography [12]. We simulate an adaptive question answering system, where we assume items with normally distributed difficulties, students with normally distributed skills, and probability of correct

answer given by a logistic function of the difference between skill and difficulty (variant of a Rasch model). We use this setting to study several interrelated question.

1.1 Impact on Student Practice

What is the impact of prediction accuracy (as measured by RMSE) on the behavior of an adaptive educational system and students' learning experience?

Impact of small differences in predictive performance on student under-practice and over-practice (7-20%) has been demonstrated using real student data [18], but insight from a single study is limited. The relation of RMSE to practical system behavior has been analyzed also in the field of recommender systems [2] (using offline analysis of real data). This issue has been studied before using simulated data is several studies [5, 6, 10, 13]. All of these studies use very similar setting – they use Bayesian Knowledge Tracing (BKT) or its extensions and their focus is on mastery learning and student under-practice and over-practice. They differ only in specific aspects, e.g., focus on setting thresholds for mastery learning [5] or relation of moment of learning to performance metrics [13]. In our previous work [16] have performed similar kind of simulated experiments (analysis of under-practice and over-practice) both with BKT and with student models using logistic function and continuous skill.

In this work we complement these studies by performing simulated experiments in slightly different setting. Instead of using BKT and mastery learning, we use (variants of) the Rasch model and adaptive question answering setting. We study different models and the relation between their prediction accuracy and the set of items used by the system.

1.2 Prediction Accuracy and Model Parameters

Can RMSE be used to identify good model parameters? What is the relation of RMSE to the quality of model parameters?

In student modeling we often want to use interpretable models since we are interested not only in predictions of future answers, but also in reconstructing properties of students and educational domains. Such outputs can be used to improve educational systems as was done for example by Koedinger at al. [9]. When model evaluation shows that model A achieves better prediction accuracy (RMSE) then model B, results are often interpreted as evidence that model A better reflects "reality". Is RMSE a suitable way to find robust parameters? What differences in metric value are meaningful, i.e., when we can be reasonably sure that the better model really models reality in better way? Is statistical significance of differences enough? In case of real data it is hard to answer these question since we have no direct way to evaluate the relation of a model to reality. However, we can study these questions with simulated data, where we have access to the ground truth parameters. Specifically, in our experiments we study the relation of metric values with the accuracy of reconstructing the mapping between items and knowledge components.

1.3 Feedback between Data Collection and Evaluation

Can the feedback loop between student models and adaptive choice of items influence evaluation of student models?

We also propose novel use of simulated students to study a feedback loop between student models and data collection. The data that are used for model evaluation are often collected by a system which uses some student model for adaptive choice of items. The same model is often used for data collection and during model evaluation. Such evaluation may be biased – it can happen that the used model does not collect data that would show its deficiencies. Note that the presence of this feedback loop is an important difference compared to other forecasting domains. For example in weather forecasting models do not directly influence the system and cannot distort collected data. In student modeling they can.

So far this feedback has not been thoroughly studied in student modeling. Some issues related to this feedback have been discussed in previous work on learning curves [6, 11, 8]. When a tutoring system uses mastery learning, students with high skill drop out earlier from the system (and thus from the collected data), thus a straightforward interpretation of aggregated learning curves may be misleading. In this work we report experiment with simulated data which illustrate possible impact of this feedback loop on model evaluation.

2 Methodology

For our experiments we use a simulation of a simplified version of an adaptive question answering systems, inspired by our widely used application for learning geography [12]. Fig. 1 presents the overall setting of our experiments. System asks students about items, answers are dichotomous (correct/incorrect), each student answers each item at most once. System tries to present items of suitable difficulty. In evaluation we study both the prediction accuracy of models and also sets of used items. This setting is closely related to item response theory and computerized adaptive testing, specifically to simulated experiments with Elo-type algorithm reported by Doebler et al. [3].

Simulated Students and Items We consider a set of simulated students and simulated items. To generate student answers we use logistic function (basically the Rasch model, respectively one parameter model from item response theory): $P(correct|\theta_s, d_i) = 1/(1 + e^{-(\theta_s - d_i)})$, where θ_s is the skill of a student s and d_i is difficulty of an item i.

To make the simulated scenarios more interesting we also consider multiple knowledge components. Items are divided into disjoint knowledge components and students have different skill for each knowledge component. Student skills and item difficulties are sampled from a normal distribution. Skills for individual knowledge components are independent from one another.

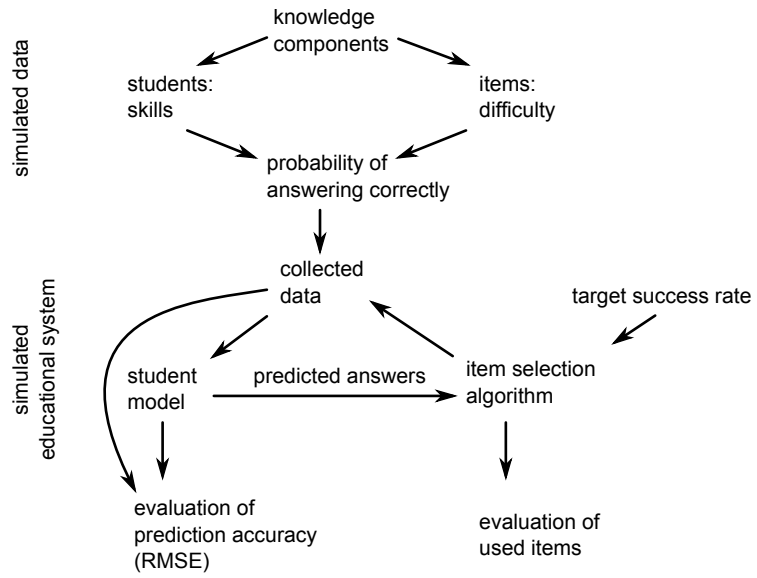

Fig. 1. Setting of our experiments

Item Selection Algorithm The item selection algorithm has as a parameter a target success rate t. It repeatedly presents items to a (simulated) student, in each step it selects an item which has the best score with respect to the distance of the predicted probability of correct answer p and the target rate t (illustrated by gray dashed line in Fig. 3). If there are multiple items with the same score, the algorithm randomly selects one of them.

Student Models Predictions used by the item selection algorithm are provided by a student model. For comparison we consider several simple student models:

- Optimal model – Predicts the exact probability that is used to generate the answer (i.e., a "cheating" model that has access to the ground truth student skill and item difficulty).
- Optimal with noise – Optimal model with added (Gaussian) noise to the difference $\theta_s - d_i$ (before we apply logistic function).
- Constant model – For all students and items it provides the same prediction (i.e., with this model the item selection algorithm selects items randomly).
- Naive model – Predicts the average accuracy for each item.
- Elo model – The Elo rating system [4, 14] with single skill. The used model corresponds to the version of the system as described in [12] (with slightly modified uncertainty function).
- Elo concepts – The Elo system with multiple skills with correct mapping of items to knowledge components.

– Elo wrong concepts – The Elo system with multiple skills with wrong mapping of items to knowledge components. The wrong mapping is the same as the correct one, but 50 (randomly choosen) items are classified incorrectly.

Data We generated 5,000 students and 200 items. Items are divided into 2 knowledge components, each user has 2 skills corresponding to the knowledge components and each item has a difficulty. Both skills and difficulties were sampled from standard normal distribution (the data collected from the geography application suggests that these parameters are approximately normally distributed). The number of items in a practice session is set to 50 unless otherwise noted.

3 Experiments

We report three types of experiments, which correspond to the three types of questions mentioned in the introduction.

3.1 Impact on Student Practice

Our first set of experiments studies differences in the behavior of the simulated system for different models. For the evaluation of model impact we compare the sets of items selected by the item selection algorithm. We make the assumption that the algorithm for item selection using the optimal model generates also the optimal practice for students. For each user we simulate practice of 50 items (each item is practiced at most once by each student). To compare the set of practiced items between those generated by the optimal model and other models we look at the size of the intersection. We assume that bigger intersection with the set of practiced items using the optimal model indicates better practice. Since the intersection is computed per user, we take the mean.

This is, of course, only a simplified measure of item quality. It is possible that an alternative model selects completely different set of items (i.e., the intersection with the optimal set is empty) and yet the items are very similar and their pedagogical contribution is nearly the same. However, for the current work this is not probable since we are choosing 50 items from a pool of only 200 items. For future work it would be interesting to try to formalize and study the "utility" of items.

Noise Experiment The optimal model with noise allows us to easily manipulate differences in predictive accuracy and study their impact on system behavior. Experiment reported in the left side of Fig. 2 shows both the predictive accuracy (measured by RMSE) and the impact on system behavior (measured by the size of the intersection with the optimal practiced set as described above) depending on the size of noise (we use Gaussian noise with a specified standard deviation). The impact of noise on RMSE is approximately quadratic and has a slow rise –

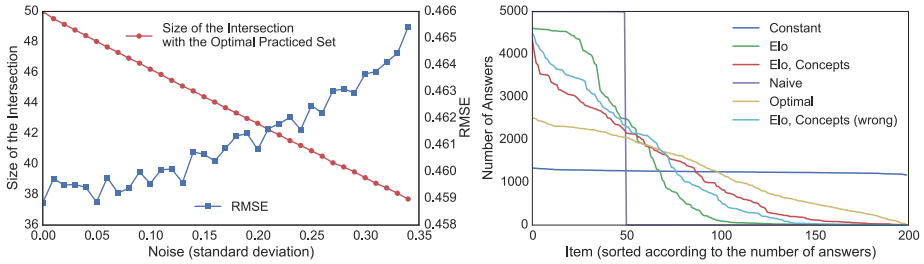

Fig. 2. Size of the intersection with the optimal practiced set of items and RMSE depending on Guassian noise in optimal model (left side). Distribution of answers over the items based on the given model (right side).

this is a direct consequence of the quadratic nature of the metric. The impact on used items is, however, approximately linear and rather steep. The most interesting part is for noise values in the interval $[0, 0.1]$. In this interval the rise in RMSE values is very small and unstable, but the impact on used items is already high.

Model Comparison Right side of the Fig. 2 shows the distribution of the number of answers per item for different models. The used models have similar predictive accuracy (specific values depend on what data we use for their evaluation, as discussed below in Section 3.3), yet the used model can dramatically change the form of the collected data.

When we use the optimal model, the collected data set covers almost fairly most items from the item pool. In the case of worse models the use of items is skewed (some items are used much more frequently than others). Obvious exception is the constant model for which the practice is completely random. The size of the intersection with the optimal practiced set for these models is – Constant: 12.5; Elo: 24.2; Elo, Concepts: 30.4; Elo, Concepts (wrong): 28.5; Naive: 12.0. Fig. 3 presents a distribution of answers according to the true probability of their correctness (given by the optimal model). Again there is a huge difference among the given models, especially between simple models and those based on Elo.

3.2 Prediction Accuracy and Model Parameters

Metrics of prediction accuracy (e.g., RMSE) are often used for model selection. Model that achieves lower RMSE is assumed to have better parameters (or more generally better "correspondence to reality"). Parameters of a selected model are often interpreted or taken into account in improvement of educational systems. We checked validity of this approach using experiments with knowledge components.

We take several models with different (random) mappings of items to knowledge components and evaluate their predictive accuracy. We also measure the

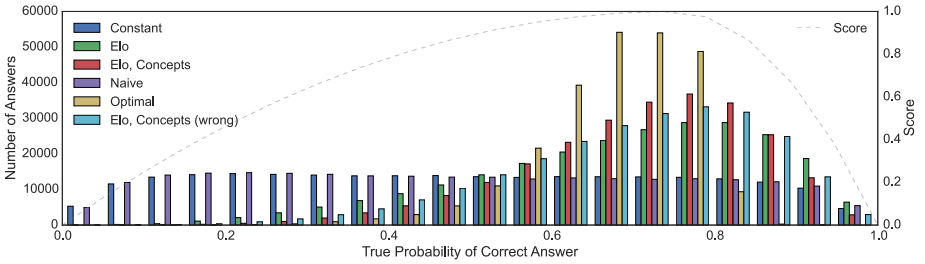

Fig. 3. Distribution of answers according to the true probability of correct answer. The gray dashed line stands for the score function used by the algorithm for item selection.

quality of the used mappings – since we use simulated data, we know the ground truth mapping and thus can directly measure the quality of each mapping. Quality is expressed as the portion of items for which the mapping agrees with the ground truth mapping. The names of the knowledge components are irrelevant in this setting. Therefore, we compute quality for each one-to-one mapping from the names of the components in the model to the names of the components in the ground truth. We select the highest quality as the quality of the model's item-to-component mapping. To focus only on quality of knowledge components, we simplify other aspects of evaluation, specifically each student answers all items and their order is selected randomly.

These experiments do not show any specific surprising result, so we provide only general summary. Experiments show that RMSE values correlate well with the quality of mappings. In case of small RMSE differences there may be "swaps", i.e., a model with slightly higher RMSE reflects reality slightly better. But such results occur only with insufficiently large data and are unstable. Whenever the differences in RMSE are statistically significant (as determined by t-test over different test sets), even very small differences in RMSE correspond to improvement in the quality of the used mappings. These results thus confirm that it is valid (at least in the studied setting) to argue that a model A better corresponds to reality than a model B based on the fact that the model A achieves better RMSE than the model B (as long as the difference is statistically significant). It may be useful to perform this kind of analysis for different settings and different performance metrics.

3.3 Feedback between Data Collection and Evaluation

To study feedback between the used student model and collected data (as is described in subsection 1.3) we performed the following experiment: We choose one student model and use it as an input for adaptive choice of items. At the same time we let all other models do predictions as well and log answers together with all predictions.

Fig. 4 shows the resulting RMSE for each model in individual runs (data collected using specific model). The figure shows several interesting results. When

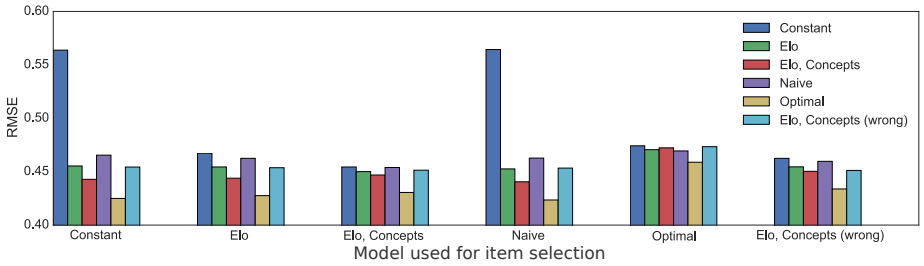

Fig. 4. RMSE comparison over data collected using different models.

the data are collected using the optimal model, the RMSE values are largest and closest together; even the ordering of models is different from other cases. In this case even the constant model provides comparable performance to other models – but it would be very wrong to conclude that "predictive accuracy of models is so similar that the choice of model does not matter". As the above presented analysis shows, different models lead to very different choice of items and consequently to different student experience. The reason for small differences in RMSE is not similarity between models, but characteristics of data ("good choice of suitable items"), which make predictions difficult and even a naive predictor comparatively good.

Another observation concerns comparison between the "Elo concepts" and "Elo concepts (wrong)" models. When data are collected by the "Elo concepts (wrong)" model, these two models achieve nearly the same performance, i.e., models seem to be of the same quality. But the other cases show that the "Elo concepts" model is better (and in fact it is by construction a better student model).

4 Conclusions

We have used simulated data to show that even small differences in predictive accuracy of student models (as measured by RMSE) may have important impact on behavior of adaptive educational systems and for interpretation of results of evaluation. Experiments with simulated data, of course, cannot demonstrate the practical impact of such small differences. We also do not claim that small differences in predictive accuracy are always important. However, experiments with simulated data are definitely useful, because they clearly illustrate mechanisms that could play role in interpretation of results of experiments with real student data. Simulated data also provide setting for formulation of hypotheses that could be later evaluated in experiments with real educational systems.

Simulated data also enable us to perform experiments that are not practical for realization with actual educational systems. For example in our experiment with the "feedback loop" we have used different student models as a basis for item selection. Our set of models includes even a very simple "constant model",

which leads to random selection of practiced item. In real setting we would be reluctant to apply such a model, as it is in contrary with the advertised intelligent behavior of our educational systems. However, experiments with this model in simulated setting provide interesting results – they clearly demonstrate that differences in predictive accuracy of models do not depend only on the intrinsic quality of used student models, but also on the way the data were collected.

Our analysis shows one particularly interesting aspect of student modeling. As we improve student models applied in educational systems, we should expect that evaluations of predictive accuracy performed over these data will show worse absolute values of performance metrics and smaller and smaller differences between models (even if models are significantly different), just because virtues of our models enable us to collect less predictable data.

References

1. Joseph E Beck and Xiaolu Xiong. Limits to accuracy: How well can we do at student modeling. In *Proc. of Educational Data Mining*, 2013.
2. Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.
3. Philipp Doebler, Mohsen Alavash, and Carsten Giessing. Adaptive experiments with a multivariate elo-type algorithm. *Behavior research methods*, pages 1–11, 2014.
4. Arpad E Elo. *The rating of chessplayers, past and present*, volume 3. Batsford London, 1978.
5. Stephen E Fancsali, Tristan Nixon, and Steven Ritter. Optimal and worst-case performance of mastery learning assessment with bayesian knowledge tracing. In *Proc. of Educational Data Mining*, 2013.
6. Stephen E Fancsali, Tristan Nixon, Annalies Vuong, and Steven Ritter. Simulated students, mastery learning, and improved learning curves for real-world cognitive tutors. In *AIED Workshops*. Citeseer, 2013.
7. Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.
8. Tanja Käser, Kenneth R Koedinger, and Markus Gross. Different parameters-same prediction: An analysis of learning curves. In *Proceedings of 7th International Conference on Educational Data Mining. London, UK*, 2014.
9. Kenneth R Koedinger, John C Stamper, Elizabeth A McLaughlin, and Tristan Nixon. Using data-driven discovery of better student models to improve student learning. In *Artificial intelligence in education*, pages 421–430. Springer, 2013.
10. Jung In Lee and Emma Brunskill. The impact on individualizing student models on necessary practice opportunities. *International Educational Data Mining Society*, 2012.
11. R Charles Murray, Steven Ritter, Tristan Nixon, Ryan Schwiebert, Robert GM Hausmann, Brendon Towle, Stephen E Fancsali, and Annalies Vuong. Revealing the learning in learning curves. In *Artificial Intelligence in Education*, pages 473–482. Springer, 2013.

12. Jan Papoušek, Radek Pelánek, and Vít Stanislav. Adaptive practice of facts in domains with varied prior knowledge. In *Proc. of Educational Data Mining*, pages 6–13, 2014.
13. Zachary A Pardos and Michael V Yudelson. Towards moment of learning accuracy. In *AIED 2013 Workshops Proceedings Volume 4*, page 3, 2013.
14. Radek Pelánek. Application of time decay functions and Elo system in student modeling. In *Proc. of Educational Data Mining*, pages 21–27, 2014.
15. Radek Pelánek. Metrics for evaluation of student models. *Journal of Educational Data Mining*, 2015. To appear.
16. Radek Pelánek. Modeling student learning: Binary or continuous skill? In *Proc. of Educational Data Mining*, 2015.
17. Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M McNee, Joseph A Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 127–134. ACM, 2002.
18. Michael V Yudelson and Kenneth R Koedinger. Estimating the benefits of student model improvements on a substantive scale. In *EDM 2013 Workshops Proceedings*, 2013.

Using Data from Real and Simulated Learners to Evaluate Adaptive Tutoring Systems

José P. González-Brenes[1], Yun Huang[2]

[1] Pearson School Research & Innovation Network, Philadelphia, PA, USA
jose.gonzalez-brenes@pearson.com
[2] Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA, USA
yuh43@pitt.edu

Abstract. Classification evaluation metrics are often used to evaluate adaptive tutoring systems— programs that teach and adapt to humans. Unfortunately, evidence suggests that existing convention for evaluating tutoring systems may lead to suboptimal decisions. In a companion paper, we propose Teal, a new framework to evaluate adaptive tutoring. In this paper we propose an alternative formulation of Teal using simulated learners. The main contribution of this novel formulation is that it enables approximate inference of Teal, which may useful on the cases that Teal becomes computationally intractable. We believe that this alternative formulation is simpler, and we hope it helps as a bridge between the student modeling and simulated learners community.

1 Introduction

Adaptive systems teach and adapt to humans and improve education by optimizing the subset of *items* presented to students, according to their historical performance [3], and on features extracted from their activities [6]. In this context, items are questions, or tasks that can be graded individually. Adaptive tutoring may be evaluated with randomized control trials. For example, in a seminal study [3] that focused on earlier adaptive tutors, a controlled trial measured the time students spent on tutoring, and their performance on post-tests. The study reported that the adaptive tutoring system enabled significantly faster teaching, while students maintained the same or better performance on post-tests

Unfortunately, controlled trials can become extremely expensive and time consuming to conduct: they require institutional review board approvals, experimental design by an expert, recruiting and often payment of enough participants to achieve statistical power, and data analysis. Automatic evaluation metrics improve the engineering process because they enable less expensive and faster comparisons between alternative systems.

The adaptive tutoring community has tacitly adopted conventions for evaluating tutoring systems [4]. Researchers often evaluate their models with classification evaluation metrics that assess the *student model* component of the tutoring system— student models are the subsystems that forecast whether a learner will answer the next item correctly. However, automatic evaluation metrics are

intended to measure an outcome of the end user. For example, the PARADISE [9] metric used in spoken dialogue systems correlates to user satisfaction scores. We are not aware of evidence that supports that classification metrics correlate with learning outcomes; yet there is a growing body of evidence [2, 5] that suggests serious problems with them. For example, classification metrics ignore that an adaptive system may not help learners— which could happen with a student model with a flat or decreasing learning curve [1, 8]. A decreasing learning curve implies that student performance decreases with practice; this curve is usually interpreted as a modeling problem, because it operationalizes that learners are better off with no teaching.

We study a novel formulation of the Theoretical Evaluation of Adaptive Learning Systems (Teal) [5] evaluation metric. The importance of evaluation metrics is that they help practitioners and researchers quantify the extent that a system helps learners.

2 Theoretical Evaluation of Adaptive Learning Systems

In this section, we just briefly summarize Teal and do not compare it with a related method called ExpOppNeed [7]. Teal assumes the adaptive tutoring system is built using a single-skill Knowledge Tracing Family model [3, 6]. Knowledge Tracing uses a Hidden Markov Model (HMM) per skill to model the student's knowledge as latent variables. It models whether a student applies a practice opportunity of a skill correctly. The latent variables are used to model the latent student proficiency, which is often modeled with a binary variable to indicated mastery of the skill.

To use Teal on data collected from students, we first train a model using an algorithm from the Knowledge Tracing family, then we use the learned parameters to calculate the effort and outcome for each skill.

- Effort: Quantifies how much practice the adaptive tutor gives to students. In this paper we focus on counting the number of items assigned to students but, alternatively, amount of time could be considered.
- Outcome: Quantifies the performance of students after adaptive tutoring. For simplicity, we operationalize performance as the percentage of items that students are able to solve after tutoring. We assume that the performance on solving items is aligned to the long-term interest of learners.

Algorithm 1 describes our novel formulation. Teal calculates the expected number of practice that an adaptive tutor gives to students. We assume that the tutor stops teaching a skill once the student is very likely to answer the next item correctly according to a model from the Knowledge Tracing Family [6]. The adaptive tutor teaches an additional item if two conditions hold: (i) it is likely that the student will get the next item wrong— in other words, the probability of answering correctly the next item is below a threshold τ; and (ii) the tutor has not decided to stop instruction already.

The inputs of Teal are:

– Real student performance data from m students practicing a skill. Data from each student is encoded into a sequence of binary observations of whether the student was able to apply correctly the skill at different points in time.
– A threshold $\tau \in \{0 \dots 1\}$ that indicates when to stop tutoring. We operationalize this threshold as the target probability that the student will apply the skill correctly.
– A parameter T that indicates the number of practice opportunities each of the simulated students will practice the skill.

Algorithm 1 Teal algorithm for models with one skill per item

Require: real student data $\mathbf{y}^{(1)} \dots \mathbf{y}^{(m)}$, threshold τ, # of simulated time steps T
function TEAL
  $\theta \leftarrow$ Knowledge_Tracing($\mathbf{y}^{(1)} \dots \mathbf{y}^{(m)}$)
  $e \leftarrow \{\ \}$
  $s \leftarrow \{\ \}$
  for $\hat{\mathbf{y}} \in$ get_simulated_student(θ, T) **do**:
    $e \leftarrow$ calculate_effort($\hat{\mathbf{y}}, \theta, \tau$)
    if $e < T$ **then**
      $s \leftarrow$ calculate_score($\hat{\mathbf{y}}$, e)
    else
      $s \leftarrow$ imputed_value
  return mean(e), mean(s)

Teal learns a Knowledge Tracing model from the data collected from real students interacting with a tutor. Our new formulation uses simulated learners sampled from the Knowledge Tracing parameters. This enables us to decide how many simulated students to generate. Our original formulation required 2^m sequences to be generated, which can quickly become computationally intractable. If an approximate solution is acceptable, our novel formulation allows more efficient calculations of Teal. Teal quantifies the effort and outcomes of students in adaptive tutoring. Even though measuring effort and outcomes is not novel by itself, Teal's contribution is measuring both without a randomized trial. Teal quantifies effort as how much practice the tutor gives. For this, we count the number of items assigned to students. For a single simulated student, this is:

$$\text{calculate\_effort}(y_1, \dots, y_T, \theta, \tau) \equiv \arg\min_t p(y_t | y_1 \dots y_{t-1}, \theta) > \tau \qquad (1)$$

The threshold τ implies a trade-off between student effort and scores and responds to external expectations from the social context. Teal operationalizes the outcome as the performance of students after adaptive tutoring as the percentage of items that students are able to solve after tutoring:

$$\text{calculate\_score}(y_1, \dots, y_T, e) \equiv \sum_{t=e} \frac{\delta(\mathbf{y}_t, \text{correct})}{T - e} \qquad (2)$$

Here, $\delta(\cdot, \cdot)$ is the Kronecker function that returns 1 iff its arguments are equal.

3 Discussion

Simulation enables us to measure effort and outcome for a large population of students. Previously, we required Teal to be computed exhaustively on all student outcomes possibilities. We relax the prohibitively expensive requirement of calculating all student outcome combinations. Our contribution is that Teal can be calculated with a simulated dataset size that is large yet tractable.

References

1. R. Baker, A. Corbett, and V. Aleven. More Accurate Student Modeling through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. In B. Woolf, E. Aïmeur, R. Nkambou, and S. Lajoie, editors, *Intelligent Tutoring Systems*, volume 5091 of *Lecture Notes in Computer Science*, pages 406–415. Springer Berlin / Heidelberg, 2008.
2. J. Beck and X. Xiong. Limits to accuracy: how well can we do at student modeling? In S. K. D'Mello, R. A. Calvo, and A. Olney, editors, *Proceedings of the 6th International Conference on Educational Data Mining, Memphis, Tennessee, USA, July 6-9, 2013*, pages 4–11. International Educational Data Mining Society, 2013.
3. A. Corbett and J. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1995.
4. A. Dhanani, S. Y. Lee, P. Phothilimthana, and Z. Pardos. A comparison of error metrics for learning model parameters in bayesian knowledge tracing. Technical Report UCB/EECS-2014-131, EECS Department, University of California, Berkeley, May 2014.
5. González-Brenes and Y. José P., Huang. Your model is predictive— but is it useful? theoretical and empirical considerations of a new paradigm for adaptive tutoring evaluation. In J. G. Boticario, O. C. Santos, C. Romero, and M. Pechenizkiy, editors, *Proceedings of the 8th International Conference on Educational Data Mining*, Madrid, Spain, 2015.
6. J. P. González-Brenes, Y. Huang, and P. Brusilovsky. General Features in Knowledge Tracing: Applications to Multiple Subskills, Temporal Item Response Theory, and Expert Knowledge. In M. Mavrikis and B. M. McLaren, editors, *Proceedings of the 7th International Conference on Educational Data Mining*, London, UK, 2014.
7. J. I. Lee and E. Brunskill. The impact on individualizing student models on necessary practice opportunities. In K. Yacef, O. R. Zaïane, A. Hershkovitz, M. Yudelson, and J. C. Stamper, editors, *Proceedings of the 5th International Conference on Educational Data Mining*, pages 118–125, Chania, Greece, 2012.
8. D. Rai, Y. Gong, and J. E. Beck. Using dirichlet priors to improve model parameter plausibility. In T. Barnes, M. Desmarais, C. Romero, and S. Ventura, editors, *Proceedings of the 2nd International Conference on Educational Data Mining*, Cordoba, Spain, 2009.
9. M. Walker, C. Kamm, and D. Litman. Towards developing general models of usability with PARADISE. *Natural Language Engineering*, 6(3):363–377, 2001.

Authoring Tutors with Complex Solutions: A Comparative Analysis of Example Tracing and SimStudent

Christopher J. MacLellan[1], Erik Harpstead[1], Eliane Stampfer Wiese[1], Mengfan Zou[2], Noboru Matsuda[1], Vincent Aleven[1], and Kenneth R. Koedinger[1]

[1] Carnegie Mellon University, Pittsburgh PA, USA,
{cmaclell, eharpste, stampfer, noboru.matsuda, aleven, koedinger}@cs.cmu.edu,
[2] Tsinghua University, Beijing, China,
zmf11@mails.tsinghua.edu.cn

Abstract. Problems with many solutions and solution paths are on the frontier of what non-programmers can author with existing tutor authoring tools. Popular approaches such as Example Tracing, which allow authors to build tutors by demonstrating steps directly in the tutor interface. This approach encounters difficulties for problems with more complex solution spaces because the author needs to demonstrate a large number of actions. By using SimStudent, a simulated learner, it is possible to induce general rules from author demonstrations and feedback, enabling efficient support for complexity. In this paper, we present a framework for understanding solution space complexity and analyze the abilities of Example Tracing and SimStudent for authoring problems in an experimental design tutor. We found that both non-programming approaches support authoring of this complex problem. The SimStudent approach is 90% more efficient than Example Tracing, but requires special attention to ensure model completeness. Example Tracing, on the other hand, requires more demonstrations, but reliably arrives at a complete model. In general, Example Tracing's simplicity makes it good for a wide range problems, a reason for why it is currently the most widely used authoring approach. However, SimStudent's improved efficiency makes it a promising non-programmer approach, especially when solution spaces become more complex. Finally, this work demonstrates how simulated learners can be used to efficiently author models for tutoring systems.

Keywords: Tutor Authoring, Intelligent Tutoring Systems, Cognitive Modeling, Programming-by-Demonstration

1 Introduction

Intelligent Tutoring Systems (ITSs) are effective at improving student learning across many domains– from mathematics to experimental design [10, 13, 5]. ITSs

also employ a variety of pedagogical approaches for learning by doing, including intelligent novice [7], invention [12], and learning by teaching [9]. Many of these approaches require systems that can model complex solution spaces that accommodate multiple correct solutions to a problem and/or multiple possible paths to each solution. Further, modeling complex spaces can be desirable pedagogically: student errors during problem solving can provide valuable learning opportunities, and therefore may be desirable behaviors. Mathan and Koedingers spreadsheet tutor provides experimental support for this view– a tutor that allowed exploration of incorrect solutions led to better learning compared to one that enforced a narrower, more efficient solution path [7]. However, building tutoring systems for complex solution spaces has generally required programming. What options are available to the non-programmer? Authoring tools have radically reduced the difficulties and costs of tutor building [2, 6], and have allowed authoring without programming. Through the demonstration of examples directly in the tutor interface, an author can designate multiple correct solutions, and many correct paths to each solution. Yet, the capabilities of these tools for authoring problems with complex solution spaces has never been systematically analyzed.

In this paper, we define the concept of solution space complexity and, through a case study, explore how two authoring approaches deal with this complexity. Both approaches (Example Tracing and SimStudent) are part of the Cognitive Tutor Authoring Tools (CTAT) [1]. Our case study uses the domain of introductory experimental design, as problems in this area follow simple constraints (only vary one thing at a time), but solutions can be arbitrarily complex depending on how many variables are in the experiment and how many values each can take.

2 Solution Space Complexity

Solution spaces have varying degrees of complexity. Our framework for examining complexity considers both how many correct solutions satisfy a problem and how many paths lead to each solution. Within this formulation, we discuss how easily a non-programmer can author tutors that support many solutions and/or many paths to a solution.

How might this formulation of complexity apply to an experimental design tutor? Introductory problems in this domain teach the control of variables strategy (only manipulating a single variable between experimental conditions to allow for causal attribution) [3]. Due to the combinatorial nature of experiments (i.e., multiple conditions, variables, and variable values), the degree of complexity in a particular problem depends on how it is presented. To illustrate, imagine that students are asked to design an experiment to determine how increasing the heat of a burner affects the melting rate of ice in a pot (see Figure 1). The following tutor prompts (alternatives to the prompt in Figure 1) highlight how different problem framings will affect the solution complexity:

One solution with one path Design an experiment to determine how increasing the heat of a Bunsen burner affects the rate at which ice in a pot will

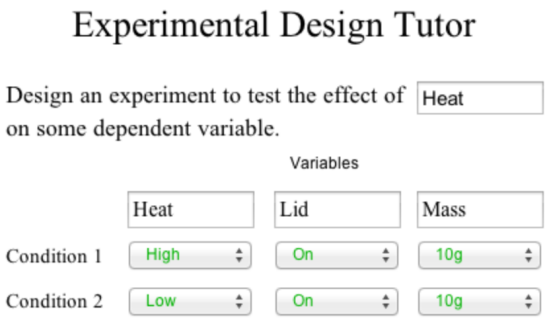

Fig. 1. Experimental design tutor interface

melt by assigning the first legal value to the variables in left to right, top down order as they appear in the table.

One solution and many paths Design an experiment to determine how increasing the heat of a Bunsen burner affects the rate at which ice in a pot will melt by assigning the first legal value to variables.

Many solutions each with one path Design an experiment to determine how increasing the heat of a Bunsen burner affects the rate at which ice in a pot will melt by assigning values to variables in left to right, top down order as they appear in the table.

Many solutions with many paths Design an experiment to determine how increasing the heat of a Bunsen burner affects the rate at which ice in a pot will melt.

While these examples show that solution space complexity can be qualitatively changed (i.e., one solution vs. many solutions) by reframing a problem, quantitative changes are also possible. For example, adding a fourth variable to the interface in Figure 1 would require two more steps per solution path (setting the variable for each condition), while adding another value to each variable increases the number of possible options at each step of the solution path. As this example illustrates, solution space complexity is not an inherent property of a domain, but rather arises from an authors design choices.

3 Tutor Authoring

Our analysis focuses on the Cognitive Tutor Authoring Tools (CTAT), as CTAT is the most widely used tutor authoring tool and the approaches it supports are representative of authoring tools in general [2]. CTAT supports non-programmers in building both tutor interfaces and cognitive models (for providing feedback). Cognitive models can be constructed with Example Tracing or SimStudent. In this section, we step through how Example-Tracing and SimStudent approaches would be applied by non-programmers to the experimental design task, using the

interface shown in Figure 1. Further, we discuss the features of each approach for handling solution space complexity in the context of this example.

3.1 Example Tracing

When building an Example-Tracing tutor in CTAT, the author demonstrates correct solutions directly in the tutoring interface. These demonstrated steps are recorded in a behavior graph. Each node in the behavior graph represents a state of the tutoring interface, and each link represents an action that moves the student from one node to another. In Example Tracing each link is produced as a result of a single action demonstrated directly in the tutor interface; many legal actions might be demonstrated for each state, creating branches in the behavior graph.

Figure 2 shows an example of our experimental design tutor interface and an associated behavior graph. The particular prompt chosen has 8 solutions and many paths to each solution. These alternative paths correspond to different orders in which the variables in the experimental design can be assigned. The Example-Tracing approach allows authors to specify that groups of actions can be executed in any order. In the context of our example, this functionality allows the author to demonstrate one path to each of the 8 unique solutions (these 8 paths are visible in Figure 2) and then specify that the actions along that path can be executed in any order. Unordered action groups are denoted in the behavior graph by colored ellipsoids.

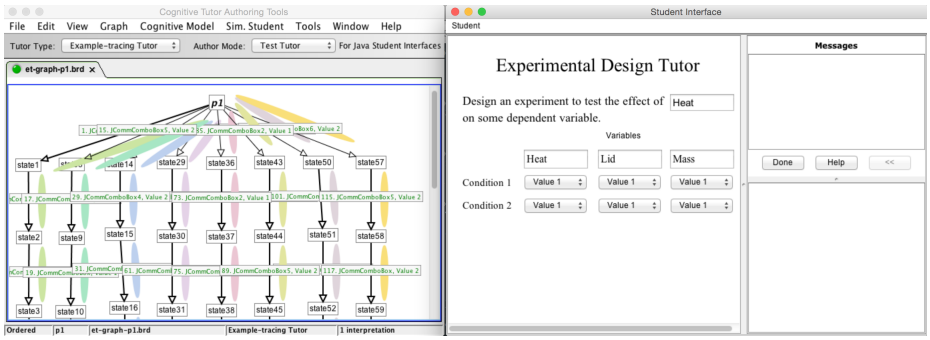

Fig. 2. An experimental design tutor (right) and its associated behavior graph (left). This tutor supports students in designing an experiment to test the effect of heat on a dependet variable. The correct answer is to pick two different values for the "Heat" variable and to hold the values constant for other variables.

Once a behavior graph has been constructed for a specific problem (e.g. determine the effect of heat on ice melting), that behavior graph can be generalized to other problems (e.g. determine the effect of sunlight on plant growth) using mass production. The mass production feature allows the author to replace specific values in the interface with variables and then to instantiate an arbitrary

number of behavior graphs with different values for the variables. This approach is powerful for supporting many different problems that have identical behavior graph structure, such as replacing all instances of "heat" with another variable, "sunlight". However, if a problem varies in the structure of its behavior graph, such as asking the student to manipulate a variable in the second column instead of the first (e.g., "lid" instead of "heat"), then a new behavior graph would need to be built to reflect the change in the column of interest.

How efficient is Example Tracing in building a complete cognitive model for the experimental design problem? The complete model consists of 3 behavior graphs (one for each of the three variable columns that could be manipulated). Each graph took 56 demonstrations and required 8 unordered action groups to be specified. Thus, the complete cognitive model required 168 demonstrations and 24 unordered group specifications. Using estimates from a previously developed Keystroke-Level Model [6], which approximates the time needed for an error-free expert to perform each interface action, we estimate that this model would take about 27 minutes to build using Example Tracing. Notably, the ability to specify unordered action groups offers substantial efficiency gains - without it, authoring would take almost 100 hours. Furthermore, with mass production, this model can generalize to any set of authored variables.

3.2 SimStudent

While the Example-Tracing behavior graph creates links from user demonstrations, the SimStudent system extends these capabilities by inducing production rule models from demonstrations and feedback (for details on this rule induction see [8]). In the experimental design tutor, SimStudent might learn a rule that sets one of the variables to an arbitrary value when no values for that variable have been assigned. Then, it might learn different rules for setting a variables second value based on whether or not it is being manipulated.

Authoring with SimStudent is similar to Example Tracing in that SimStudent asks for demonstrations when it does not know how to proceed. However, when SimStudent already has an applicable rule, it fires the rule and shows the resulting action in the tutor interface. It then asks the author for feedback on that action. If the feedback is positive, SimStudent may refine the conditions of its production rules before continuing to solve the problem. If the feedback is negative, SimStudent will try firing a different rule. When SimStudent exhausts all of its applicable rules, it asks the author to demonstrate a correct action. Figure 3 shows how SimStudent asks for demonstrations and feedback. When authoring with SimStudent, the author does not have to specify rule order - as long as a rule's conditions are satisfied, it is applicable. Authoring with SimStudent produces both a behavior graph (of the demonstrations and actions SimStudent took in the interface) and a production rule model.

To evaluate the efficiency of the SimStudent approach we constructed a complete model for the experimental design tutor. It can be difficult to determine when a SimStudent model is correct and complete from the authoring interactions alone. In most cases the SimStudent model is evaluated with set of held-out

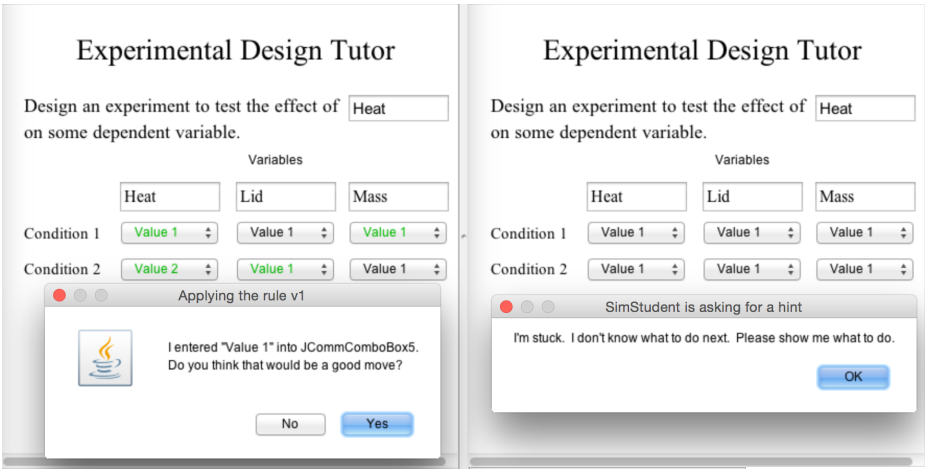

Fig. 3. SimStudent asking for feedback (left) and for a demonstration (right).

test problems (i.e., unit tests). However, in this case the learned rules were simple enough to evaluate by direct inspection. We noticed that SimStudent learned one correct strategy, but had not explored other solutions. This is typical of SimStudent - once it learns a particular strategy it applies it repeatedly. Therefore, authors must give it additional demonstrations of alternative paths. With the experimental design tutor, we noticed that SimStudent was always choosing the first value for non-manipulated variables, so we gave it additional demonstrations where non-manipulated variables took values besides those demonstrated on the initial run.

Ultimately, SimStudent acquired a complete model after 7 demonstrations and 23 feedback responses. Using the same Keystroke-Level Model from [6], we estimate that building a cognitive model using SimStudent would take an error-free expert about 2.12 minutes – much shorter than Example Tracing. Like Example Tracing, the model produced by SimStudent can work with arbitrary variables. Unlike Example Tracing, the learned model can work for unauthored variables; for example, students could define their own variables while using the tutor. This level of generality could be useful in inquiry-based learning environments [4]. Finally, if another variable column was added to the tutor, the SimStudent model would be able to function without modification. For Example Tracing, such a change would constitute a change to the behavior graph structure, so a completely new behavior graphs would need to be authored to support this addition.

4 Discussion

Both Example Tracing and SimStudent can create tutors for problems with complex solution spaces. However, our analysis shows that the two approaches

differ in terms of their efficiency and, as a result, how many solutions and paths they can handle in practice.

First, the Example-Tracing approach worked very well, even though the experimental design problems have a combinatorial structure. In particular, unordered action groups and mass production drastically reduced the number of demonstrations needed to cover the solution space, 168 vs. 40,362. The simplicity of Example Tracing combined with the power afforded by these features is likely why Example Tracing is the most widely used authoring approach today [2].

The SimStudent approach was more efficient than Example Tracing (approx. 2.12 vs. 27 minutes), but this comparison requires several caveats. The machine learning mechanisms of SimStudent generalize demonstrations and feedback into rules, which allows SimStudent to only model unique actions and the conditions under which they apply. However, this means SimStudent may not acquire a complete model. In the experimental design case study, SimStudent at first only learned that non-manipulated variables take their first value (rather than any value that is constant across conditions). In general, this problem arises when SimStudent acquires a model that can provide at least one correct solution for any problem. In these situations, it never prompts an author to provide alternative demonstrations; leading an unsuspecting author to create an incomplete model. A related complication is determining when the SimStudent model is complete. While determining the completeness of models in both Example Tracing and SimStudent can be difficult, authors must attempt to infer completeness from SimStudent's problem solving performance– a method that can be rather opaque at times. Thus, an open area for simulated learning systems is how best to evaluate the quality of learned models.

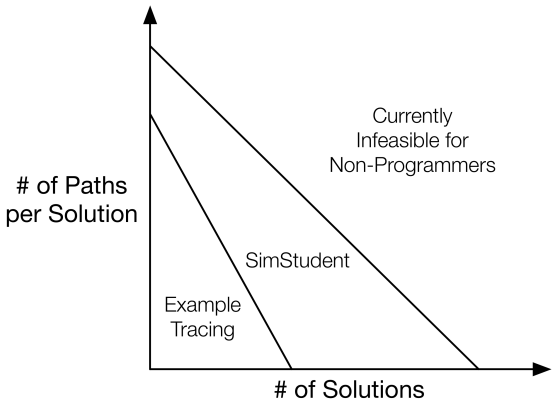

Fig. 4. How the space of solution space complexity is handled by existing non-programmer authoring approaches.

Overall our findings, when paired with those of previous work [6], suggest an interpretation depicted in Figure 4. In this figure the potential space of complexity is depicted in terms of number of unique solutions and number of paths per solution. The inner region denotes the area of the complexity space where we believe Example Tracing will maximize non-programmers' authoring utility. This region is skewed towards a higher number of paths, owing to Example Tracing's capacity to specify unordered actions. This portion of the complexity space contains many of the tutors that have already been built using Example Tracing [2]. As the complexity of a problem's solution space increases, Example Tracing becomes less practical (though still capable) and SimStudent becomes a more promising option, despite the caveats for using it. SimStudent's power of rule generalization gives it the ability to deal with more paths and unique solutions with less author effort, however, these capabilities come with the risk of producing incomplete models (without the author being aware).

Notably missing in the figure is any coverage of the upper right quadrant. This area would be a fruitful place to direct future work that supports non-programmers in authoring problems with many solutions with many paths. In particular, simulated learning systems might be extended to give non-programmers access to this portion of the space. One existing approach for dealing with highly complex solution spaces is to only model the aspects of the space that students are most likely to traverse. For example, work by Rivers and Koedinger [11] has explored the use of prior student solutions to seed a feedback model for introductory programming tasks. As it stands this area can only be reached using custom built approaches and would benefit from authoring tool research.

One limitation of our current approach is the assumption that there is a body of non-programmers that wants to build tutors for more complex problems. Our analysis here suggests that there is an open space for non-programming tools that support highly complex solution spaces, but it is less clear that authors have a desire to create tutors in this portion of the space. A survey of authors interested in building complex tutors without programming would help to shed light on what issues non-programmers are currently having in building their tutors. It is important that such a survey also include the perspective of those outside the normal ITS community to see if there are features preventing those who are interested from entering the space.

From a pedagogical point of view, it is unclear how much of the solution space needs to be modeled in a tutor. Waalkens et al. [16] have explored this topic by implementing three versions of an Algebra equation solving tutor, each with progressively more freedom in the number of paths that students can take to a correct solution. They found that the amount of freedom did not have an effect on students learning outcomes. However, there is evidence that the ability to use and decide between different strategies (i.e. solution paths) is linked with improved learning [14]. Further, subsequent work [15] has suggested that students only exhibit strategic variety if they are given problems that favor different strategies. Regardless of whether modeling the entire solution space is

pedagogically necessary, it is important that available tools support the ability to model complex spaces so that these research questions can be further explored.

5 Conclusion

The results of our analysis suggest that both the Example Tracing and SimStudent authoring approaches are promising methods for non-programmers to create tutors even for problems with many solutions with many paths. More specifically, we found that SimStudent was more efficient for authoring a tutor for experimental design, but authoring with SimStudent had a number of caveats related to ensuring that the authored model was complete. In contrast, Example Tracing was simple to use and it was clear that the authored models were complete. Overall, our analysis shows that Example Tracing is good for a wide range of problems that non-programmers might want to build tutors for (supported by its extensive use in the community [2]). However, the SimStudent approach shows great promise as an efficient authoring approach, especially when the solution space becomes complex. In any case, more research is needed to expand the frontier of non-programmers' abilities to author tutors with complex solution spaces.

Finally, this work demonstrates the feasibility and power of utilizing a simulated learning system (i.e., SimStudent) to facilitate the tutor authoring process. In particular authoring tutors with SimStudent took only 10% of the time that it took to author a tutor with Example-Tracing, a non-simulated learner approach. Educational technologies with increasingly complex solution spaces are growing in popularity (e.g. educational games and open-ended learning environments), but current approaches do not support non-programmers in authoring tutors for these technologies. Our results show that simulated learning systems are a promising tool for supporting these non-programmers. However, more work is needed to improve our understanding of how simulated learners can contribute to the authoring process and how the models learned by these systems can be evaluated.

6 Acknowledgements

We would like to thank Caitlin Tenison for her thoughtful comments and feedback on earlier drafts. This work was supported in part by a Graduate Training Grant awarded to Carnegie Mellon University by the Department of Education (#R305B090023) and by the Pittsburgh Science of Learning Center, which is funded by the NSF (#SBE-0836012). This work was also supported in part by National Science Foundation Awards (#DRL-0910176 and #DRL-1252440) and the Institute of Education Sciences, U.S. Department of Education (#R305A090519). All opinions expressed in this article are those of the authors and do not necessarily reflect the position of the sponsoring agency.

References

1. Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In: Ikeda, M., Ashley, K.D., Tak-Wai, C. (eds.) ITS '06. pp. 61–70. Springer (2006)
2. Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors. IJAIED 19(2), 105–154 (2009)
3. Chen, Z., Klahr, D.: All Other Things Being Equal: Acquisition and Transfer of the Control of Variables Strategy. Child Development 70(5), 1098–1120 (1999)
4. Gobert, J.D., Koedinger, K.R.: Using Model-Tracing to Conduct Performance Assessment of Students' Inquiry Skills within a Microworld. Society for Research on Educational Effectiveness (2011)
5. Klahr, D., Triona, L.M., Williams, C.: Hands on what? The relative effectiveness of physical versus virtual materials in an engineering design project by middle school children. Journal of Research in Science Teaching 44(1), 183–203 (Jan 2007)
6. MacLellan, C.J., Koedinger, K.R., Matsuda, N.: Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality. In: Trausen-Matu, S., Boyer, K. (eds.) ITS '14 (2014)
7. Mathan, S.A., Koedinger, K.R.: Fostering the Intelligent Novice: Learning From Errors With Metacognitive Tutoring. Educational Psychologist 40(4), 257–265 (2005), http://www.tandfonline.com/doi/abs/10.1207/s15326985ep4004_7
8. Matsuda, N., Cohen, W.W., Koedinger, K.R.: Teaching the Teacher: Tutoring SimStudent Leads to More Effective Cognitive Tutor Authoring. IJAIED 25(1), 1–34 (2014)
9. Matsuda, N., Yarzebinski, E., Keiser, V., Cohen, W.W., Koedinger, K.R.: Learning by Teaching SimStudent – An Initial Classroom Baseline Study Comparing with Cognitive Tutor. IJAIED (2011)
10. Pane, J.F., Griffin, B.A., McCaffrey, D.F., Karam, R.: Effectiveness of Cognitive Tutor Algebra I at Scale. Tech. rep., RAND Corporation, Santa Monica, CA (2013)
11. Rivers, K., Koedinger, K.R.: Automating Hint Generation with Solution Space Path Construction. In: ITS '14, pp. 329–339. Springer (2014)
12. Roll, I., Aleven, V., Koedinger, K.R.: The Invention Lab : Using a Hybrid of Model Tracing and Constraint-Based Modeling to Offer Intelligent Support in Inquiry Environments. In: ITS '10. pp. 115–124 (2010)
13. Sao Pedro, M.A., Gobert, J.D., Heffernan, N.T., Beck, J.E.: Comparing Pedagogical Approaches for Teaching the Control of Variables Strategy. In: Taatgen, N., van Rijn, H. (eds.) CogSci '09. pp. 1–6 (2009)
14. Schneider, M., Rittle-Johnson, B., Star, J.R.: Relations among conceptual knowledge, procedural knowledge, and procedural flexibility in two samples differing in prior knowledge. Developmental Psychology 47(6), 1525–1538 (2011)
15. Tenison, C., MacLellan, C.J.: Modeling Strategy Use in an Intelligent Tutoring System: Implications for Strategic Flexibility. In: ITS '14, pp. 466–475. Springer (2014)
16. Waalkens, M., Aleven, V., Taatgen, N.: Computers & Education. Computers & Education 60(1), 159–171 (Jan 2013)

Methods for Evaluating Simulated Learners: Examples from SimStudent

Kenneth R. Koedinger[1], Noboru Matsuda[1], Christopher J. MacLellan[1], and Elizabeth A. McLaughlin[1]

[1] Carnegie Mellon University, Pittsburgh, PA
koedinger@cmu.edu

Abstract. We discuss methods for evaluating simulated learners associated with four different scientific and practical goals for simulated learners. These goals are to develop a precise theory of learning, to provide a formative test of alternative instructional approaches, to automate authoring of intelligent tutoring systems, and to use as a teachable agent for students to learn by teaching. For each goal, we discuss methods for evaluating how well a simulated learner achieves that goal. We use SimStudent, a simulated learner theory and software architecture, to illustrate these evaluation methods. We describe, for example, how SimStudent has been evaluated as a theory of student learning by comparing, across four domains, the cognitive models it learns to the hand-authored models. The SimStudent-acquired models generally yield more accurate predictions of student data. We suggest future research into directly evaluating simulated learner predictions of the process of student learning.

Keywords: cognitive models, learning theory, instructional theory

1 Introduction

When is a simulated learner a success? We discuss different approaches to evaluating simulated learners (SLs). Some of these evaluation approaches are technical in nature, whether or how well a technical goal has been achieved, and some are empirical, whereby predictions from the SL are compared against data. These approaches can be framed with respect to four goals for developing SLs (see Table 1). These goals have been pursued in prior SL research, such as the use of "pseudo-students" [1] to test the quality of an instructional design (#2 in Table 1). Before describing different evaluation approaches appropriate for different goals, we first introduce SimStudent.

1.1 SimStudent: A Simulated Learner Theory and Software Architecture

SimStudent [2,3] is an SL system and theory in the class of adaptive production systems as defined by [4]. As such, it is similar to cognitive architectures such as ACT-R [5], Soar [6], and Icarus [7], however, it distinctly focuses on modeling inductive knowledge-level learning [8] of complex academic skills learning. SimStudent learns

from a few primary forms of instruction, including examples of correct actions, skill labels on similar actions, clues for what information in the interface to focus on to infer a next action, and finally yes-or-no feedback on actions performed by SimStudent.

Table 1. Scientific and Practical Goals for Simulated Learners (SLs)

1. *Precise Theory*. Use SLs to develop and articulate precise theory of learning.
   a. *Cognitive Model*. Create theories of domain expertise
   b. *Error Model*. Create theories of student domain misconceptions
   c. *Prior Knowledge*. Create theories of how prior knowledge changes learning
   d. *Learning Process*. Create theories of change in knowledge and performance
2. *Instructional Testing*. Use SLs as a "crash test" to evaluate instruction
3. *Automated Authoring*. Use SLs to automatically an intelligent tutoring system
4. *Teachable Agent*. Use SLs as a teachable agent or peer

To tutor SimStudent, a problem is entered in the tutoring interface (e.g., 2x = 8 in row 1 of Figure 1). SimStudent attempts to solve the problem by applying productions learned so far. If an applicable production is found, it is fired and problem interface is updated. The author then provides correctness *feedback* on SimStudent's step. If no correct production application is found, SimStudent asks the author to demonstrate the next step directly in the interface. When providing a demonstration, the author first specifies the *focus of attention* (i.e. input fields relevant to the current step) by double-clicking the corresponding interface elements (e.g., the cells containing 2x and 8 in Figure 1). The author takes action using the relevant information (e.g., entering divide 2 in Figure 1). Finally, the *author specifies a skill name* by clicking on the newly added edge of the behavior graph. This skill label is used to help guide SimStudent's learning and to make production rule names more readable.

SimStudent uses three machine-learning mechanisms (*how*, *where*, and *when*) to acquire production rules. When given a new demonstration (i.e., a positive example of a rule), SimStudent uses its *how* learner to produce a general composition of functions that replicate the demonstrated steps and ones like it. For example, in Figure 1, when given the demonstration "divide 2*"* for the problem 2x=8, SimStudent induces that the result of the "get-first-integer-without-sign" function when applied to left side of the problem and appended to the word "divide" explains the demonstration.

After an action sequence has been discovered, SimStudent uses its *where* learner to identify a generalized path to the focus of attention in the tutor interface. In Figure 1, the *where* learner discovers retrieval paths for the three cells in the first column. These paths are generalized as more positive examples and are acquired for a given rule. For example, when the author demonstrates the application of the divide rule shown in Figure 1 to the second row of the equation table, then the production retrieval path is generalized to work over any row in the equation table.

Finally, after learning an action sequence and general paths to relevant information, SimStudent uses its *when* learner to identify the conditions under which the learned production rule produces correct actions. For example, in Figure 1 SimStudent learns that this rule can only be correctly applied when one side of the equation

has a coefficient. In situations when SimStudent receives positive and negative feedback on its rule applications, it uses the *when* learner to update the conditions on the rules. Note, the *how* and *where* learners primarily use positive examples.

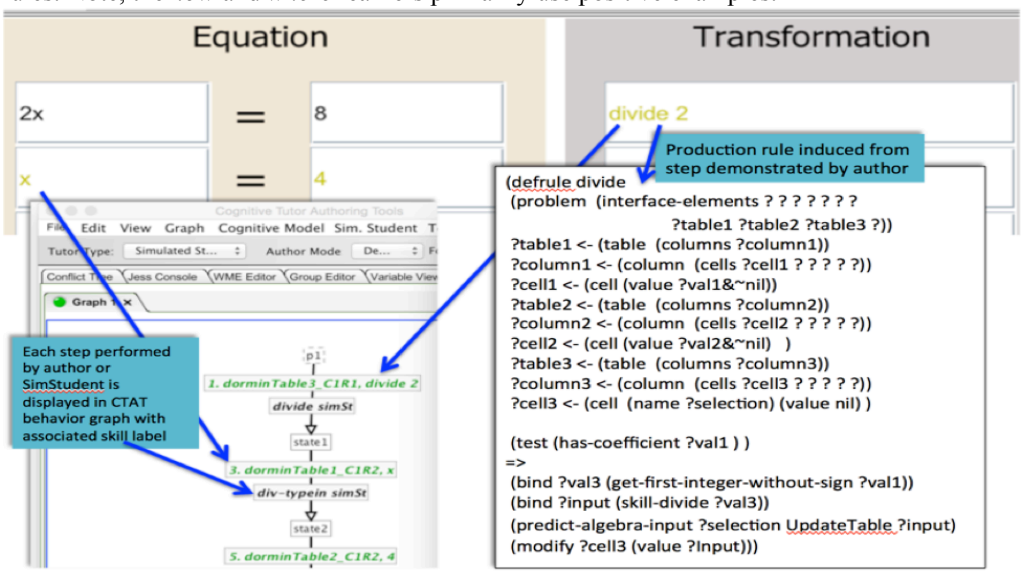

Fig. 1. After entering a problem, "2x=8" (top left), teaching of SimStudent occurs either by giving yes-or-no feedback when SimStudent attempts a step or by demonstrating a correct step when SimStudent cannot (e.g., "divide 2").

SimStudent is also capable of learning the representation of the chunks that make up the production system's working memory and are the informational basis on which productions are learned. It does so using an unsupervised grammar induction approach [3]. This feature sets it apart from other production rule learning systems.

2 Evaluating Simulated Learners as Theories of Learning

It is helpful to distinguish a general theory of learning from a theory of *student* learning. We focus on student learning because of the goals of AI in Education. However, it is worth mentioning evaluation criteria for a general learning theory, such as how quickly and independently learning takes place and how general and accurate is resulting performance. These criteria facilitate comparative evaluations. For instance, hierarchical Bayesian models are arguably better models of learning than other classification or neural network models because they learn as well with fewer examples [9].

2.1 Good Learning Theory Should Generate Accurate Cognitive Models

A student learning theory should produce the kind of expertise that human students acquire. In other words, the result of teaching an SL should be a cognitive model of

what human student's know after instruction. Thus, one way to evaluate an SL is to evaluate the quality of the cognitive models it produces. We proposed [10] six constraints to evaluate the quality of a cognitive model: 1) solution sufficiency, 2) step sufficiency, 3) choice matching, 4) computational parsimony, 5) acquirability, and 6) transfer. The first two are empirical and qualitative: Is the cognitive model that the SL acquires able to solve tasks and do so with steps that are consistent with human students? The third is quantitative: Does the frequency of strategy use and common error categories generated by the cognitive model on different tasks correspond with the same frequencies exhibited by human students? The last three are rational in character, involving inspection of the cognitive model.

These constraints were designed with hand-authored models in mind, so some, like the acquirability constraint (#5), appear trivial in the SL context. There is no question that the components of an SL-produced cognitive model are plausibly acquired because the SL does, in fact, acquire them. Similarly, the solution sufficiency constraint (#1) is straightforwardly achieved if the SL does not indeed succeed in learning the task domain. If the cognitive model that is produced solves problems using the kinds of intermediate steps used in student solutions, for example, it performs its solution in a step-based tutoring system interface, then the step sufficiency constraint (#2) is met.

How, then, can the remaining constraints be evaluated? In [11], we employed an educational data mining approach that evaluates the accuracy of a cognitive model by a "smooth learning curve" criteria [cf., 12,13]. Using a relatively simple statistical model of how instructional opportunities improve the accuracy of knowledge, this approach can measure and compare cognitive models in terms of their accuracy in predicting learning curve data. To employ the statistical model fit, the cognitive model is simplified into a "Q matrix", which maps each observed task performed (e.g., entering a step in a problem solving) to the knowledge components hypothesized to be needed to successfully perform that task. For any appropriate dataset uploaded into DataShop (learnlab.org/DataShop), the website allows users to edit and upload alternative cognitive models (in the Q matrix format), automatically performs statistical model fits, renders learning curve visualizations, and displays a rank ordering of the models in terms of their predictive accuracy [14].

We used this approach to evaluate the empirical accuracy of the cognitive models that SimStudent learns as compared to hand-authored cognitive models [11]. SimStudent was tutored in four domains: algebra, fractions, chemistry, and English grammar, in which we had existing human data and existing hand-authored cognitive models. In each domain SimStudent induced, from examples and from practice with feedback, both new chunk structures to represent the organization (or "grammar") of the perceptual input and new production rules that solve problems (e.g., add two fractions) or make decisions (e.g., select when to use "the" or "a" in English sentences). In each case, the production rules that SimStudent acquired were converted into the Q matrix format. Then the DataShop cognitive model comparison was employed to compare whether these models fit student learning curve data better than the hand-authored cognitive models do.

In all four domains, the SimStudent-acquired cognitive models made distinctions not present in the hand-authored models (e.g., it had two different production rules

across tasks for which the hand-authored model had one) and thus it tended to produce models with more knowledge components (as shown in Table 2). For example, SimStudent learned two different production rules for the typical last step in equation solving where one production covered typical cases (e.g., from 3x = 12 the student should "divide by 3") and another covered a perceptually distinct special case (e.g., from -x = 12 the student should divide by -1).

In all four domains, at least some of these distinctions improved the predictive fit to the learning curve data for the relevant tasks. For example, the SimStudent-acquired cognitive model in algebra leads to better accuracy because real students had a much higher error rate on tasks like -x=12 (where the coefficient, -1, is implicit) than on tasks like 3x=12 (where the coefficient, 3, is explicitly visible). In one domain (Fraction Addition,), the SimStudent-acquired cognitive model failed to make a key distinction present in the hand-authored model and thus, while better in some cases, its overall fit was worse. In the three other domains, the SimStudent-acquired cognitive models were found to be more accurate than the hand-authored cognitive models.

Table 2. A comparison of human-generated and SimStudent-discovered models.

	Number of Production Rules		Cross-Validated RMSE	
	Human-Generated Model	SimStudent Discovered Model	Human-Generated Model	SimStudent Discovered Model
Algebra	12	21	0.4024	**0.3999**
Stoichiometry	44	46	0.3501	**0.3488**
Fraction Addition	8	6	**0.3232**	0.3343
Article selection	19	22	0.4044	**0.4033**

In other words, this "smooth learning curve" method of evaluation can provide evidence that an SL, SimStudent in this case, is a reasonable model of student learning in that it acquires knowledge at a grain size (as represented in the components of the cognitive model) that is demonstrably consistent with human data.

One limitation of this approach is that it *indirectly* compares an SL to human learners through the process of fitting a statistical model. In the case of algebra, for example, SimStudent's acquisition of two different productions for tasks of the form Nx=N versus tasks of the form -x=N gets translated into a prediction that student performance will be *different* in these situations, but the not direction of the difference. The parameter estimation in statistical model fit yields the prediction for which of these task categories (Nx=N or -x=N) is harder. A more *direct* comparison would not use an intermediate statistical model fit. It would require the SL to not only produce a relevant distinction, but to make a prediction of student performance differences, such as whether it takes longer to successfully learn some kinds of tasks than others. Such an evaluation approach is discussed in section 2.3.

2.2 Matching Student Errors and Testing Prior Knowledge Assumptions

As a model of student learning, a good SL should not only produce accurate performance with learning, but should also produce the kinds of errors that students produce [cf.,15]. Thus, comparing SL errors to student errors is another way to evaluate an SL.

One theory of student errors is that students learn incorrect knowledge (e.g., incorrect production rules or schemas) from *correct example-based instruction* due to the necessary fallibility of inductive learning processes. A further hypothesis is that inductive learning errors are more likely when students have "weak" (i.e., more domain general) rather than "strong" (i.e., more domain specific) prior knowledge. With weak prior knowledge, students may interpret examples shallowly, paying attention to more immediately perceived surface features, rather than more deeply, by making domain-relevant inferences from those surface features. Consider example-based instruction where a student is given the equation "3x+5 = 7" and told that "subtract 5" from both sides is a good next step. A novice student with weak prior knowledge might interpret this example shallowly, as subtracting a number (i.e., 5) instead of more deeply, as subtracting a term (i.e., +5). As a consequence, the student may induce knowledge that produces an error on a subsequent problem, such as "4x-2=5" where they subtract 2 from both sides. Indeed, this error is common among beginning algebra students.

We evaluated SimStudent by comparing induction errors it makes with human student errors [16]. More specifically, we evaluated the weak prior knowledge hypothesis expressed above. We conducted a simulation study by having multiple instances of SimStudent get trained by the Algebra Cognitive Tutor. We compared SimStudent behaviors with actual student data from the Cognitive Tutor's logs of student interactions with the system. When SimStudent starts with weak prior knowledge rather than strong prior knowledge, it learns more slowly, that is, the accuracy of learned skills is lower given the same amount of training. More importantly, SimStudent's ability to predict student errors increased significantly when given weak rather than strong prior knowledge. In fact, the errors generated by SimStudent with strong prior knowledge were almost never the same kinds of errors commonly made by real students.

In addition to illustrating how an SL can be evaluated by comparing its error generation to human errors, this example illustrates how an SL can be used to test assumptions about student prior knowledge. In particular, SimStudent provides a theoretical explanation of empirical results [17] showing correlations between tasks measuring prior knowledge (e.g., identify the negative terms in "3x-4 = -5-2x") and subsequent learning of target skills (e.g., solving algebra equations).

Some previous studies of students' errors focus primarily on a descriptive theory to explain why students made particular errors, for example, repair theory [15], the theory of bugs [18], and the theory of extrapolation technique [19]. With SLs, we can better understand the process of acquiring the incorrect skills that generate errors. The precise understanding that computational modeling facilitates provides us with insights into designing better learning environments that mitigate error formation.

2.3 Good Student Learning Theory Should Match Learning Process Data

Matching an SL's performance to learning process data is similar to the cognitive model evaluation discussed above in section 2.1. However, as indicated above, that approach has the limitation of being an indirect comparison with human data whereby there the fit to human data is, in a key sense, less challenging because it is mediated by a separate step parameter estimation of a statistical model. A more direct compari-

son is, in simple terms, to match the behavior of multiple instances of an SL (i.e., a whole simulated class) with the behavior of multiple students. The SLs interact with a tutoring system (like one shown in Figure 2) just as a class of human students would and their behavior is logged just as human student data is. Then the simulated and human student data logs can be compared, for example, by comparing learning curves that average across all (simulated and human) student participants.

3 Evaluating Simulated Learners as Instruction Testers

A number of projects have explored the use of an SL to compare different forms of instruction. VanLehn was perhaps the first to suggest such a use of a "pseudo student" [1]. A version of ACT-R's utility learning mechanism was used to show that the SL was more successful when given error feedback not only on target performance tasks (e.g., solving two-step equations), but also on shorter subtasks (e.g., one-step equations) [10]. A SimStudent study showed better learning from a combination of examples and problems to solve, than just giving it examples [2]. Another showed that interleaving problem types is better for learning than blocking problem types because interleaving provides better opportunities correcting over-generalization errors [20].

For a general theory of instruction, it is of scientific interest to understand the effectiveness of different forms of instruction for different kinds of SL systems even if the SL is not an accurate model of student learning. Such understanding is relevant to advancing applications of AI and is directly relevant to using an SL for automated ITS authoring (next section). Such theoretical demonstrations may also have relevance to a theory of *human* instruction as they may 1) provide theoretical explanations for instructional improvements that have been demonstrated with human learners or 2) generate predictions for what may work with human students.

These instructional conclusions can only be reliably extended to human learners when the SL is an accurate model of student learning. The most reliable evaluation of an SL as instructional tester is a follow-up random assignment experiment with human learners that demonstrates that the instructional form that was better for the SLs is also better for students. In the examples given above, there is some evidence that the SLs are accurate models of student learning (e.g., past relevant human experiments). However, in none of them was the ideal follow-up experiment performed.

4 Evaluating Simulated Learners as ITS Authoring Tools

In addition to their use as theories of learning and for testing instructional content, simulated learning systems can also be used to facilitate the authoring of Intelligent Tutoring Systems (ITS). In particular, once an SL has been sufficiently trained, the cognitive model it learns can then be used directly as an expert model. Previous work, such as Example Tracing tutor authoring [21], has explored how models can be acquired by demonstration. However, by using a simulated learning system to induce general rules form the demonstrations more general models can be acquired more efficiently. For example, the use of SimStudent as authoring tool is still experimental,

but there is evidence that it may accelerate the authoring process and produce more accurate cognitive models than hand authoring. One demonstration explored the benefits of a traditional programming by demonstration approach to authoring in SimStudent versus a programming by tutoring approach [2]. In the latter, SimStudent asks for demonstrations only at steps where it has no relevant productions. Otherwise, it performs a step and asks the author for feedback as to whether the step is correct or not. Programming by tutoring was found to be much faster than programming by demonstration (77 minutes vs. 238 minutes) and produced a more accurate cognitive model whereby there were fewer productions that produced over-generalization errors. Programming by tutoring is now the standard approach because of its improved efficiency and effectiveness. Better efficiency is obtained because many author demonstrations are replaced by SimStudent actions with a quick yes-or-no response. Better effectiveness is obtained because these actions expose over-generalization errors to which the author responds "no" and the system learns new if-part preconditions to more appropriately narrow the generality of the modified production rule.

A second demonstration of SimStudent as an authoring tool [22] compared authoring in SimStudent with authoring example-tracing tutors in CTAT. Tutoring SimStudent has considerable similarity with creating an example-tracing tutor except that SimStudent starts to perform actions for the author, which can be merely checked as desirable or not, saving the time it otherwise takes for an author to perform those demonstrations. This study reported a potential savings of 43% in authoring time.

5 Evaluating a Simulated Learner as a Teachable Agent

Simulated learner systems can be more directly involved in helping students learn when they are used as a teachable agent whereby students learn by teaching [cf., 23]. Evaluating the use of an SL in this form ideally involves multiple steps. One should start with an SL that has already received some positive evaluation as a good model of student learning (see section 2). Then incorporate it into a teachable agent architecture and, as early and often as possible, perform pilot students with individual students [cf., 24 on think aloud user studies) and revise the system design. Finally, for both formative and summative reasons, use random assignment experiments to compare student learning from the teachable agent with reasonable alternatives.

Using SimStudent, we built a teachable agent environment, called APLUS, in which students learn to solve linear equations by teaching SimStudent [25]. To evaluate the effectiveness of APLUS and advance the theory of learning by teaching, we conducted multiple *in vivo* experiments [25,26,27,28]. Each of the classroom studies have been randomized controlled trials with two conditions varying one instructional approach. In one study [25], the self-explanation hypothesis was tested. To do so, we developed a version of APLUS in which SimStudent occasionally asked "why" questions. For example, when a student provided negative feedback to a step SimStudent performed, SimStudent asked, "Why do you think adding 3 here on both sides is incorrect?" Students were asked to respond to SimStudent's questions either by selecting pre-specified menu items or entering a free text response. The results showed that

the amount and the level of elaboration of the response had a reliable correlation with students' learning measured by online pre- and post-tests.

6 Conclusion

We outlined four general purposes for simulated learners (see Table 1) and reviewed methods of evaluation that align with these purposes. To evaluate an SL as a precise theory of learning, one can evaluate the cognitive model that results from learning, evaluate the accuracy of error predictions as well as prior knowledge assumptions needed to produce those errors, or evaluate the learning process, that is, the changes in student performance over time. To evaluate an SL as an instructional test, one should not only evaluate the SL's accuracy as a theory of student learning, but should also perform human experiments to determine whether the instruction that works best for SLs also works best for human students. To evaluate an SL as an automated authoring tool, one can evaluate the speed and precision of rule production, the frequency of over-generalization errors and the fit of the cognitive models it produces. More ambitiously, one can evaluate whether the resulting tutor produces as good (or better!) learning than an existing tutor. Similarly, to evaluate an SL as a Teachable Agent, one can not only evaluate the system features, but also perform experiments on whether students learn better with that system than with reasonable alternatives.

Simulated learner research is still in its infancy so most evaluation methods have not been frequently used. We know of just one such study [29] that evaluated an SL as an instructional tester by following up a predicted difference in instruction with a random assignment experiment with real students. It used an extension of the ACT-R theory of memory to simulate positive learning effects of an optimized practice schedule over an evenly spaced practice schedule. The same experiment was then run with human students and it confirmed the benefits of the optimized practice schedule. Such experiments are more feasible when the instruction involved is targeting simpler learning processes, such as memory, but will be more challenging as they target more complex learning processes, such as induction or sense making [31].

The space of instructional choices is just too large, over 200 trillion possible forms of instruction [32], for a purely empirical science of learning and instruction to succeed. We need parallel and coordinated advances in theories of learning *and* instruction. Efforts to develop and evaluate SLs are fundamental to such advancement.

References

1. VanLehn (1991). Two pseudo-students: Applications of machine learning to formative evaluation. In Lewis & Otsuki (Eds.), *Adv Res on Comp in Ed* (pp. 17-26). Amsterdm: Els.
2. Matsuda, Cohen, Koedinger (2015). Teaching the teacher. *Int J of AI in Ed, 25*, 1-34.
3. Li, Matsuda, Cohen, Koedinger (2015). Integrating representation learning and skill learning in a human-like intelligent agent. *AI, 219,* 67-91.
4. Anzai, Y. & Simon (1979). The theory of learning by doing. *Psych Rev, 86 (2)*, 124-140.
5. Anderson, J.R., & Lebiere, C. (1998). *The Atomic Components of Thought*. Hillsdale: Erl.

6. Laird, Newell, & Rosenbloom (1987). Soar. *AI, 33*(1), 1–64.
7. Langley & Choi (2006). A unified cognitive architecture for physical agents. In *Proc of AI*.
8. Newell, Allen. 1990. *Unified Theories of Cognition*. Cambridge, MA: Harvard U. Press.
9. Tenenbaum, J. B., Griffiths, T. L., & Kemp.C. (2006). Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, *10*, 309-318.
10. MacLaren & Koedinger (2002). When and why does mastery learning work: Instructional experiments with ACT-R "SimStudents". In *Proc of ITS*, 355-366. Berlin: Spr-Ver.
11. Li, Stampfer, Cohen, & Koedinger (2013). General and efficient cognitive model discovery using a simulated student. In *Proc of Cognitive Science*. (pp. 894-9) Austin, TX.
12. Martin, Mitrovic, Mathan & Koedinger (2011). Evaluating and improving adaptive educational systems with learning curves. *User Modeling and User-Adapted Int, 21*(3), 249-283.
13. Stamper, J.C. & Koedinger, K.R. (2011). Human-machine student model discovery and improvement using data. In *Proc of AI in Ed*, pp. 353-360. Berlin: Springer.
14. Koedinger, Baker, Cunningham, Skogsholm, Leber, Stamper, (2010). A Data Repository for the EDM community: The PSLC DataShop. In *Hdbk of Ed Data Min*. Boca Rat: CRC.
15. Brown, J. S., & VanLehn, K. (1980). Repair theory. *Cognitive Science, 4,* 379-426.
16. Matsuda, Lee, Cohen, & Koedinger, (2009). A computational model of how learner errors arise from weak prior knowledge. In *Proc of Cognitive Science*. pp. 1288-1293.
17. Booth, J.L., & Koedinger, K.R. (2008). Key misconceptions in algebraic problem solving. In Love, McRae & Sloutsky (Eds.), *Proc of Cognitive Science,* pp. 571-576.
18. VanLehn, K. (1982). Bugs are not enough. *Journal of Mathematical Behavior, 3(2)*, 3-71.
19. Matz, M. (1980). Towards a process model for high school algebra errors. In Sleeman & Brown (Eds.), *Intelligent Tutoring Systems* (pp. 25-50). Orlando, FL: Academic Press.
20. Li, N., Cohen, W. W., & Koedinger, K. R. (2012). Problem Order Implications for Learning Transfer. In *Proceedings of Intelligent Tutoring Systems*, 185–194.
21. Aleven, V., McLaren, B., Sewall, J., & Koedinger, K. R. (2009). Example-tracing tutors: A new paradigm for intelligent tutoring systems. *Int J of AI in Education, 19,* 105-154.
22. MacLellan, Koedinger & Matsuda (2014). Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality. *Proc of Intelligent Tutoring Systems,* 551-560.
23. Biswas, G., Schwartz, D., Leelawong, K., Vye, N. (2005). Learning by Teaching: A New Agent Paradigm for Educational Software. *Applied Artificial Intelligence, 19*, 363-392.
24. Gomoll, K., (1990). Some techniques for observing users. In Laurel B. (ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley, Reading, MA, pp. 85-90.
25. Matsuda, Yarzebinski, Keiser, Raizada, William, Stylianides & Koedinger (2013). Cognitive anatomy of tutor learning. *J of Ed Psy, 105(4)*, 1152-1163.
26. Matsuda, Cohen, Koedinger, Keiser, Raizada, Yarzebinski, Watson, Stylianides (2012). Studying the effect of tutor learning using a teachable agent. In *Proc of DIGITEL*, 25-32.
27. Matsuda, Griger, Barbalios, Stylianides, Cohen, & Koedinger (2014). Investigating the effect of meta-cognitive scaffolding for learning by teaching. In *Proc of ITS,* 104-113.
28. Matsuda, Keiser, Raizada, Tu, Stylianides, Cohen, Koedinger (2010). Learning by Teaching SimStudent: In *Proceedings of Intelligent Tutoring Systems,* 317-326.
29. Ritter, Anderson, Koedinger, & Corbett (2007). Cognitive tutor: Applied research in mathematics education. *Psychonomic Bulletin &Review, 14(2)*, 249-255.
30. Pavlik & Anderson (2008). Using a model to compute the optimal schedule of practice. *Journal of Experimental Psychology: Applied, 14*, 101-117.
31. Koedinger, Corbett, Perfetti (2012). The KLI framework. *Cog Sci, 36 (5),* 757-798.
32. Koedinger, Booth, Klahr (2013). Instructional complexity. *Science, 342*, 935-937.

Simulated learners in peers assessment for introductory programming courses

Alexandre de Andrade Barbosa[1,3] and Evandro de Barros Costa[2,3]

[1] Federal University of Alagoas - Arapiraca Campus, Arapiraca - AL, Brazil
[2] Federal University of Alagoas - Computer Science Institute, Maceio - AL, Brazil
[3] Federal University of Campina Grande, Campina Grande - PB, Brazil

{*alexandre.barbosa@arapiraca.ufal.br*, *ebc.academico@gmail.com*}

Abstract. Programming is one of the basic competences in computer science, despite its importance, it is easy to find students with difficulties to understand the concepts required to use this skill. Several researchers report that the impossibility to achieve a quick and effective feedback, is one of the motivators for the problematic scenario. The professor, even when helped by the TAs, is not able to perform the reviews quickly, for this activity requires a huge amount of time. Fast feedback is extremely important to enable the learning of any concept. Some researches suggest the use of peer assessment as a means of providing feedback. However, it is quite common that the feedback provided by peers is not adequate. In this paper, we propose the use of simulated learners in a peer assessment approach as part of the teaching and learning processes of programming. Currently a software tool is being developed to include the proposal described in this paper.

1 Introduction

Programming is one of the basic competences in computer science, it is the basis for the development of several other competences required for professionals in the area. However, despite its importance, it is easy to find students who are demotivated and with difficulties to understand the concepts required to use this skill [7]. These difficulties causes a large number of failures, dropouts or the approval of students without the required level of knowledge [14] [6] [5].

Many factors are identified in literature as causing the problematic scenario related to programming courses. Several researchers report that the impossibility to achieve a quick, effective and individualized feedback, is one of the motivators for the problematic scenario [10] [12]. An individual follow up is impossible due to many students enrolled in the courses. In addition, there is a great complexity involved in the evaluation of a program, for it is necessary to understand how the programmer has developed the algorithm, so the professor needs to comprehend the line of reasoning adopted by the student. In this way, the professor, even when helped by the TAs, cannot provide an adequate and fast feedback about the solutions created by the students. This activity will require a huge amount

of time to manually open the code, compile, run and verify the output of every students solution for programming assignment. If the grading depends on the structure and the quality of code, in addition to program output correctness, the situation is a lot worse. Traditionally the real comprehension state of the contents of a programming course is known only months after the beginning of the course, when an evaluation activity is performed. After an evaluation it may be too late to make any intervention.

Fast feedback is of extreme importance to enable the learning of any concept [12]. Thus, some researches have been developed with the aim to propose methods and tools to facilitate the monitoring of the activities of students in programming courses. Some of these researches, such as [9][11][13], suggests the use of peer assessment as a means of providing fast and effective feedback. This solution is broadly used in Massive Open Online Courses (MOOCs), as described in [3][8], where the courses are applied to hundreds or thousands of people enrolled in them, and just as occurs in the context of programming, it is impossible for the professor to evaluate each solution. However, the peer assessment approach as a means of providing feedback has some problems. Many times the feedback provided by peers is not adequate, because the results are often not similar to the analysis of an expert [8]. It is quite common to find comments that are summarized to a phrase of congratulation or critique.

The reasons related to lack of effectiveness of feedback provided are quite distinct, these may occur due to poor understanding of the content of the activity, because of the student's low motivation, or due to the short time that one has available for the activities.

In [2] paper, it was observed the impact of learning was observed when a student is influenced by the performance of their peers, the authors describe that some students are encouraged to perform better, but others experiencing the same situations end up discouraged to perform better.

In this paper is proposed the use of simulated learners in a peer assessment approach used as part of the teaching and learning processes of programming. Two concerns are explored in this proposal: the first is related to the search of methods that enable a positive influence between students; the second concern is related to an approach that allows a less costly way of testing any proposal of applicability of peer assessment approach.

This paper is divided in five sections. In Section 2 the concept of peer assessment is presented. Observations on the implementation of peer assessment in a programming course context are shown in Section 3. The proposal of using simulated learners in the context of peer assessment for introductory programming is presented in Section 4. Finally the conclusions and future work are shown in the last section.

2 Peer Assessment

Peer assessment, or peer review, is an evaluation method where students have responsibilities that traditionally belong to professors only. Among these respon-

sibilities there are the review and the critique of the solutions proposed by their peers. This way, they can experience the discipline as students and also from the perspective of a TA. Usually in a peer assessment environment, students also conduct self assessment. This way, they can reflect on their solution when compared to other solutions, develop their critical thinking skills and improve understanding of the concepts covered in the course.

In traditional approach, the professor, even when helped by TAs, can not provide fast and adequate feedback for each solution proposed by the students. The comments provided by the professor are generic observations based on observation of all students solutions.

In accordance with [9], peer review is a powerful pedagogical method, because once students need to evaluate the work of their peers, they begin to teach and learn from each other. Thus, the learning process becomes much more active, making the learning qualitatively better than the traditional approach. Sstudents can spend more time on analysis and construction of their comments, creating more particular descriptions on a given solution and enriching discussion about the topic studied.

Thus, the use of peer review can reduce the workload on the professor, permitting the professor to focus on other pedagogical activities [9][3]. This evaluation approach can also enable the evaluation of large-scale complex exercises, which can not be evaluated in a automatically or semi-automatic fashion [3][8].

The success of peer assessment approach is strongly influenced by the quality of feedback provided. However, this feedback if often not adequate, the results are often not similar to the analysis of an expert [8]. In [8] is described that in many cases the evaluations of the students are similar to the TAs evaluation, however there are situations where the evaluations are graded 10% higher than the TAs evaluation, in extreme cases the grades could be 70% higher than the TAs evaluation. In [3] is mentioned that in general, there is a high correlation between the grades provided by students and TAs, but often in the evaluations from students the grades are 7% higher than the grades given by TAs.

Thus, we can conclude that peer assessment approach is a promising evaluation method, however there are improvements and adjusts to be applied to obtain richer discussions and more accurate assessments.

3 Peer Assessment in introductory programing courses

Human interaction is described as an essential feature for learning in many domains, including the introductory programming learning [13]. In classroom programming courses the contact between students occurs on a daily basis, allowing, for example, the discussion of the problems presented in the exercise lists, the developed solutions and the formation of groups for the projects of the course. This contact is many times inexistent in online programming courses, interactions in this environment are the human-machine type. Thus, using the peer assessment approach may enable human interaction on online courses, or enhance the interaction between humans in presencial classroom courses.

To encourage the assimilation of the topics, the use of practical exercises is quite common in programming courses, the practice of programming skills is crucial for learning. Many researchers also argue that the programming learning involves the reading and understanding of third-party code. Through peer assessment approach both characteristics can be obtained. The professor can develop new exercises, or choose problems proposed by others, while students will have to observe, understand and evaluate the codes of their peers, as well to compare these codes with their solution.

In [11] the use of a peer assessment approach to the context of programming courses is described, this approach is supported by a web application. The results described on the paper have a high correlation between the evaluations of the TAs and students, the correlation is lowest when the complexity of the exercise is higher.

An approach of peer assessment evaluation for the context of programming learning, also supported by a web application is presented in [9]. Five activities where graded using peer assessment, the occurrence of conflicts ranged from 61 % to activity with a lower incidence of conflict, up to 80 % for the activity with the highest occurrence of conflicts. The system considers that a conflict occurs when the student does not agree with the assessment provided.

In [9] the authors describes that if the peer reviews are conducted in an inadequate way, the failure rates can increase. For the teaching approach used at the programming course described in [13] there are two types of activities that require assessment, quizzes and mini projects. Among these activities only the mini projects are evaluated through peer assessment. Thus, students are not overloaded and the approach can be used appropriately.

Another problem that can emerge with the use of peer review in a programming context, is the increase of plagiarism. Once the assessment activity will be distributed among the students, the similarities of self-identification of codes can become more complicated. However, solutions are widely used to carry out the detection automatically similarities, such as MOSS [1] e GPLAG [4].

4 Simulated learners as peers in a peer assessment environment for introductory programing courses

In previous sections the advantages and disadvantages associated with the use of peer assessment in a general context, and when applied to the context of programming courses have been described. In both cases, the success of the approach is strongly influenced by the quality of the feedback given. Therefore, it is necessary to identify situations where there is inadequate feedback as well as conflict situations. Situations where inadequate feedback occurs are when, for any reason, the feedback does not help in the learning process. Conflict situations occur when the student does not agree with the assessment provided, or when there are huge variations on the evaluations provided. To perform a validation of this proposal or of any proposal involving peer assessment, it is necessary to

allocate the resources of time, physical space and adequate human resources. Thus, it can be said that the test of this approach is a costly activity.

Two concerns are explored in this proposal: how we can achieve methods that enable a positive influence between students in peer assessment environments, in other words, how a student can give a high quality feedback to their peers; and how a peer assessment approach can be tested with a lower cost, since any validation of these assessment approaches requires a huge amount of resources.

4.1 A scenario of use of peer assessment with simulated learners

Traditionally in a peer assessment environment, the professor must create the assignment and a set of assessment criteria. Then students develop their solutions observing the assessment criteria and submitting the solution to be evaluated by their peers. Each student's evaluation must meet the assessment criteria. The students should provide comments to peers explaining the reasons associated to the outcome and a grade or an evaluation concept (eg. A-, B+, C). Each student will have their code evaluated by their peers, and should assess the codes of other students. In Figure 1, it is illustrated the scenario previously described. There are variations in ways peer assessment approach is used, the scenario just mentioned has many characteristics which are similar to all the variations.

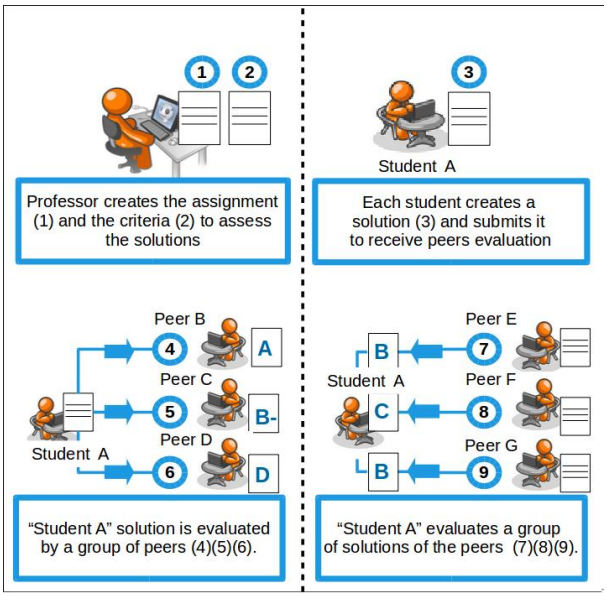

Fig. 1. A traditional peer assessment environment

In any peer assessment approach, it is possible to adopt pairing algorithms. Thereby, it is assured that evaluations are conducted by students with different

levels of knowledge. A student with low understanding of the subject will not be allocated to evaluate the work of another student in the same situation. Students with difficulties can clarify their doubts, while students with good understanding of the content should provide a good argumentation about their knowledge. However, it is not possible to ensure that a student evaluates the code that is the ideal for his/her learning and level of knowledge. As an example, in Figure 1, it is not possible to know if student "A" code is the best for peers "B", "C" and "D".

When a student does not agree with the evaluation provided by their peers, he/she will be able to request the intervention of the professor. This conflict situations are identified in [9]. However, in traditional peer assessment approach is not possible to identify incorrect evaluations provided by a student, or students that create biased evaluations only to help their fellows. As an example, in Figure 1, it is possible to see that different grades were given, but it is not possible to determine if the correct evaluations were given by peer "B", "C" or "D".

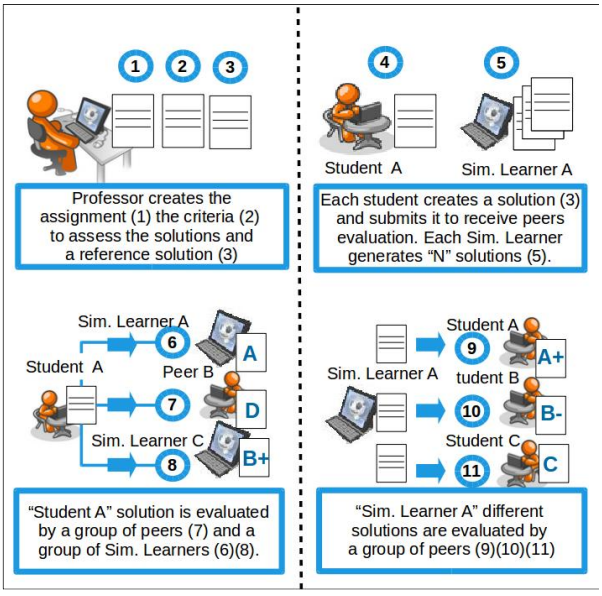

Fig. 2. A peer assessment environment using simulated learners

In a peer assessment environment that uses simulated learners, it is possible to solve the previous problems. As in traditional approach, the professor must create the assignment and a set of assessment criteria; in addition to that, he/she should provide a reference solution. Then, students develop their solutions, observing the assessment criteria and submitting the solution to be evaluated by their peers. At the same time, once a pairing algorithm can perform pairing of

the evaluators, each simulated learner must generate a code that is ideal for the learning and appropriate to the level of knowledge of each one of their peers, in this case, real students. Each student will have their code evaluated by their peers and by simulated students, and they should assess codes of other students, and codes of simulated students. In Figure 2 it is illustrated peer assessment environment with simulated learners. As an example, in Figure 2, it is possible to see that the simulated learner "A" generates a set of codes that are ideal for each student: "A", "B" and "C".

The identification of incorrect evaluations provided by a student, as well as students, who perform biased evaluations, could be carried out through the comparison of student's evaluations and the simulated student's evaluations. As an example, in Figure 2, it is possible to see that student "B", made an evaluation that is very different from the simulated learner's evaluations. In this way, it is possible to verify if the student did not understand the solution, or if the evaluation was created to help their fellows only.

Providing useful solutions to student learning A useful solution for the student learning does not always match the presentation of a correct and efficient code. Within the context of peer review may be more useful to display an incorrect code, as a way to make students to provide a set of review observations. To identify which type of code is best for a student; simulated learners can consult the representation of their cognitive status. In that way, it will be possible to the simulated learner identify the student misconceptions and errors in previous assignments, and generate variations of the reference solution that suits best for the student. Since multiple simulated students will be used, the codes that will be shown to students can range from efficient, correct and complete solutions to incorrect and/or incomplete solutions. Like that, it will be possible to check if students have different skills related to the content. To generate the variations from the reference solution, it is possible to combine testing techniques, such as mutant generation. Each code can be generated through the use of data related to the most common student's mistakes, emulating these behaviors and creating codes that are useful to learning. Once the research is in a preliminary stage, it is still not clear which artificial intelligence approaches should be used on the implementation of simulated students behaviors.

Assessment of students solutions Unlike what occurs in other contexts, for programming the evaluation of a solution can be automated or semi-automated. Typically a set of unit tests is applied to the code proposed by a student, who receives an indication that his/her code may be correct or incorrect, but no hint or comment is provided. Some researchers have investigated the use of different techniques to help assessment of codes and provide some guidance; these techniques usually employ software engineering metrics. Thus, simulated learners must be able to identify which subset of metrics can be used to perform the evaluation of the proposed solution for a student. The simulated learner should select the set of metrics that fits best to the objectives of the assignment and

the level of understanding that the student has at that moment. For each level of learning the same student can learn better if the set of metrics is properly selected. Each simulated student will use different strategies to evaluate the solutions provided by real students. Therefore, a variation between evaluations of simulated students is expected to occur. If an evaluation provided by a student has a very large variation in relation to the set of evaluations of simulated students, it will be necessary to investigate the motivation of this disparity. An acceptable variation threshold can be used to identify incorrect evaluations provided by students.

Discussing assessment criterias Once software engineering metrics were used in the evaluation, the explanation given by the simulated learner throughout the presentation of a set of metrics, is associated to the explanation of the metric choice and, possibly, of the snippet of the code where the observation is pertinent. Thereby, the simulated learner can help the professor to identify inadequate feedback, whenever an evaluation of a student is very different from the evaluation of a simulated learner, the professor and his tutors can then intervene.

4.2 Validation of peer assessent using simulated learners

Any validation of peer assessment approaches requires lots of physical space and a huge amount of human resources. As an example, if a validation of a pairing algorithm has to be done, it will be necessary to use a set of N students; this set must allow the creation of different profiles for evaluation of the pairing alternatives. The greater the possibilities of matching, the greater the amount of students required. Through the use of simulated learners any operational proposal of peer assessment can be tested at a much lower cost, since the physical space and human resources are drastically reduced. The researcher can determine how much of human resource will be available, replacing the students with simulated students. The researcher can also specify the desired behavior of students; the simulated students should emulate students with a high degree of understanding of the contents or with low understanding. After obtaining initial results with the use of simulated learners, the number of human individuals participating in an experiment can be increased, since it may be interesting to obtain a greater statistical power associated with the conclusions.

5 Conclusions and further work

In this paper, we have proposed the use of simulated learners in a peer assessment approach adopted as a support part of a programming course. The use of simulated learners as presented in this proposal aims to two goals: influence the students to provide better quality feedback; and allow for a less costly validation for peer assessment applied to programming contexts.

The research associated with the proposal presented in this paper is in a preliminary stage. Thus, the effectiveness of this proposal will be further evaluated in controlled experiments executed in the future. An open source software tool is being developed to include all aspects described throughout this proposal.

References

1. Bowyer, K., Hall, L.: Experience using "moss" to detect cheating on programming assignments. In: Frontiers in Education Conference, 1999. FIE '99. 29th Annual. vol. 3, pp. 13B3/18–13B3/22 (Nov 1999)
2. Frost, S., McCalla, G.I.: Exploring through simulation the effects of peer impact on learning. In: Proc. of the Workshops at the 16th International Conference on Artificial Intelligence in Education AIED 2013, Memphis, USA, July 9-13, 2013 (2013), http://ceur-ws.org/Vol-1009/0403.pdf
3. Kulkarni, C., Wei, K.P., Le, H., Chia, D., Papadopoulos, K., Cheng, J., Koller, D., Klemmer, S.R.: Peer and self assessment in massive online classes. ACM Trans. Comput. Hum. Interact. 20(6), 33:1–33:31 (Dec 2013)
4. Liu, C., Chen, C., Han, J., Yu, P.S.: Gplag: Detection of software plagiarism by program dependence graph analysis. In: Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 872–881. KDD '06, ACM, New York, USA (2006)
5. Mason, R., Cooper, G.: Introductory programming courses in australia and new zealand in 2013 - trends and reasons. In: Proc. of the Sixteenth Australasian Computing Education Conference - Volume 148. pp. 139–147. ACE, Darlinghurst, Australia (2014)
6. Mason, R., Cooper, G., de Raadt, M.: Trends in introductory programming courses in australian universities: Languages, environments and pedagogy. In: Proc. of the Fourteenth Australasian Computing Education Conference - Volume 123. pp. 33–42. ACE, Darlinghurst, Australia (2012)
7. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.D., Laxer, C., Thomas, L., Utting, I., Wilusz, T.: A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. In: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education. pp. 125–180. ITiCSE-WGR, New York, USA (2001)
8. Piech, C., Huang, J., Chen, Z., Do, C.B., Ng, A.Y., Koller, D.: Tuned models of peer assessment in moocs. CoRR abs/1307.2579 (2013)
9. de Raadt, M., Lai, D., Watson, R.: An evaluation of electronic individual peer assessment in an introductory programming course. In: Proc. of the Seventh Baltic Sea Conference on Computing Education Research - Volume 88. pp. 53–64. Koli Calling, Darlinghurst, Australia (2007)
10. Singh, R., Gulwani, S., Solar-Lezama, A.: Automated feedback generation for introductory programming assignments. In: Proc. of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 15–26. PLDI, New York, USA (2013)
11. Sitthiworachart, J., Joy, M.: Computer support of effective peer assessment in an undergraduate programming class. Journal of Computer Assisted Learning 24(3), 217–231 (2008)
12. Stegeman, M., Barendsen, E., Smetsers, S.: Towards an empirically validated model for assessment of code quality. In: Proc. of the 14th Koli Calling Int. Conf. on Computing Education Research. pp. 99–108. Koli Calling, New York, USA (2014)

13. Warren, J., Rixner, S., Greiner, J., Wong, S.: Facilitating human interaction in an online programming course. In: Proc. of the 45th ACM Technical Symposium on Computer Science Education. pp. 665–670. SIGCSE, New York, USA (2014)
14. Yadin, A.: Reducing the dropout rate in an introductory programming course. ACM Inroads 2(4), 71–76 (2011)

Simulated Learners for Testing Agile Teaming in Social Educational Games

Steeve Laberge and Fuhua Lin

School of Computing and Information Systems, Athabasca University, Edmonton, Canada
slaberge@acm.org, oscarl@athabascau.ca

Abstract. This paper proposes an approach for creating and testing an multiagent systems based adaptive social educational game (SEG), QuizMASter, using the concept of simulated learners to overcome experimentation complexity and unpredictable student availability, as is typical with online learning environments. We show that simulated learners can play two roles. First, it can be used for testing the game planning, scheduling and adaptive assessment algorithms. With some degree of success met with our initial experimentation with QuizMASter, advanced planning and coordination algorithms are now needed to allow the game-based assessment platform to realize its full potential. The multi-agent system approach is suitable for modeling and developing adaptive behaviour in SEGs. However, as we have found with our early prototypes, verifying and validating such a system is very difficult in an online context where students are not always available. MAS-based assessment game planning and coordination algorithms are complex and thus need simulated learners for testing purposes. Second, to overcome unpredictable student availability, we modeled QuizMASter as a new class of socio-technical system, human-agent collective (HAC). In the system, human learners and simulated learners (smart software agents) engage in flexible relationship in order to achieve both their individual and collective goals, while simulated learners are selected for serving as virtual team members.

Keywords: social educational agents, multiagent systems, simulated learners

1 Introduction

For decades, educational games have proven to be an effective means to motivate learners and enhance learning. Social (multi-player) educational games (SEGs) offer many opportunities to improve learning in ways that go beyond what a single-player game can achieve because SEGs allow players to be social, competitive, and collaborative in their problem solving. The presence of other players can be used to increase playability and to help teach team-work and social skills. SEGs promote intragroup cooperation and intergroup competition [1]. However, existing SEGs share many of the shortcomings of classroom role-playing. Setting

up existing SEGs is logistically challenging, expensive, and inflexible. Furthermore, players become bored after going through existing SEGs once or twice.

To test such a social educational game, we face two difficulties. One is how to test the planning and scheduling algorithms. Another is how to meet the need of agile team formation. In SEGs, group formation has big impact on group learning performance. Poor group formation in social games can result to homogeneity in student characteristic such that the peer learning is ineffective. Thus, there is a need to constitute a heterogeneous group SEGs that constitutes students with different collaborative competencies and knowledge levels. However, without empirical study it becomes difficult to conclude which group characteristics are desirable in the heterogeneity as different game-based learning needs may require different group orientations. Previous research has focused on various group orientation techniques and their impact on group performance like different learning styles in group orientation [2–4]. However, there is need to investigate the impact of other group orientation techniques on group performance like grouping students based on their collaboration competence levels. Furthermore, most of the previous research in group-formation focuses on classroom based learning. Also, it lacks the true experiment design methodology that is recommended when investigating learning outcomes from different game-based learning strategies. Simulated learners methodology [5] has shown a promising way to solve these challenges.

In this paper, we show that simulated learners can play two roles. First, it can be used for testing the game planning, scheduling and adaptive assessment algorithms. Second, working with human learners and forming human-agent collectives (HAC), simulated learners serve as virtual team members to enable asynchronous game-based learning in a context where student availability is unpredictable. This paper is structured as follows: In Section 2 we discuss recent advancements and related work. Section 3 describes QuizMASter. Section 4 presents the proposed architecture for development of QuizMASter. Section 5 explains how we intend to use simulated learners for testing QuizMASter. Finally, Section 6 concludes.

2 Related Work

Researchers have found that learning can be more attractive if learning experiences combine challenge and fun [6]. As social networks have become popular applications, they have given rise to social games. This kind of game is played by users of social networks as a way to interact with friends [7] and has become a part of the culture for digital natives. Social games have unique features that distinguish them from other video games. Those features are closely linked with the features of social networks [8]. Social games can make a contribution to social learning environments by applying game mechanics and other design elements, 'gamifying' social learning environments to make them more fun and engaging. For games to be effective as a learning tool, a delicate balance must be maintained between playability and educational value [9, 10], and between

game design and learning principles. Methods have been proposed for making valid inferences about what the student knows, using actions and events observed during gameplay. Such methods include evidence-centered-design (ECD) [11, 12]; the learning progressions model [13], the ecological approach to design of e-learning environments [14], stealth assessment [15], game analytics [16], and learning analytics [17]. Most of the new concepts target an ever-changing learning environment and learner needs, as today's education moves toward a digital, social, personalized, and fun environment. Moreover, as is the case for all competitive games, an equal match between players is essential to self-esteem and to maintain a high degree of player interest in the game. Hence, we need mechanisms and models that can aggregate the current performance and preferences of players, and accurately predict student performance in the game. Software agents have been used to implement consistent long-term intelligent behaviour in games [18], multi-agent collaborative team-based games [19], and adaptive and believable non-player character agents simulating virtual students [20]. The use of agent technologies leads to a system characterized by both autonomy and a distribution of tasks and control [21]. This trend has two aspects. First, game-based learning activities should be carefully orchestrated to be social and enjoyable. Second, game scheduling and coordination should be highly adaptive and flexible. However, nobody has yet developed models, algorithms, and mechanisms for planning, scheduling, and coordination that are suitable for creating and testing SEGs.

3 QuizMASter

QuizMASter is designed to be a formative assessment tool that enables students to be tested within a multi-player game [22]. Two or more students simultaneously log in remotely to the system via a Web-based interface. Each student is represented by one avatar in this virtual world. Students are able to view their own avatar as well as those of their opponents.

Each game has the game-show host who is also represented by an avatar visible to all contestants [22]. The game-show host poses each of the game questions to all the contestants. The students hear the voice of the host reading each question and view them displayed on their screens. They individually and independently from one another answer each question by, for instance, selecting an answer from available choices in a multiple-choice format. Each correct answer would receive one mark. Figure 1 shows a screen shot of QuizMASter.

3.1 Characteristics of QuizMASter

The environment for QuizMASter has the following characteristics:

Flexibility. The environment for QuizMASter needs flexibility for game enactment, to be able to cope with dynamic changes of user profiles, handle fragmentation of playing and learning time needed to accomplish activities and tasks,

Fig. 1. QuizMASter in Open Wonderland

adequately handle exceptional situations, predict changes due to external events, and offer sufficient interoperability with other software systems in educational institutions. Individual learners have particular interests, proficiency levels, and preferences that may result in conflicting learning goals.

Social ability and interactivity. The environment for QuizMASter should encourage interaction and collaboration among peers, and should be open to participation of students, teachers, parents, and experts on the subjects being taught. Web 2.0 has had a strong influence on the ways people learn and access information, and schools are taking advantage of this trend by adopting social learning environments. One way to engage learners in a collaborative production of knowledge is to promote social rewards.

User control. One of the most desirable features of social education games is to empower players with control over the problems that they solve. For example, in QuizMASter, students, parents, and teachers can design new rules to create their own games and modify the game elements to fit different knowledge levels.

Customization. Customization is a core principle that helps accommodate differences among learners [23]. Teachers could build a QuizMASter that has its own style and rules to determine the game's level of difficulty, to gear the game for specific goals or a specific group of learners. Some teachers may be interested in sharing collections of rules to fit the learning and play styles of their students. Like teachers, learners/players can be co-creators of their practice space through building new game scenarios, creating their own rules, sharing their strategies and making self-paced challenges [23].

4 The Proposed Architecture

Multi-agent technologies are considered most suitable for developing SEGs as it will lead to systems that operate in a highly dynamic, open, and distributed environment. In an MAS-based SEG, each learner/player is represented as an autonomous agent, called learner agent. MAS technologies, such as goal orientation and the Belief-Desire-Intention (BDI) paradigm, is used as the foundation for the agent architecture. These learner agents are able to reason about the learning goals, the strengths and weaknesses of learners and update the learner models.

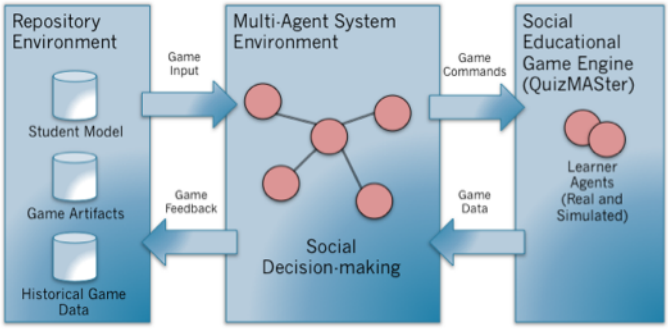

Fig. 2. Architecture for MAS-Based Social Educational Game Environment

Whenever a learner enters the system to play a social educational game, the learner agent will retrieve her/his learner model and acquire preferences about the current game-playing, and then send to a game management agent (GMA) of the system. The GMA is designed for setting up and maintaining teams for the system. The GMA will assign the learner to participate in a most suitable team that is undermanned according to the profile and preferences of the learner. The team will be configured in accordance with the game model by the GMA. Once the team has been completely formed, the GMA will create a game scheduling agent (GSA), a game host agent (GHA), and an assessment agent (AA) for each team. The GSA will continuously generate a game sequence dynamically adapted to the team's knowledge level (represented as a combined learner model [24]. The GHA will receive the game sequence from the scheduling agent and execute game sequence with the learners in the team. It will also be responsible for capturing data about learner/player performance. The AA will receive and interpret game events and communicate with the learner agents to update the learner model as necessary.

The GSA will dynamically schedule the game on the fly through interacting with other agents with a coordination mechanism, considering both the current world state and available resources, and solving conflicts in preferences and learning progression between the agents. The goal of the GSA is to optimize

the playability and educational values. We will model the game elements as resources. To solve the distributed constraint optimization problem, we are developing multiagent coordination mechanisms and scheduling algorithms to be used by the GSA.

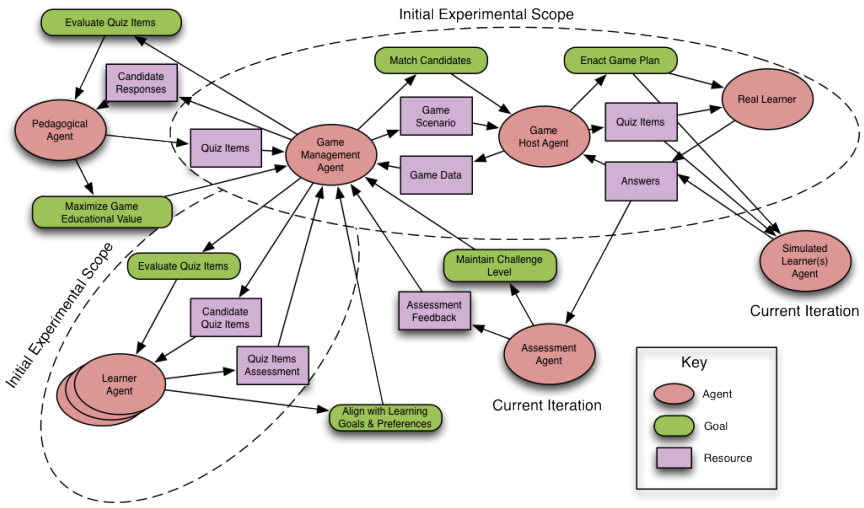

Fig. 3. MAS-Based SEG Agent Interaction Model

4.1 Planning and Scheduling Algorithms

The planning algorithms refer to the (local) planning algorithm of learner agents. To develop planning algorithms for learner agents, the following supporting models have been taken into consideration: (i) Learner models that accumulate and represent beliefs about the targeted aspects of skills. They are expressed as probability distributions for competency-model variables (called nodes) describing the set of knowledge and skills on which inferences are to be based. (ii) Evidence models that identify what the learner says or does, and provide evidence about those skills that express how the evidence depends on the competency-model variables in a psychometric model. (iii) Task/action models that express situations that can evoke required evidence. To design an action model, we adopt a model called Fuzzy Cognitive Goal Net [25] as the planning tool by combining the planning capability of Goal Net and reasoning ability of Fuzzy Cognitive Maps (FCMs). These FCMs give the learner agent a powerful reasoning ability for game context and player interactions, giving the task model accurate context awareness and learner awareness. We are developing coordination mechanisms

for the GMA and the GSA to solve the problem of team formation, scheduling and coordination in a highly flexible and dynamic manner. We considered the following concepts or methods:

(i) Contract-net protocols (CNPs) are used as a coordination mechanism by the GMA with a game model repository to timely form a team from all available players, using mutual selection and exchanging information in a structured way to converge on assignments. Each involved learner can delegate the negotiation process to its agent. These agents will strive to find a compromise team-joining decision obeying hard learning constraints while simultaneously resolving individual conflicts of interest.

(ii) The problem of scheduling and customizing a social educational game can be solved through social-choice-based customization. We view the SEG gameplay design as an optimization problem. Resources must be allocated through strategically scheduling, and coordinating a group of players according to their preferences and learning progressions. The constraints include key learning principles that inform the design of mechanics: challenge, exploration, risk taking, agency, and interactions [26-27]. The objective of the GSA is to maximize the learnability and engagement of the learners in the group. Social choice theory in MAS concerns the design and formal analysis of methods for aggregating preferences of multiple agents and collective decision-making and optimizing for preferences [28-29]. For example, we use a voting-based group decision-making approach such as Single Transferable Voting [30] to aggregate learner preferences and learning progression because it is computationally resistant to manipulation [31]. The purpose is to take information from individuals and combine it to produce the optimal result.

(iii) To support the need for dynamic decision making in the MAS-based SEG architecture, our current line of investigation is the concept of social choice Markov Decision Process (MDP) as recently proposed by Parkes and Procaccia [32]. In a social choice MDP, each state is defined by "preference profiles", which contain the preferences of all agents against a set of alternatives for a given scenario. The course of action from any given state is determined by a deterministic social choice function (the policy, in the context of the MDP) that takes into account the likelihood of transitions and their rewards. However, a preference profile is subject to change over time, especially in a live SEG context. For example, a learner that unexpectedly answers a question initially deemed beyond the learner's perceived level of comprehension would likely trigger a change of belief in the agents and potentially alter their ranking of alternatives. And since the number of alternatives in a SEG can be very large, the state space for any given SEG is huge, making the computation of optimal decision-making policies excessively difficult. We solve this problem by exploiting symmetries that exist in certain game types (e.g. in a quiz game SEG format, using a reduced set of question types that share common characteristics as a basis for alternatives as opposed to individual questions).

5 Simulated Learners

It is our view that the Belief-Desire-Intention (BDI) model is ideally suited for modeling and simulating learner behaviour. According to Jaques and Vicari (2007) [33], intelligent agents based on Bratman's Belief-Desire-Intention model, or BDI agents, are commonly used in modeling cognitive aspects, such as personality, affect, or goals. Píbil et al. (2012) claim BDI agent architecture is "a currently dominant approach to design of intelligent agents" [34]. Wong et al. (2012) describes the suitability of the BDI agent model for applications where both reactive behavior and goal-directed reasoning are required [35]. Soliman and Guetl (2012) suggest that BDI maps well onto models for pedagogically based selection of sub plans within a hierarchical planning strategy – "apprenticeship learning model" given as example [36]. They also talks about advantage of breaking plans down into smaller plans to allow for different "pedagogical permutations" allowing the agent to adapt to different learning styles, domain knowledge, and learning goals. Norling (2004) attributes the successful use of BDI agents for modeling human-like behavior in virtual characters to BDI's association to "folk psychology" [37]. This allows for an intuitive mapping of agent framework to common language that people use to describe the reasoning process. Of particular importance to this study is the way that implementations of the BDI architecture model long-term or interest goals. We have selected the Jason^{TM} [38] platform for providing multi-agent BDI programming in AgentSpeak.

A shortcoming of the BDI paradigm is that although it is intended to be goal-driven, in most implementations this means/amounts to using goals to trigger plans, but does not support the concept of long-term goals or preferences [39], such as a student's long term learning goals, or the pedagogical goals of a CA. They feel that these types of goals are difficult to represent in most BDI systems because they signify an ongoing desire that must be maintained over a long period of time compared to relative short goal processing cycles. It is left to the developer to implement this type of preference goal through the belief system of the agent, modifications to the platform or environment, or other methods of simulating long-term goals.

Hübner, Bordini, and Wooldridge (2007) describe plan patterns for implementing declarative goals, with varying levels of commitment in AgentSpeak [40]. Bordini et al. (2007) expand on this in their chapter on advanced goal-based programming [38]. While AgentSpeak and Jason support achievement goals, these patterns are intended to address the lack of support for "richer goal structures", such as declarative goals, which they feel are essential to providing agents with rational behaviour. Pokahr et al. (2005) point out that the majority of BDI interpreters do not provide a mechanism for deliberating about multiple and possibly conflicting goals [41]. It is worth noting that there are "BDI inspired" systems that are more goal-oriented, such as Practionist and GOAL [42]. The Jason multi-agent platform for BDI agents was selected for this project because it is a well-established open-source project that is being actively maintained. It supports both centralized and distributed multi-agent environments. Píbil et

al. (2012) describes Jason as "one of the popular approaches in the group of theoretically-rooted agent-oriented programming languages" [34]. A major advantage of Jason is that it is easy to extend the language through Java based libraries and other components. Internal actions can allow the programmer to create new internal functionality or make use of legacy object-oriented code [38]. However, Píbil et al. (2012) caution that the use of such extensions, if used too heavily, can make the agent program difficult to comprehend without understanding the functionality of the Java code [34]. They raise the concern that novice programmers have few guidelines for choosing how much to program in AgentSpeak, and how much too program in Java. The usefulness of being able to extend Jason can be demonstrated by two examples of current research into integrating BDI with Bayesian Networks. Modeling of some student characteristics requires a probabilistic model; Bayesian Networks (BN) being a popular choice in recent years [43-44]. Recent work by Kieling and Vicari (2011) describes how they have extended Jason to allow a BDI agent to use a BN based probabilistic model. Similarly, Silva and Gluz (2011) extend the AgentSpeak(L) language to implement AgentSpeak(PL) by extending the Jason environment. AgentSpeak(PL) integrates probabilistic beliefs into BDI agents using Bayesian Networks [45]. Experimentation with QuizMASter to date has enabled the modelling of simulated learners in virtual worlds with an initial focus on their appearance, gestures, kinematics, and physical properties [46]. Recent related research work in that area has been on the creation of engaging avatars for 3D learning environments [47]. Employing the theory of Transformed Social Interaction (TSI) [48], simulated learners were designed with the following abilities:

(i) Self-identification: The self-identification dimension of TSI was implemented using facial-identity capture with a tool called FAtiMA. Each of the users' face were morphed with their default avatar agent's face to capitalize on human beings' disposition to prefer faces similar to their own and general preference of appearing younger (see Fig. 4).

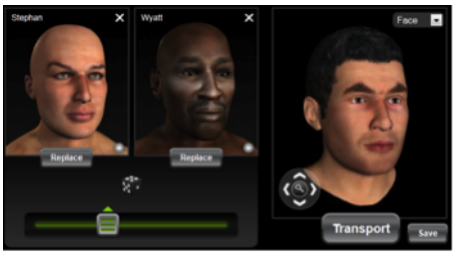

Fig. 4. Transformed Social Interaction – Image Morphing Technique

(ii) Sensory-abilities: Sensory-abilities dimension of TSI were implemented using a movement and visual tracking capability. The general challenge of sensory abilities implementation lies in two areas: the complexity of human senses and

the processing of sensory data of different modality and historicity. For the reason of simplicity, only visual tracking capability was exploited.

(iii) Situational-context: The situational-context dimension of TSI was implemented by using the best-view feature of Open Wonderland, whereby the temporal structure of a conversation can be altered.

The main idea of this research has been to explore the methodology for developing simulated learners for simulating and testing SEGs. That is, behind a simulated learner is an agent. Or we can say a simulated learner is an agent's avatar. All avatars, including real students' avatars and agent-based simulated learners, live in the virtual worlds, while the agents live in the multi-agent system. The integration of multi-agent systems with virtual worlds adds intelligence to the SEG platform and opens a number of extremely interesting and potentially useful research avenues concerning game-based learning. However, the advanced algorithms that support game planning, coordination and execution are difficult to test with real subjects considering the overhead involved is seeking authorization and the unpredictable availability of real life subjects in an online environment. This where an expanded view of simulated learners comes into play. The advantages of a simulated environment that closely approximates human behaviour include: (1) It allows for rapid and complete testing of advanced algorithms for game based adaptive assessment as well as SEG planning, coordination and execution in a simulated environment. The efficiency of the algorithms can be measured without first securing the availability of students; (2) With proper learner modeling and adaptive behaviour, simulated learners can engage with real life learners in friendly competitive games for the purpose of formative assessment, again working around the issue of availability of real students in an online learning environment.

6 Conclusions

As our recent experimentation suggests, many outstanding challenges must be addressed in developing intelligent SEGs. As we get closer to real world testing of our experimental game based assessment framework, we are faced with the complexity of enrolling real life learners in an e-learning environment and the variability that human interactions introduce in the measurement of adaptive algorithm efficiency. This is where we see the value of simulated learners. At this stage of our research, simulated learners have been rendered as Non Person Characters (NPCs) controlled by BDI agent running in the multi-agent system based virtual world. Our medium term goal is to extend the existing system to a particular learning subject (e.g., English language learning) to verify the effectiveness of the proposed virtual assessment environment and the benefit that students perceive from interacting with the proposed NPCs.

For simulated learners to be successful in our experimental framework, they must closely approximate the performance of real learners. The simple, pre-encoded behaviour we have implemented so far in the NPCs for QuizMASter will not suffice to demonstrate the efficiency of our adaptive algorithms and

allow for simulated learner agents to act as virtual players in our game based assessment framework. Current outstanding research questions within our group are:

1. How do we add intelligence and adaptive behaviour to the simulated learner agents while preserving our ability to obtain predictable and repeatable test results from our adaptive MAS framework?
2. How much autonomy can we afford to give to simulated learners in terms of independent thought and action, and to which degree should a simulated learner be able to adjust its behaviour as a function of its interactions with other agents, including real life learners?
3. How do we incorporate modern game, learning and assessment analytics in the supporting adaptive MAS framework in order to maximize the value of simulated learners as a means to perform non-intrusive, formative assessment?

References

1. Romero, M., Usart, M., Ott, M., Earp, J., de Freitas, S., and Arnab, S.: Learning through playing for or against each other. Promoting Collaborative Learning in Digital Game Based Learning. ECIS 2012 Proceedings. Paper 93 (2012)
2. Alfonseca, E., Carro, R. M., Martin, E., Ortigosa, A., and Paredes, P.: The impact of learning styles on student grouping for collaborative learning: a case study. User Modeling and User-Adapted Interaction, 16(3–4), 377–401 (2006)
3. Deibel, K.: Team formation methods for increasing interaction during in-class group work. In ACM SIGCSE Bulletin (Vol. 37, 291–295). Caparica, Portugal (2005)
4. Grigoriadou, M., Papanikolaou, K. A., and Gouli, E.: Investigating How to Group Students based on their Learning Styles. In In ICALT, 2006 1139–1140 (2006)
5. McCalla, G. and Champaign J.: AIED Workshop on Simulated Learners. AIED 2013 Workshops Proceedings, Volume 4 (2013)
6. Vassileva, J.: Toward social learning environments. IEEE Transactions on Learning Technologies, 1(4), 199–213 (2008)
7. Klopfer, E., Osterweil, S., and Salen, K.: Moving learning games forward: obstacles, opportunities and openness, the education arcade. MIT (2009)
8. Jarvinen, A.: Game design for social networks: Interaction design for playful dispositions. In Stephen N. Spencer (Ed.), Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games (Sandbox '09), 95–102. ACM, New York, NY, USA (2009)
9. Van Eck, R.: Building Artificially Intelligent Learning Games. In V. Sugumaran (Ed.), Intelligent Information Technologies: Concepts, Methodologies, Tools and Applications, 793-825 (2008)
10. Augustin, T., Hockemeyer, C., Kickmeier-Rust, M. D., and Albert, D.: Individualized Skill Assessment in Digital Learning Games: Basic Definitions and Mathematical Formalism. IEEE Trans. on Learning Technologies, 4(2), 138-147 (2011)
11. Mislevy, R. J., and Haertel, G. D.: Implications of evidence-centered design for educational testing. Educational Measurement: Issues and Practice, 25(4), 6–20 (2006)
12. Mislevy, R. J., Steinberg, L. S., and Almond, R. G.: On the structure of educational assessments. Measurement: Interdisciplinary Research and Perspectives, 1, 3-67 (2003)

13. Corcoran, T., Mosher, F. A., and Rogat, A.: Learning Progressions in Science: An Evidence-Based Approach to Reform (Research Report No. RR-63). Center on Continuous Instructional Improvement, Teachers College—Columbia University (2009)
14. McCalla, G.: The Ecological Approach to the Design of E-Learning Environments: Purpose-based Capture and Use of Information About Learners. Journal of Interactive Media in Education. (7), 1-23 (2004)
15. Shute, V. J.: Stealth assessment in computer-based games to support learning. Computer games and instruction. Charlotte, NC: Information Age Publishers, 503-523 (2011)
16. Long, P., and Siemens, G.: Penetrating the fog: analytics in learning and education. Educause Review Online 46 (5): 31–40 (2011)
17. El-Nasr, M. S., Drachen, A., and Canossa, A.: Game Analytics: Maximizing the Value of Player Data, Springer (2013)
18. Oijen, J., and Dignum, F.: Scalable Perception for BDI-Agents Embodied in Virtual Environments, Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on, Vol. 2, 46-53, 22-27 (2011)
19. Patel, P., and Hexmoor, H.: Designing BOTs with BDI agents, Collaborative Technologies and Systems, 2009. CTS '09. International Symposium on, 180-186 (2009)
20. Lim, M., Dias, J., Aylett, R., and Paiva, A.: Creating adaptive affective autonomous NPCs, Autonomous Agents and Multi-Agent Systems (AAMAS), Springer, 24(2), 287-311 (2012)
21. Sterling, L. S. and Taveter, K.: The art of agent-oriented modeling, MIT Press (2009)
22. Dutchuk, M., Muhammadi, K. A., and Lin, F.: QuizMASter - A Multi-Agent Game-Style Learning Activity, EduTainment 2009, Aug 2009, Banff, Canada, Learning by Doing, (eds.), M Chang, et al., LNCS 5670, 263-272 (2009)
23. de Freitas, S. and Oliver, M.: How can exploratory learning with game and simulation within the curriculum be most effectively evaluated? Computers and Education. 46(3), 249-264 (2006)
24. Shabani, S., Lin, F. and Graf, S.: A Framework for User Modeling in QuizMASter. Journal of e-Learning and Knowledge Society, 8(3), 1826-6223 (2012)
25. Jennings N. R., L. Moreau, D. Nicholson, S. Ramchurn, S. Roberts, T. Rodden, and A. Rogers: Human-agent collectives. Commun. ACM 57, 12 (2014), 80-88 (2014)
26. Cai, Y., Miao, C., Tan, A.-H., and Shen, Z.: Fuzzy cognitive goal net for interactive storytelling plot design. In Proceedings of the 2006 ACM SIGCHI Int. conf. on Advances in comput. entertainment technology. ACM, NY, USA, Article 56 (2006)
27. Gee, James P.: Good Video Games + Good Learning. New York: Peter Lang (2008)
28. Gee, James P.: Learning by Design: Games as Learning Machines, Interactive Educational Multimedia, Vol. 8, April Ed. 2004, 13-15 (2004)
29. Brandt, F., Conitzer, V., and Endriss, U.: Computational Social Choice, Chapter 6 of book edited by Weiss, G., Multiagent Systems (2nd edition), 213-283 (2013)
30. Conitzer, V.: Making Decisions Based on the Preferences of Multiple Agents, Communications of the ACM, 53(3), 84-94 (2010)
31. Bartholdi, J. J. and Orlin, J. B: Single Transferable Vote Resists Strategic Voting. Social Choice and Welfare, 8, 341-354 (1991)
32. Parkes, D. C. and Procaccia, A. D.: Dynamic Social Choice with Evolving Preferences. In M. desJardins and M. L. Littman (eds.), AAAI : AAAI Press (2013)
33. Jaques, P. A., Vicari, R. M.: A BDI approach to infer student's emotions in an intelligent learning environment. Computers & Education, 49(2), 360–384 (2007)

34. Píbil, R., Novák, P., Brom, C., Gemrot, J.: Notes on Pragmatic Agent-Programming with Jason. In L. Dennis, O. Boissier, & R. H. Bordini (Eds.), Programming Multi-Agent Systems, 58–73. Springer Berlin Heidelberg (2012)
35. Wong, W., Cavedon, L., Thangarajah, J., Padgham, L.: Flexible Conversation Management Using a BDI Agent Approach. In Y. Nakano, M. Neff, A. Paiva, & M. Walker (Eds.), Intelligent Virtual Agents, 7502, 464–470). Springer (2012)
36. Soliman, M., Guetl, C.: Experiences with BDI-based design and implementation of Intelligent Pedagogical Agents. In 2012 15th International Conference on Interactive Collaborative Learning (ICL) (1–5). Presented at the 2012 15th International Conference on Interactive Collaborative Learning (ICL) (2012)
37. Norling, E.: Folk Psychology for Human Modelling: Extending the BDI Paradigm. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1 (202–209). Washington, DC, USA: IEEE (2004)
38. Bordini, R. H., Hübner, J. F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. John Wiley & Sons (2007)
39. Bellotti, F., Berta, R., De Gloria, A., Lavagnino, E.: Towards a conversational agent architecture to favor knowledge discovery in serious games. In Proc. of the 8th Int. Conf. on Advances in Comput. Entertainment Technology, 17:1–17:7 (2011)
40. Hübner, J. F., Bordini, R. H., Wooldridge, M.: Programming Declarative Goals Using Plan Patterns. In M. Baldoni & U. Endriss (Eds.), Proceedings on the Fourth International Workshop on Declarative Agent Languages and Technologies, held with AAMAS 2006 (123–140). Springer Berlin Heidelberg (2007)
41. Pokahr, A., Braubach, L., Lamersdorf, W. (2005). A BDI architecture for goal deliberation. In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (1295–1296). New York, NY, USA: ACM (2005)
42. Braubach, L., Pokahr, A.: Representing Long-Term and Interest BDI Goals. In L. Braubach, J.-P. Briot, & J. Thangarajah (Eds.), Programming Multi-Agent Systems (pp. 201–218). Springer Berlin Heidelberg (2010)
43. Conati, C., Maclaren, H.: Empirically building and evaluating a probabilistic model of user affect. User Modeling and User-Adapted Interaction, 19(3), 267–303 (2009)
44. Chrysafiadi, K., Virvou, M.: Student modeling approaches: A literature review for the last decade. Expert Systems with Applications, 40(11), 4715–4729 (2013)
45. Silva, D. G., Gluz, J. C.: AgentSpeak(PL): A New Programming Language for BDI Agents with Integrated Bayesian Network Model. In 2011 International Conference on Information Science and Applications (ICISA) (pp. 1–7). Presented at the 2011 International Conference on Information Science and Applications (ICISA) (2011)
46. McClure, G., Chang, M., Lin, F.: MAS Controlled NPCs in 3D Virtual Learning Environment, The International Workshop on Smart Learning Environments at the 9th International Conference on Signal-Image Technology and Internet-Based Systems, Japan, Kyoto, 2013-12-02, 1026 – 1033, DOI: 10.1109/SITIS.2013.166 (2013)
47. Walsh, T.: An Empirical Study of the Manipulability of Single Transferable Voting, Proceedings of the 2010 conference on ECAI, 257-262 (2010)
48. Leung, S., Virwaney, S., Lin, F., Armstrong, A J, Dubbelboer, A.: TSI-enhanced Pedagogical Agents to Engage Learners in Virtual Worlds", International Journal of Distance Education Technologies, 11(1), 1-13 (2013)

Is this model for real? Simulating data to reveal the proximity of a model to reality

Rinat B. Rosenberg-Kima[1], Zachary A. Pardos[2]

[1] Tel-Aviv University
rinat.rosenberg.kima@gmail.com
[2] University of California, Berkeley
pardos@berkeley.edu

Abstract. Simulated data plays a central role in Educational Data Mining and in particular in Bayesian Knowledge Tracing (BKT) research. The initial motivation for this paper was to try to answer the question: given two datasets could you tell which of them is real and which of them is simulated? The ability to answer this question may provide an additional indication of the goodness of the model, thus, if it is easy to discern simulated data from real data that could be an indication that the model does not provide an authentic representation of reality, whereas if it is hard to set the real and simulated data apart that might be an indication that the model is indeed authentic. In this paper we will describe analyses of 42 GLOP datasets that were performed in an attempt to address this question. Possible simulated data based metrics as well as additional findings that emerged during this exploration will be discussed.

Keywords: Bayesian Knowledge Tracing (BKT), simulated data, parameters space.

1 Introduction

Simulated data has been increasingly playing a central role in Educational Data Mining [1] and Bayesian Knowledge Tracing (BKT) research [1, 4]. For example, simulated data was used to explore the convergence properties of BKT models [5], an important area of investigation given the identifiability issues of the model [3]. In this paper, we would like to approach simulated data from a slightly different angle. In particular, we claim that the question "*given two datasets could you tell which of them is real and which of them is simulated?"* is interesting as it can be used to evaluate the goodness of a model and may potentially serve as an alternative metric to RMSE, AUC, and others. In a previous work [6] we started approaching this problem by contrasting two real datasets with their corresponding two simulated datasets with Knowledge Tracing as the model. We found a surprising close to identity between the real and simulated datasets. In this paper we would like to continue this investigation by expanding the previous analysis to the full set of 42 Groups of Learning Opportunities (GLOPs) real datasets generated from the ASSISTments platform [7].

Knowledge Tracing (KT) models are widely used by cognitive tutors to estimate the latent skills of students [8]. Knowledge tracing is a Bayesian model, which assumes that each skill has 4 parameters: two knowledge parameters include initial (prior knowledge) and learn rate, and two performance parameters include guess and slip. KT in its simplest form assumes a single point estimate for prior knowledge and learn rate for all students, and similarly identical guess and slip rates for all students. Simulated data has been used to estimate the parameter space and in particular to answer questions that relate to the goal of maximizing the log likelihood (LL) of the model given parameters and data, and improving prediction power [7, 8, 9].

In this paper we would like to use the KT model as a framework for comparing the characteristics of simulated data to real data, and in particular to see whether it is possible to distinguish between the real and simulated datasets.

2 Data Sets

To compare simulated data to real data we started with 42 Groups of Learning Opportunities (GLOPs) real datasets generated from the ASSISTments platform[1] from a previous BKT study [7]. The datasets consisted of problem sets with 4 to 13 questions in linear order where all students answer all questions. The number of students per GLOP varied from 105 to 777. Next, we generated two synthetic, simulated datasets for each of the real datasets using the best fitting parameters that were found for each respective real datasets as the generating parameters. The two simulated datasets for each real one had the exact same number of questions, and same number of students.

3 Methodology

The approach we took to finding the best fitting parameters was to calculate LL with a grid search of all the parameters (prior, learn, guess, and slip). We hypothesized that the LL gradient pattern of the simulated data and real data will be different across the space. For each of the datasets we conducted a grid search with intervals of .04 that generated 25 intervals for each parameter and 390,625 total combinations of prior, learn, guess, and slip. For each one of the combinations LL was calculated and placed in a four dimensional matrix. We used fastBKT [12] to calculate the best fitting parameters of the real datasets and to generate simulated data. Additional code in Matlab and R was generated to calculate LL and RMSE and to put all the pieces together[2].

[1] Data can be obtained here: http://people.csail.mit.edu/zp/

[2] Matlab and R code will be available here: www.rinatrosenbergkima.com/AIED2015/

4 What are the Characteristics of the Real Datasets Parameters Space?

Before we explored the relationships between the real and sim datasets, we were interested to explore the BKT parameter profiles of the real datasets. We calculated the LL with a grid search of 0.04 granularity across the four parameters resulting in a maximum LL for each dataset and corresponding best prior, learn, guess, and slip. Figure 1 present the best parameters for each datasets, taking different views of the parameters space. The first observation to be made is that the best guess and slip parameters fell into two distinct areas (see figure 1, guess x slip).

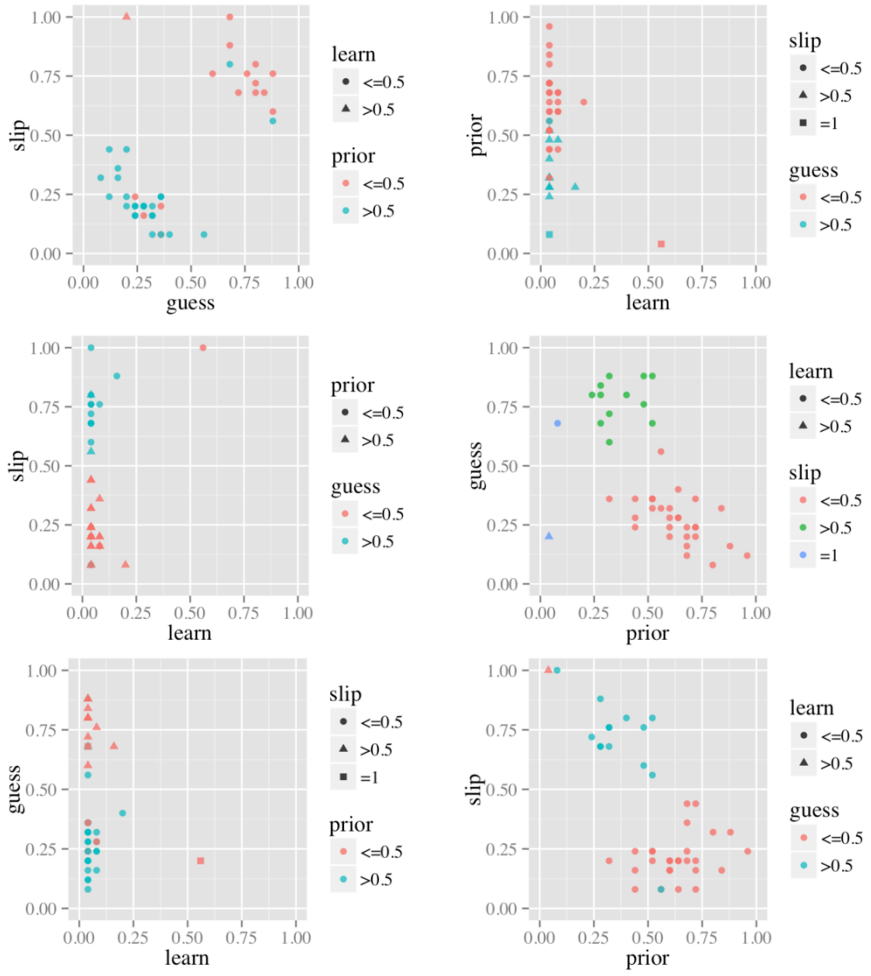

Figure 1. Best parameters across the 42 GLOP real datasets.

Much attention has been given to this LL space, which revealed the apparent co-linearity of BKT with two primary areas of convergence, the upper right area being a false, or "implausible" converging area as defined by [3]. What is interesting in this figure is that real data also converged to these two distinct areas. To further investigate this point, we looked for the relationships between the best parameters and the number of students in the dataset (see figure 2). We hypothesized that perhaps the upper right points were drawn from datasets with small number of students; nevertheless, as figure 2 reveals, that was not the case. Another interesting observation is that while in the upper right area (figure 1, guess x slip) most of the prior best values were smaller that 0.5, in the lower left area most of the prior best values were bigger than 0.5, thus revealing interrelationships between slip, guess, and prior that can be seen in the other views. Another observation is that while prior is widely distributed between 0 and 1, most of best learn values are smaller than 0.12.

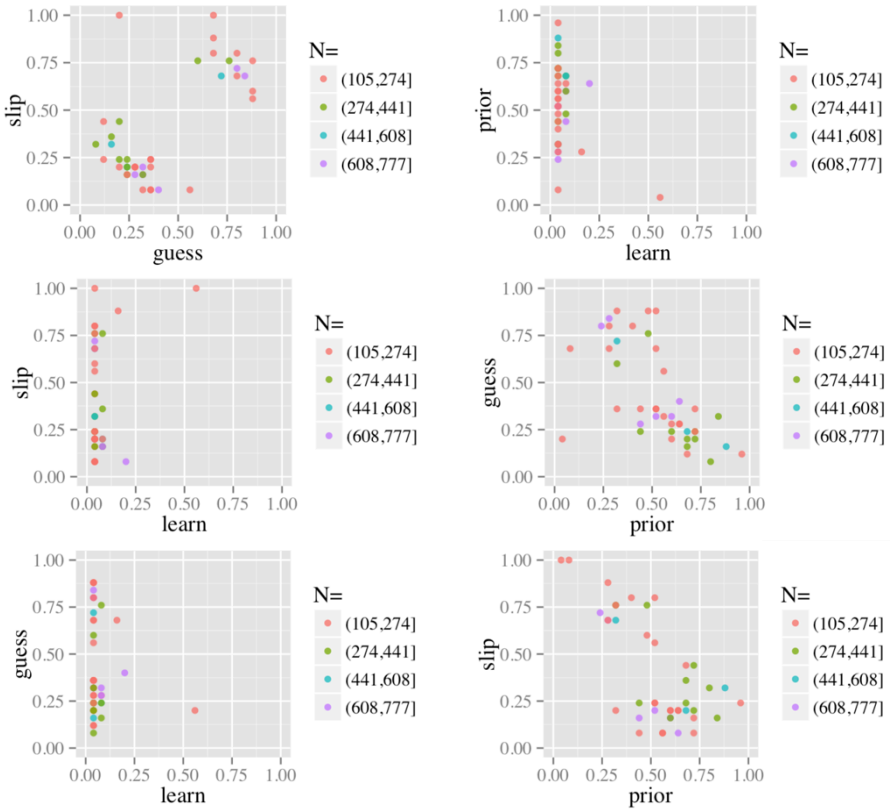

Figure 2. Best parameters across the 42 GLOP real datasets by number of students.

5 Does the LL of Sim vs. Real Datasets Look Different?

Our initial thinking was that as we are using a simple BKT model, it is not authentically reflecting reality in all its detail and therefore we will observe different patterns of LL across the parameters space between the real data and the simulated data. The LL space of simulated data in [5] was quite striking in its smooth surface but the appearance of real data was left as an open research question. First, we examined the best parameters spread across the 42 first set of simulated data we have generated. As can be seen in figure 3, the results are very similar (although not identical) to the results we received with the real data (see figure 1). This is not surprising, after all, the values of learn, prior, guess, and slip were inputs to the function generating the simulated data.

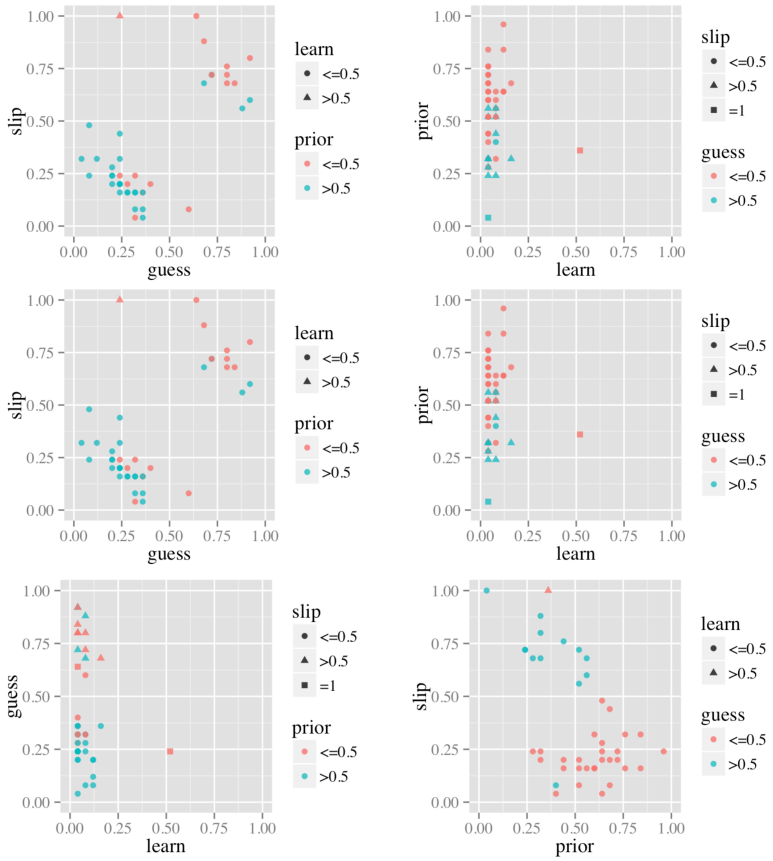

Figure 3. Best parameters across 42 GLOP simulated datasets.

In order to see if the differences between real and sim were more than just the difference between samples from the same distribution, we generated *two* simulated versions of each real dataset (sim1 and sim2) using the exact same number

of questions, number of students, generated with the best fitting parameters from the real dataset. We then visualized 2D LL heatmaps looking at two parameter plots at a time where the other two parameters were fixed to the best fitting values. For example, when visualizing LL heatmaps for the combination of guess and slip, we fixed learn and prior to be the best learn and the best prior from the real data grid search. To our surprise, when we plotted heatmaps of the LL matrices of the real data and the simulated data (the first column in figure 4 represents the real datasets, the second column represents the corresponding sim1, and the third column the corresponding sim2) we received what appears to be extremely similar heatmaps. Figure 4 and 5 displays a sample of 4 datasets, for each one displaying the real dataset heatmap and the corresponding two simulated datasets heatmaps.

The guess vs. slip heatmaps (see figure 4) prompted interesting observations. As mentioned above, the best guess and slip parameters across datasets fell into two areas (upper right and lower left). Interestingly, these two areas were also noticeable in the individual heatmaps. While in some of the datasets they were less clear (e.g., G5.198 in figure 4), most of the datasets appear to include two distinct global maxima areas. In some of the datasets the global maxima converged to the lower left expected area, as did the corresponding simulated datasets (e.g., G4.260 in figure 4), in other datasets the global maxima converged to the upper right "implausible" area, as did the corresponding simulated datasets (e.g., G6.208 in figure 4). Yet in some cases, one or more of the simulated dataset converged to a different area than that of the real dataset (e.g., G4.205 in figure 4). The fact that so many of the real datasets converged to the "implausible" area is surprising and may be due to small number of students or to other limitations of the model.

The learn vs. prior heatmaps were also extremely similar within datasets and exhibited a similar pattern also across datasets (see figure 5), although not all datasets had the exact pattern (e.g., G5.198 is quite different than the other 3 datasets in figure 5). While best learn values were low across the datasets, the values of best prior varied. As with guess vs. slip, in some cases the two simulated datasets were different (e.g., G4.205 had different best parameters also with respect to prior). Similar patterns of similarities within datasets and similarities with some clusters across datasets were also noticeable in the rest of the parameters space (learn vs. guess, learn vs. slip, prior vs. guess, prior vs. slip not displayed here due to space considerations).

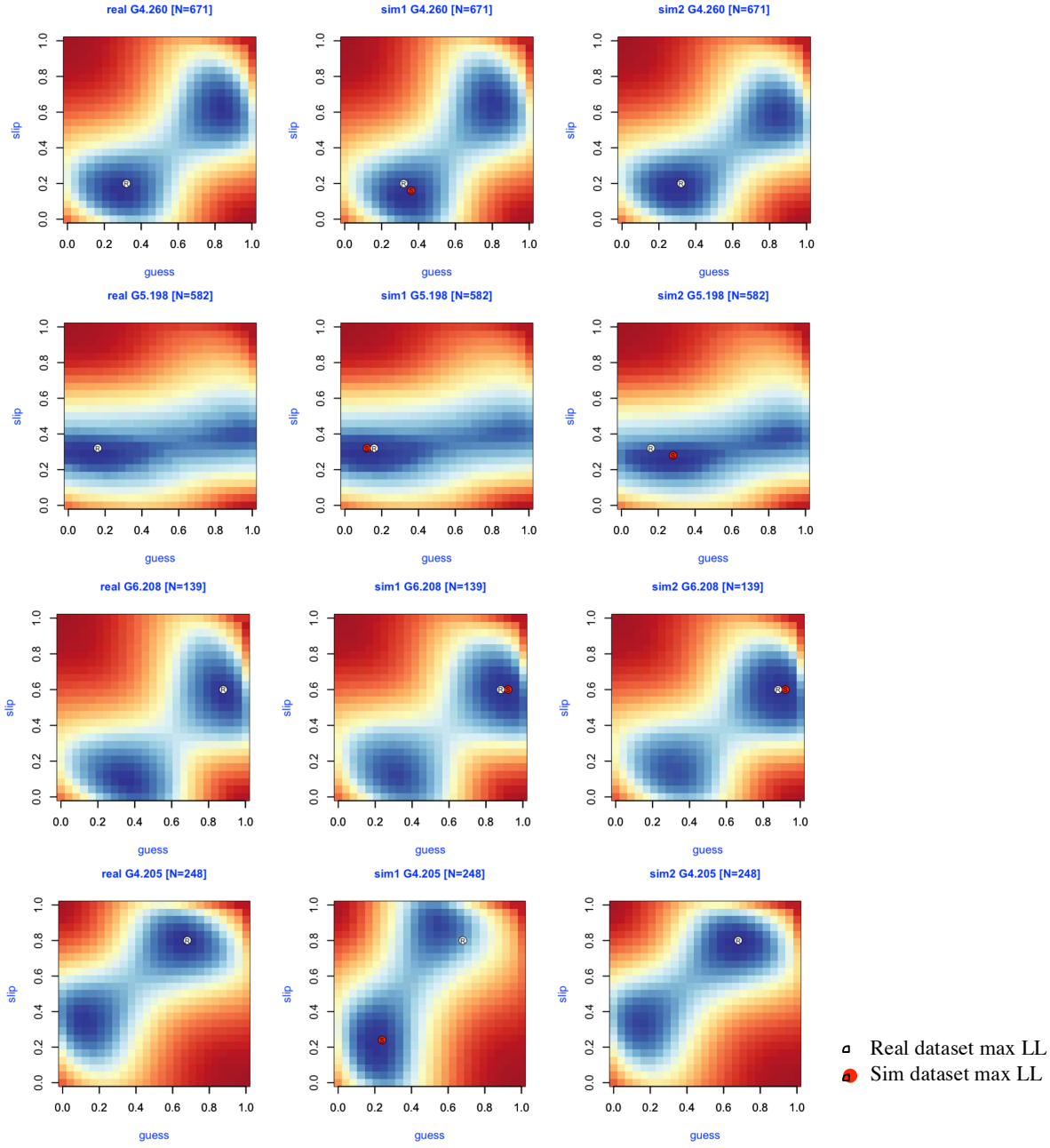

Figure 4. Heatmaps of (guess vs. slip) LL of 4 sample real GLOP datasets and the corresponding two simulated datasets that were generated with the best fitting parameters of the corresponding real dataset.

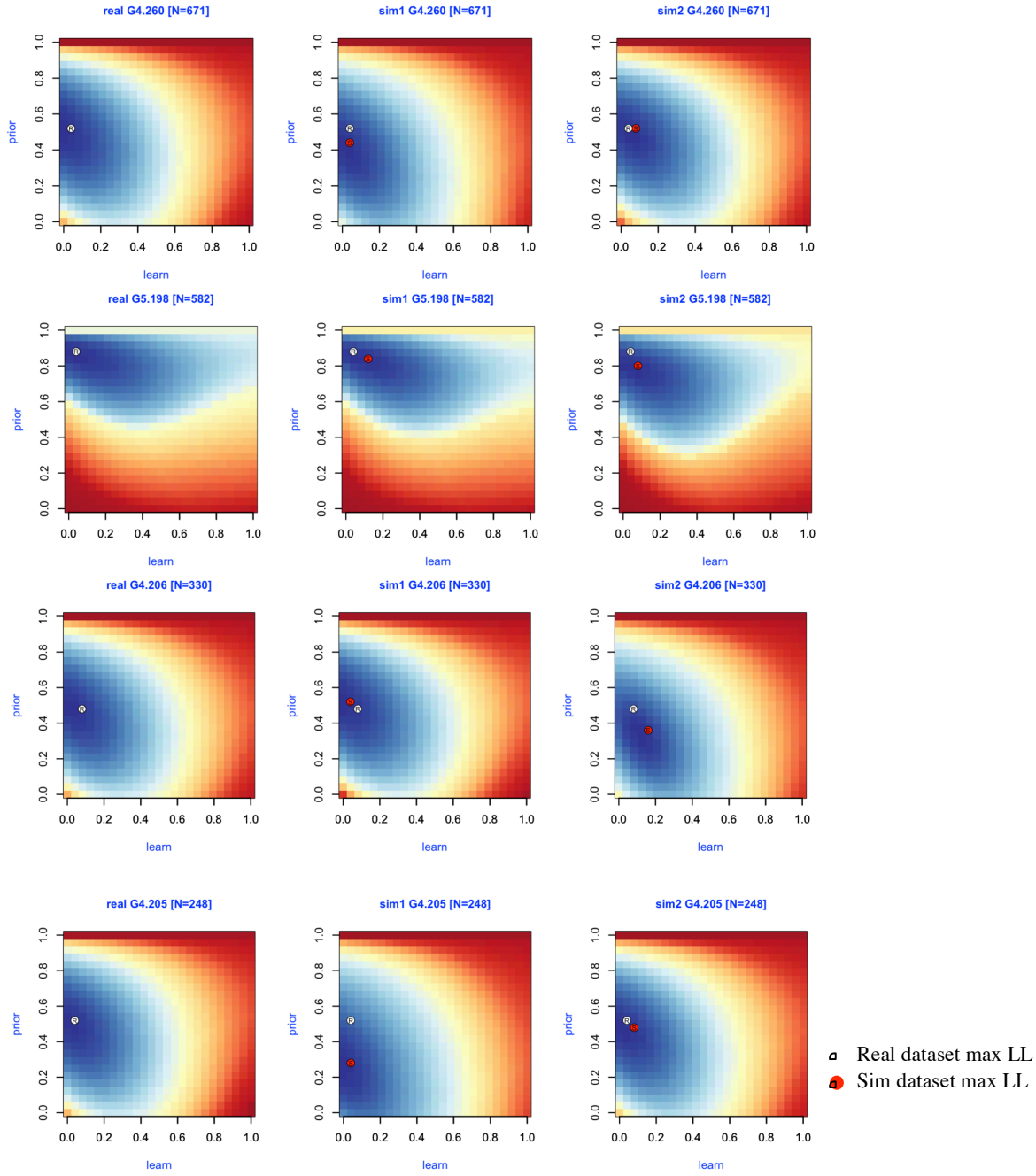

Figure 5. Heatmaps of (learn x prior) LL of 4 sample real GLOP datasets and the corresponding two simulated datasets that were generated with the best fitting parameters of the corresponding real dataset.

6 Exploring Possible Metrics Using the Real and Sim Datasets

In natural science domains, simulated data is often used as a mean to evaluate its underlying model. For example, simulated data is generated from a hypothesized model of the phenomena and if the simulated data appears to be similar to the real data observed in nature, it serves as evidence for the accuracy of the model. Then, if the underlying is validated, simulated data is used to make predictions (e.g., in the recent earthquake in Nepal a simulation was used to estimate the number of victims). Can this approach be used in education as well? What would be an indication of similarity between real and simulated data?

Figure 5 displays two preliminary approaches for comparing the level of similarity between the simulated and real data. First, the Euclidean distance between the real dataset parameters and the simulated data parameters was compared to the Euclidean distance between the two simulated datasets parameters. The idea is that if the difference between the two simulated datasets is smaller than the difference between the real and the simulated dataset this may be an indication that the model can be improved upon. Thus, points on the right side of the red diagonal indicate good fit of the model to the dataset. Interestingly, most of the points were on the diagonal and a few to the left of it. Likewise the max LL distance between the real and simulated datasets was compared to the max LL distance of the two simulated datasets. Interestingly, datasets with larger number of students did not result in higher similarity between the real and simulated dataset. Also, here we *did* find distribution of the points to the left and to the right of the diagonal.

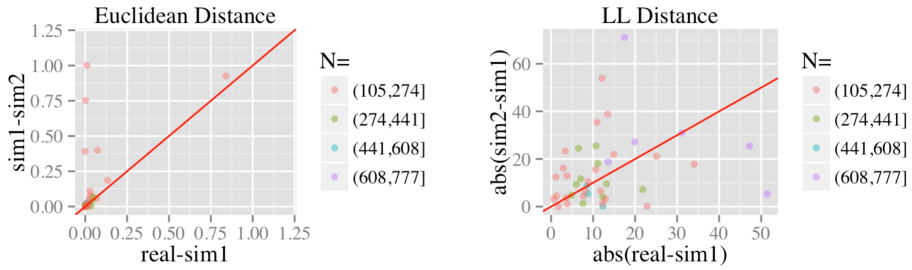

Figure 5. Using Euclidean distance and LL distance as means to evaluate the model.

7 Contribution

The initial motivation of this paper was to find whether it is possible to discern a real dataset from a simulated dataset. If for a given model it is possible to tell apart a simulated data from a real dataset then the authenticity of the model can be questioned. This line of thinking is in particular typical of simulation use in Science contexts, where different models are used to generate simulated data, and then if a simulated data has a good fit to the real phenomena at hand, then it may be possible to claim that the model provides an authentic explanation of the system [13]. We believe

that finding such a metric can serve as the foundation for evaluating the goodness of a model by comparing a simulated data from this model to real data and that such a metric could provide much needed substance in interpretation beyond that which is afforded by current RMSE and AUC measures. This can afford validation of the simulated data, which can than be used to make predictions on learning scenarios; decreasing the need to test them in reality, and at minimum, serving as an initial filter to different learning strategies.

References

[1] R. S. Baker and K. Yacef, "The state of educational data mining in 2009: A review and future visions," *J. Educ. Data Min.*, vol. 1, no. 1, pp. 3–17, 2009.

[2] M. C. Desmarais and I. Pelczer, "On the Faithfulness of Simulated Student Performance Data.," in *EDM*, 2010, pp. 21–30.

[3] J. E. Beck and K. Chang, "Identifiability: A fundamental problem of student modeling," in *User Modeling 2007*, Springer, 2007, pp. 137–146.

[4] Z. A. Pardos and M. V. Yudelson, "Towards Moment of Learning Accuracy," in *AIED 2013 Workshops Proceedings Volume 4*, 2013, p. 3.

[5] Z. A. Pardos and N. T. Heffernan, "Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence of the Expectation Maximization algorithm.," in *EDM*, 2010, pp. 161–170.

[6] R. B. Rosenberg-Kima and Z. Pardos, "Is this Data for Real?," in *Twenty Years of Knowledge Tracing Workshop*, London, UK, pp. 141–145.

[7] Z. A. Pardos and N. T. Heffernan, "Modeling individualization in a bayesian networks implementation of knowledge tracing," in *User Modeling, Adaptation, and Personalization*, Springer, 2010, pp. 255–266.

[8] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Model. User-Adapt. Interact.*, vol. 4, no. 4, pp. 253–278, 1994.

[9] S. Ritter, T. K. Harris, T. Nixon, D. Dickison, R. C. Murray, and B. Towle, "Reducing the Knowledge Tracing Space.," *Int. Work. Group Educ. Data Min.*, 2009.

[10] R. S. d Baker, A. T. Corbett, S. M. Gowda, A. Z. Wagner, B. A. MacLaren, L. R. Kauffman, A. P. Mitchell, and S. Giguere, "Contextual slip and prediction of student performance after use of an intelligent tutor," in *User Modeling, Adaptation, and Personalization*, Springer, 2010, pp. 52–63.

[11] R. S. Baker, A. T. Corbett, and V. Aleven, "More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing," in *Intelligent Tutoring Systems*, 2008, pp. 406–415.

[12] Z. A. Pardos and M. J. Johnson, "Scaling Cognitive Modeling to Massive Open Environments (in preparation)," *TOCHI Spec. Issue Learn. Scale*.

[13] U. Wilensky, "GasLab—an Extensible Modeling Toolkit for Connecting Micro- and Macro-properties of Gases," in *Modeling and simulation in science and mathematics education*, Springer, 1999, pp. 151–178.

Preface

This workshop, a follow-up to the successful first Simulated Learners workshop held at AIED 2013, is intended to bring together researchers who are interested in simulated learners, whatever their role in the design, development, deployment, or evaluation of learning systems. Its novel aspect is that it isn't simply a workshop about pedagogical agents, but instead focuses on the other roles for simulated learners in helping system designers, teachers, instructional designers, etc.

As learning environments become increasingly complex and are used by growing numbers of learners (sometimes in the hundreds of thousands) and apply to a larger range of domains, the need for simulated learners (and simulation more generally) is compelling, not only to enhance these environments with artificial agents, but also to explore issues using simulation that would be otherwise be too expensive, too time consuming, or even impossible using human subjects. While some may feel that MOOCs provide ample data for experimental purposes, it is hard to test specific hypotheses about particular technological features with data gathered for another purpose. Moreover, privacy concerns, ethics approval, attrition rates and platform constraints can all be barriers to this approach. Finally, with thousands of learners at stake, it is wise to test a learning environment as thoroughly as possible before deployment.

Since this is a follow-up to the 2013 workshop, we build on some of the ideas that emerged there (see proceedings at: http://goo.gl/12ODji).

The workshop explores these and other issues with the goal of further understanding the roles that simulated learners may play in advanced learning technology research and development, and in deployed learning systems.

John Champaign and Gord McCalla
Workshop Co-Chairs

Second Workshop on Simulated Learners

held in conjunction with

Seventeenth International Conference on Artificial Intelligence in Education (AIED 2015)

Friday, June 26, 2015
Madrid, Spain

Workshop Co-Chairs:

John Champaign[1] and Gord McCalla[2]

[1] *University of Illinois at Springfield, Springfield, IL, 62703, USA*
[2] *University of Saskatchewan, Saskatoon, S7N 5C9, Canada*

https://sites.google.com/site/simulatedlearners/

Table of Contents

Preface

This workshop, a follow-up to the successful first Simulated Learners workshop held at AIED 2013, is intended to bring together researchers who are interested in simulated learners, whatever their role in the design, development, deployment, or evaluation of learning systems. Its novel aspect is that it isn't simply a workshop about pedagogical agents, but instead focuses on the other roles for simulated learners in helping system designers, teachers, instructional designers, etc.

As learning environments become increasingly complex and are used by growing numbers of learners (sometimes in the hundreds of thousands) and apply to a larger range of domains, the need for simulated learners (and simulation more generally) is compelling, not only to enhance these environments with artificial agents, but also to explore issues using simulation that would be otherwise be too expensive, too time consuming, or even impossible using human subjects. While some may feel that MOOCs provide ample data for experimental purposes, it is hard to test specific hypotheses about particular technological features with data gathered for another purpose. Moreover, privacy concerns, ethics approval, attrition rates and platform constraints can all be barriers to this approach. Finally, with thousands of learners at stake, it is wise to test a learning environment as thoroughly as possible before deployment.

Since this is a follow-up to the 2013 workshop, we build on some of the ideas that emerged there (see proceedings at: http://goo.gl/12ODji).

The workshop explores these and other issues with the goal of further understanding the roles that simulated learners may play in advanced learning technology research and development, and in deployed learning systems.

John Champaign and Gord McCalla
Workshop Co-Chairs

An Approach to Developing Instructional Planners for Dynamic Open-Ended Learning Environments

Stephanie Frost and Gord McCalla

ARIES Lab, Department of Computer Science, University of Saskatchewan, Canada
stephanie.frost@usask.ca, mccalla@cs.usask.ca

Abstract. *Instructional planning* (IP) technology has begun to reach large online environments. However, many approaches rely on having centralized metadata structures about the learning objects (LOs). For *dynamic open-ended* learning environments (DOELEs), an approach is needed that does not rely on centralized structures such as prerequisite graphs that would need to be continually rewired as the LOs change. A promising approach is collaborative filtering based on learning sequences (CFLS) using the ecological approach (EA) architecture. We developed a CFLS planner that compares a given learner's most recent path of LOs (of length b) to other learners to create a neighbourhood of similar learners. The future paths (of length f) of these neighbours are checked and the most successful path ahead is recommended to the target learner, who then follows that path for a certain length (called s). We were interested in how well a CFLS planner, with access only to pure behavioural information, compared to a traditional instructional planner that used explicit metadata about LO prerequisites. We explored this question through simulation. The results showed that the CFLS planner in many cases exceeded the performance of the simple prerequisite planner (SPP) in leading to better learning outcomes for the simulated learners. This suggests that IP can still be useful in DOELEs that often won't have explicit metadata about learners or LOs.

Keywords: instructional planning, collaborative filtering, dynamic open-ended learning environments, simulated learning environments, simulated learners, ecological approach

1 Introduction

Online courses need to be able to personalize their interactions with their many learners not only to help each learner overcome particular impasses but also to provide a path through the learning objects (LOs) that is appropriate to that particular individual. This is the role of *instructional planning* (IP), one of the core AIED sub-disciplines. IP is particularly needed in open-ended learning environments (OELEs), where learners choose their own goals, because it has been shown that sometimes learners require an outside push to move forward

[11]. An added challenge is what we call a *dynamic open-ended* learning environment (DOELE), where both the learners and LOs are constantly changing. Some learners might leave before finishing the course, while others may join long after other learners have already begun. New material (LOs) may need to be added in response to changes in the course or the material, or to learner demand. Sometimes new material will be provided by the course developers, but the big potential is for this to be crowd sourced to anybody, including learners themselves. Other material may fade away over time.

Note that a DOELE is similar to, but not the same as, a "traditional" open-eded learning environment [8, 11]. A traditional open-ended environment also gives students choice, but mostly in the problems they solve and how they solve them, with the course itself fixed in its content, order and goals. In a DOELE everything is open-ended and dynamic, including even what is to be learned, how deeply, when it needs to be learned, and in what order.

An impediment to IP in a DOELE is that there is no centralized representation of knowledge about the content or the learners. Work has been done to make IP possible in online environments, such as [7], where authors showed that by extending the LO metadata, instructional plans could be improved to adapt based on individual learning styles as well as a resource's scheduling availability. But for IP to work in DOELEs, an approach to IP is needed where centralized course structures would not need to be continually revamped (by instructional designers, say) as learners and LOs change.

We wish to explore how IP can be done in a DOELE. We model a DOELE in the ecological approach (EA) architecture [14]. In the EA there is no overall course design. Instead, courses are conceived as collections of learning objects each of which captures usage data as learners interact with it. Over time this usage data accumulates and can be used for many pedagogical purposes, including IP [2]. Drawing inspiration from work like [1, 5], we propose a new IP algorithm based on collaborative filtering of learning sequences (CFLS). For a given learner our planner finds other learners who have traversed a similar sequence of learning objects with similar outcomes (i.e. similar paths). Then it suggests paths to the learner that were successful for these similar learners (peers) going forward.

To evaluate IP techniques in such an environment, one could implement a real course with thousands of learners using the EA to capture learner interactions with the various LOs in the course. However, after doing this it would take several years for enough learners to build up enough interactions with each LO to provide useful data to be used by an instructional planner. Also, in a course with thousands of learners, there is risk of causing confusion or inconvenience to a vast multitude if there are problems while the planner is under development. Finally, there are unanswered design questions such as the criteria to use for identifying an appropriate peer, how many LOs should be recommended for a learner before re-planning occurs, and appropriate values for many other parameters that would be used by the planner. In order to overcome these challenges and gain insight into these questions immediately, we have thus turned to simulation.

2 Simulation Environment

Before describing the CFLS planner and experiment in detail, we describe the simulation environment. The simulation is low-fidelity, using very simple abstractions of learners and LOs, as in our earlier work [6]. Each of the 40 LOs has a difficulty level and possible prerequisite relationships with other LOs. Each simulated learner has an attribute, *aptitude-of-learner*, a number between (0,1) representing a learner's basic capability for the subject and allows learners to be divided into groups: low ($\leq .3$), medium ($.4 - .7$) and high aptitude ($\geq .8$).

A number called *P[learned]* is used to represent the learning that occurred when a learner visits a LO, or the probability that the learner learned the LO. P[learned] is generated by an *evaluation function*, a weighted sum: 20% of the learner's score on a LO is attributed to *aptitude-of-learner*, 50% attributed to whether the learner has mastered all of the prerequisite LOs, 20% attributed to whether the learner had seen that LO previously, and 10% attributed to the difficulty level of the LO. We feel this roughly captures the actual influences on how likely it is that real learners would master a learning object.

The simulated learners move through the course by interacting with the LOs, one after another. After each LO is encountered by a simulated learner, the above evaluation function is applied to determine the learner's performance on the LO, the P[learned] for that learner on that LO. In the EA architecture, everything that is known about a learner at the time of an interaction with a LO (in this case, including P[learned]) is captured and associated with that LO. The order of the LOs visited can be set to random, or it can be determined by a planner such as the CFLS planner. To allow for the comparison of different planning approaches without advantaging one approach, each simulated learner halts after its 140th LO regardless of the type of planner being used.

3 Experiment

By default, the simulation starts with an empty history - no simulated learners have yet viewed any LOs. However, because the CFLS planner relies on having previous interaction data, it is necessary to initialize the environment. Thus, a simple prerequisite planner (SPP) was used to initialize the case base with a population of simulated learners. The SPP is privy to the underlying prerequisite structure and simply delivers LOs to learners in prerequisite order. As Table 1 shows, the SPP works much better than a random planner. The data from the 65 simulated learners who used the SPP thus was used to initialize the environment before the CFLS planner took over. This interaction data generated by the SPP also provides a baseline for comparison with the CFLS planner. Our simulation experiment was aimed at seeing if, with appropriate choices of b and f (described below) the CFLS planner could work as well or better than the SPP.

We emphasize that the CFLS planner has no knowledge about the underlying prerequisite structure of the learning objects. This is critical for CFLS planning to work in a DOELE. However, there are two places where clarification

Table 1. Baseline results for each group of simulated learners (high, medium and low aptitude) when visiting LOs randomly and following a simple prerequisite planner.

Planning Type / Aptitude	low	medium	high
Random	N=21	N=26	N=18
Average Score on Final Exam (P[learned])	0.107	0.160	0.235
Simple Prerequisite Planner (SPP)	N=21	N=26	N=18
Average Score on Final Exam (P[learned])	0.619	0.639	0.714

is required. First, while the SPP is running, the evaluation function will be used by the simulation to calculate P[learned] values for each LO visited. This usage data will contain implicit evidence of the prerequisite relationships. So, at a later time when the CFLS planner is given access to the same usage data, the CFLS planner could implicitly discover prerequisite relationships from the interaction data. Second, during the CFLS planner execution, the underlying prerequisite structure is still being consulted by the evaluation function. However, the CFLS planner knows nothing about such prerequisites, only the P[learned] outcome provided by the evaluation function. When simulated learners are replaced with real learners, the evaluation function would disappear and be replaced with a real world alternative, such as quizzes or other evidence to provide a value for P[learned]. Similarly, the CFLS planner does not require knowledge of the difficulty level of each LO, nor does it require knowledge of the aptitude of each learner; these are just stand-in values for real world attributes used by the simulation and would disappear when the planner is applied in a real world setting.

Different studies can use simulated student data in varying ways. In some cases, low fidelity modelling is not adequate. For example, in [4] it was found that the low fidelity method of generating simulated student data failed to adequately capture the characteristics of real data. As a result, when the simulated student dataset was used for training the cognitive diagnosis model, its predictive power was worse than when the cognitive diagnosis model was trained with a simulated student dataset that had been generated with a higher fidelity method. In our study, using a low fidelity model is still informative. We are less concerned with the exactness of P[learned] and are more interested in observing possible relative changes of P[learned] for certain groups of students, as different variations of the planner are tried on identical populations of simulated students.

The CFLS planner works as follows. For a given target learner the CFLS planner looks backward at the b most recent learning objects traversed. Then, it finds other learners who have traversed the same b learning objects with similar P[learned] values. These b LOs can be in any order, a simplification necessary to create a critical mass of similar learners. These are learners in the target learner's "neighbourhood". The planner then looks forward at the f next LOs traversed by each neighbour and picks the highest value path, where value is defined as the average P[learned] achieved on those f LOs ahead. This path is then recommended to the learner, who must follow it for at least s (for "sticky") LOs before replanning occurs. Of course, s is always less than f. In our research

we explored various values of b and f to find which leads to the best results (we set $f = s$ for this experiment). "Best results" can be defined many ways, but we focused on two measurements that were taken for each learner at the end of each simulation: the percentage of LOs mastered, and the score on a final exam. A LO is considered to be mastered when a score of P[learned] = 0.6 or greater is achieved. The score on the final exam is taken as the average P[learned] on the LOs that are the leafs of the prerequisite graph (interpreted as the ultimate target concept, which in the real world might well be final exams).

There is still a cold start problem even after the simulation has been initialized with the interaction data from the SPP. This is because the simulated learners who are to follow the CFLS planner have not yet viewed any LOs themselves as they begin the course, so there is no history to match the b LOs to create the plan. In this situation, the CFLS planner matches the learner with another arbitrary learner (from the interaction data from the SPP), and recommends whatever initial path that the other learner took when they first arrived in the course. While another solution to the cold start problem could be to start the new learner with the SPP, we did this to avoid any reliance whatsoever on knowing the underlying prerequisite structure.

The most computationally expensive part of the CFLS planner is finding the learners in the neighbourhood, which is at worst linear on the number of learners and linear on the amount of LO interaction history created by each learner. Each learner's LO interaction history must be searched to check for a match with b, with most learners being removed from the list during this process. The forward searching of f is then executed using only the small resulting dataset.

4 Results

We ran the CFLS planner 25 different times with all pairings of the values of b and s ranging from 1 to 5, using a population of 65 simulated learners. This population had the same distribution of aptitudes as the population used to generate the baseline interaction data described above. The heat maps in Figs. 1 and 2 show the measurements for each of the 25 simulations, for each aptitude group, with the highest relative scores coloured red, mid-range scores coloured white, and the lowest scores coloured blue. In general, simulated learners achieved higher scores when following the CFLS planner than when given LOs randomly. The CFLS planner even exceeded the SPP in many cases.

A success triangle is visible in the lower left of each aptitude group. The success triangles can be interpreted to mean that if a path is going to be recommended, never send the learner any further ahead (s) than you have matched them in the past (b). For example if a learner's neighbourhood was created using their $b = 2$ most recent LOs, then never make the learner follow in a neighbour's steps further than $s = 2$ LOs. One reason for the eventual drop at high values of b is that no neighbour could be found and a random match is used instead. However, the abrupt drop at $b > s$ was unexpected. To be sure the pattern was real, an extended series of simulations was run. We ran $b = 6$ and $s = 5$ to see

if there would be a drastic drop in performance, and indeed this was the case. We also ran another row varying b with a fixed $s = 6$, and again found a drop at $b = 7$.

LOW				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
100	21.9	32.5	31	37.7
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
89.6	86	36.9	36.9	40.4
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
72.1	68.6	62	43.21	40.1
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
77.3	74.4	72.1	66.1	49.3
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
68.1	70.7	67.5	67.4	63.5

MEDIUM				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
100	50	45.8	42.2	44.1
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
100	100	42.9	40.3	38
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
100	99.4	98.6	42.4	43
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
100	99.3	99.5	99.4	50.8
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
100	100	100	100	100

HIGH				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
100	75	39.3	39.7	41.1
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
100	100	41.7	34.9	40
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
100	100	100	42.5	50
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
100	100	100	100	61
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
100	100	100	100	100

Fig. 1. Average % Learning Objects Mastered by aptitude group

LOW				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
0.6587	0.1036	0.1314	0.1283	0.146
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
0.5178	0.4387	0.1398	0.1248	0.1363
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
0.4051	0.266	0.2256	0.1586	0.132
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
0.4138	0.2984	0.3016	0.2755	0.176
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
0.357	0.2884	0.2859	0.2679	0.2249

MEDIUM				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
0.6894	0.1851	0.2105	0.2099	0.2425
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
0.7004	0.698	0.2058	0.22	0.1972
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
0.6942	0.6761	0.6715	0.1944	0.2152
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
0.6931	0.6867	0.6874	0.6856	0.2292
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
0.6912	0.6884	0.6924	0.6965	0.6899

HIGH				
b=1 s=1	b=2 s=1	b=3 s=1	b=4 s=1	b=5 s=1
0.7641	0.2514	0.2805	0.2866	0.2702
b=1 s=2	b=2 s=2	b=3 s=2	b=4 s=2	b=5 s=2
0.77	0.7694	0.2673	0.2738	0.2748
b=1 s=3	b=2 s=3	b=3 s=3	b=4 s=3	b=5 s=3
0.7653	0.7638	0.7727	0.3019	0.3097
b=1 s=4	b=2 s=4	b=3 s=4	b=4 s=4	b=5 s=4
0.768	0.7697	0.7633	0.7697	0.3431
b=1 s=5	b=2 s=5	b=3 s=5	b=4 s=5	b=5 s=5
0.7601	0.7612	0.7591	0.7644	0.7636

Fig. 2. Average Score on Final Exam (P[learned]) by aptitude group

A hot spot of successful combinations of b and s appeared for each aptitude group. For low aptitude learners, it was best to only match on the $b = 1$ most recent learning objects, and to follow the selected neighbour for only $s = 1$ LOs ahead before replanning. This combination of b and s is the only time when the CFLS planner outperformed the SPP for the low aptitude group. However, for the medium and high aptitude groups, the CFLS planner outperformed the SPP in all cases within the success triangle. Looking at final exam scores (Fig. 2), medium aptitude learners responded well to being matched with neighbours using $b = 1$ or 2 and sticking with the chosen neighbour for the same distance ahead. The high aptitude group responded very well to using neighbourhoods created with $b = 3$ and recommending paths of $s = 3$.

Within the success triangles, the rows and columns of Fig. 2 were checked to see if there existed an ideal b for a given s, and vice versa. Wherever there appeared to be a large difference, Student's t-test was used to check for statistical significance. We are able to use paired t-tests because the simulated learners have exactly the same characteristics in all the simulation runs, the only difference being the order in which LOs were interacted with. For example, learner #3 always has *aptitude-of-learner* = .4, so, there is no difference in that learner

between simulation runs. We used a two-tailed t-test because it was not certain whether one distribution was going to be higher or lower than the other.

Looking along the rows, when s is held the same, there are some cases where one value of b is better than another. For the low aptitude group, for the most part the lower the b, the better. For the medium aptitude group, there were no significant advantages to changing b. For the high aptitude group, when $s = 3$, the t-test was used to check if $b = 3$ was significantly more advantageous than using $b = 2$. The measurements for Score on the Final Exam for the high aptitude learners were compared between both simulation results, ($b = 2$ and $s = 3$) and ($b = 3$ and $s = 3$). With N=19 learners in this group, the calculated p-value was 0.009, indeed a statistically significant difference.

Looking along the columns, when b is held the same there was a case where increasing s, i.e. sticking to a longer plan ahead, was statistically advantageous. In the medium aptitude group, when $b = 1$ it was statistically better to use $s = 2$ than to use $s = 1$ with a p-value of 0.011. None of the increases of s with the same b were significant for the high aptitude group, and there were no increases for the low aptitude group.

5 Analysis and Future Work

Through simulation, we have shown that a CFLS planner can be "launched" from an environment that has been conditioned with interaction data from another planner, such as an SPP, and operate successfully using only learner usage data kept by the EA and not needing centralized metadata such as a prerequisite graph. This is one of the key requirements for DOELEs. Like biological evolution, the EA is harsh in that it observes how learners succeed or fail as various paths are tried. Successful paths for particular types of learners, regardless of whether they follow standard prerequisites, is the only criterion of success. New learners or new learning objects will find their niche - some paths will work for some learners but not for others, and this is discovered automatically through usage.

More experiments are needed to explore the many possibilities of the simulation environment. While this experiment was not a true test of a DOELE because new learners and LOs were not inserted, this can be readily explored in future work. New additions could be matched randomly a few times in order to build enough data in the EA, and then automatically incorporated into neighbourhood matches or into future plans.

Given the evaluation function that was selected, we found that planning ahead and sticking to the plan worked best for high aptitude learners and a reactive approach (planning ahead but sticking to the plan for only a short time) worked best for the low aptitude learners. Would a different pattern emerge if a different evaluation function were chosen? Would a different threshold for mastery than P[learned] > 0.6 make any difference? In future work, would it be worthwhile to break down the aptitude groups into six: very-high, high, medium-high, medium-low, low, and very-low? This may assist with more easily tuning the weights of the evaluation function, as there was not much difference in our

results between the high and medium aptitude groups. In addition, more experiments where $s < f$ are needed to answer the question of whether the drop along the edge of each success triangle was because of s or f. Also, in this work we did not look at the many different types of pedagogical interactions (ex. asking the student a question, giving a hint etc.) and focused on very abstract representations. More work is needed to explore this approach on systems later in the design process, when more detail about the content and the desired interactions with learners is known.

Future work could also investigate the usage of a differential planner, where different settings are tuned for different situations. For example, when creating a neighbourhood for a low aptitude learner, medium aptitude learners could be allowed into the neighbourhood if they have a matching b. Results could reveal situations where for example a low aptitude learner is helped by following in the steps of a medium aptitude learner. A differential planner could also dynamically choose the values of b and s for a given individual instead of using the same values for everyone at all times. For example, in a real world setting a CFLS planner may try to create a plan using a neighbourhood of $b = 3$, knowing it is optimal, but if for the specific case there is not enough data, it could change to $b = 2$ on the fly. Other aspects that could be changed are the criteria for creating the neighbourhood: rather than filtering by aptitude, another attribute could be chosen such as click behaviour or learning goals.

6 Conclusion

In this paper, we have described the need for instructional planning in DOELEs with many LOs aimed at large numbers of learners. Instructional planners such as [13] use AI planning technology that is based on states, actions and events, which are difficult to infer from an unstructured online environment. In recent years, instructional planning has been replaced by instructional design approaches such as [3]. Advanced instructional planners from the 1990s, such as PEPE and TOBIE [16] can blend different teaching strategies to appropriate situations. We have shown that instructional planning can still be done in the less rigid courses envisioned by the EA architecture and likely to be commonplace in the future, using only learner usage data kept by the EA and not needing centralized metadata about the course.

We have shown a specific planning technique, the CFLS planner, that is appropriate for DOELEs, and how to experiment in this domain. The simulation experiment revealed the number of LOs from a target learner's recent browsing history should be used for creating a neighbourhood (b), a question that has also been investigated by other researchers, such as in [18]. We have also found recommendations for settings for how far ahead to plan (s and f) for different groups of learners, and identified questions for future work. As is the case with collaborative filtering and case-based approaches, the quality of the plans created is limited to the quality of LOs within the repository and the quality

of interactions that have previously occurred between learners and sequences of LOs.

The bottom-up discovery of prerequisite relationships has been investigated by others, such as [17]. When the need for centralized metadata about a course is discarded, and when the further step is taken that different paths can be found to work better for different learners, then a shift in thinking occurs. Each individual learner could effectively have a unique ideal (implicit) prerequisite graph. Whether or not a prerequisite relationship even exists between two LOs could vary from learner to learner. The notion of prerequisite can thus be viewed not only as a function of the content relationships, but also as a function of the individual learner.

Making recommendations of sequences has also been identified as a task in the recommender systems domain [9]. An approach such as a CFLS planner is a step in the direction of building recommender systems that can use sequence information to recommend sequences. This has also been accomplished with standards approaches such as [15]. Simulation with the EA provides another method for developing and testing such approaches.

Overall, the research we have done to date and the questions it raises, shows the value of exploring these complex issues using simulation. We were able to essentially generate some 25 different experiments exploring some issues in instructional planning, in a very short time when compared to what it would have taken to explore these same issues with real learners. Others have also used simulation for developing an educational planner, such as [10] for social assessment games. To be sure our simulation model was of low fidelity, but we suspect that there are some properties of the CFLS planner that we have uncovered that apply in the real world (the lower triangles seem to be very strong and consistent patterns). And, there are some very real issues that we can explore fairly quickly going forward that might reveal other strong patterns, as discussed. We believe that it isn't always necessary to have simulations with high cognitive fidelity (as in SimStudent [12]) to find out interesting things. Low fidelity simulations such as the ones we have used in this and our earlier work [6] (and those of [2]) have a role to play in AIED. Especially as we move into the huge questions of dynamic open-ended learning environments with thousands of learners and big privacy issues, the sharp minimalist modelling possible with low fidelity simulation should allow quick and safe experimentation without putting too many real learners at risk and without taking years to gain insights.

Acknowledgements
We would like to thank the Natural Sciences and Engineering Research Council of Canada for funding some aspects of this research.

References

[1] Cazella, S., Reategui, E., and Behar, P.: Recommendation of Learning Objects Applying Collaborative Filtering and Competencies. IFIP Advances in Information and Communication Technology, 324, pp 35-43 (2010)

[2] Champaign, J.: Peer-Based Intelligent Tutoring Systems: A Corpus-Oriented Approach. Ph.D. Thesis, University of Waterloo, Waterloo, Canada (2012)
[3] Drachsler, H., Hummel, H. and Koper, R.: Using Simulations to Evaluate the Effects of Recommender Systems for Learners in Informal Learning Networks. SIRTEL Workshop (Social Information Retrieval for Technology Enhanced Learning) at the 3rd EC-TEL (European Conf. on Technology Enhanced Learning) Maastricht, The Netherlands: CEUR-WS.org, online CEUR-WS.org/Vol-382/paper2.pdf (2008)
[4] Desmarais, M., and Pelczer, I.: On the Faithfulness of Simulated Student Performance Data. In de Baker, R.S.J. et al. (Eds.), Proc. of the 3rd Int. Conf. on Educ. Data Mining, pp 21-30. Pittsburg USA (2010)
[5] Elorriaga, J. and Fernández-Castro, I.: Using Case-Based Reasoning in Instructional Planning: Towards a Hybrid Self-improving Instructional Planner. Int. Journal of Artificial Intelligence in Educ., 11(4), pp 416-449 (2000)
[6] Erickson, G., Frost, S., Bateman, S., and McCalla, G.: Using the Ecological Approach to Create Simulations of Learning Environments. In Lane, H.C. et al. (Eds), Proc. of the 16th Int. Con. on AIED, pp 411-420. Memphis USA: Springer (2013)
[7] Garrido, A. and Onaindia, E.: Assembling Learning Objects for Personalized Learning: An AI Planning Perspective. Intelligent Systems, IEEE, 28(2), pp 64-73 March/April (2013)
[8] Hannafin, M.J.: Learning in Open-Ended Environments: Assumptions, Methods and Implications. Educational Technology, 34(8), pp 48-55 (1994)
[9] Herlocker, J., Konstan, J., Terveen, L., and Riedl, J.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems (TOIS) 22(1), pp 5-53 (2004)
[10] Laberge, S., Lenihan, T., Shabani, S., and Lin, F.: Multiagent Coordination for Planning and Enacting an Assessment Game. Workshop on MultiAgent System Based Learning Environments of Int. Tutoring Systems (ITS) Honolulu, USA (2014)
[11] Land, S.: Cognitive Requirements for Learning with Open-Ended Learning Environments. Deuce. Technology Research and Development, 48(3), pp 61-78 (2000)
[12] Matsuda, N., Cohen, W. and Koedinger, K.: Teaching the Teacher: Tutoring SimStudent Leads to More Effective Cognitive Tutor Authoring. Int. Journal of Artificial Intelligence in Educ., 25(1), pp 1-34 (2014)
[13] Matsuda, N., and VanLehn, K.: Decision Theoretic Instructional Planner for Intelligent Tutoring Systems. In B. du Boulay (Ed.), Workshop Proc. on Modelling Human Teaching Tactics and Strategies, ITS 2000 pp 72-83. (2000)
[14] McCalla, G.: The Ecological Approach to the Design of e-Learning Environments: Purpose-based Capture and Use of Information about Learners. Journal of Interactive Media in Educ., http://jime.open.ac.uk/jime/article/view/2004-7-mccalla (2004)
[15] Shen, L. and Shen, R.: Learning Content Recommendation Service Based on Simple Sequencing Specification. In Liu W et al. (Eds.) Advances in Web-Based Learning - ICWL 2004 3rd Int. Conf. Web-based Learning, LNCS 3143, pp 363-370. Beijing, China:Springer (2004)
[16] Vassileva, J. and Wasson, B.: Instructional Planning Approaches: from Tutoring Towards Free Learning. Proceedings of Euro-AIED'96, Lisbon, Portugal (1996)
[17] Vuong, A., Nixon, T., and Towle, B.: A Method for Finding Prerequisites Within a Curriculum. In Pechenizkiy, M. et al. (Eds.) , Proc. of the 4th Int. Con. on Educ. Data Mining, pp 211-216. Eindhoven, the Netherlands (2011)
[18] Zhang, Y., and Cao, J.: Personalized Recommendation Based on Behavior Sequence Similarity Measures. In Cao, L. et al. (Eds.) Int. Workshop on Behaviour and Social Informatics / Behaviour and Social Informatics and Computing (BSI/BSIC 2013), Gold Coast QLD Australia / Beijing China, LNCS 8178, pp 165-177 (2013)

Exploring the Issues in Simulating a Semi-Structured Learning Environment: the SimGrad Doctoral Program Design

David Edgar K. Lelei and Gordon McCalla

ARIES Laboratory, Department of Computer Science, University of Saskatchewan
davidedgar.lelei@usask.ca and mccalla@cs.usask.ca

Abstract. The help seeking and social integration needs of learners in a semi-structured learning environment require specific support. The design and use of educational technology has the potential to meet these needs. One difficulty in the development of such support systems is in their validation because of the length of time required for adequate testing. This paper explores the use of a simulated learning environment and simulated learners as a way of studying design validation issues of such support systems. The semi-structured learning environment we are investigating is a graduate school, with a focus on the doctoral program. We present a description of the steps we have taken in developing a simulation of a doctoral program. In the process, we illustrate some of the challenges in the design and development of simulated learning environments. Lastly, the expected contributions and our research plans going forward are described.

Keywords: Simulated learners, Simulated learning environment, Agent-based simulation, Help seeking, Doctoral learners, Multi-agent system.

1 Introduction

Artificial Intelligence in Education (AIED) is one of the research fields whose focus is the use of technology to support learners of all ages and across all domains[1]. Although, one shortcoming of AIED research is the limited research attention that very dynamic and semi-structured domains, such as a graduate school, have received. There is little research that investigates how technology can be used to help connect learners (help seeker and potential help givers) in the graduate school domain. Consequently, there is a gap in our understanding of how such technology may mitigate graduate learners' attrition rates and time-to-degree. We have suggested the use of reciprocal recommender technology to assist in the identification of a suitable helper [1]. However, the nature of graduate school means that validation of any education system designed to be used in a semi-structured environment would take a long time (measured in years). This paper aims to address this challenge by exploring the use of

[1] http://iaied.org/about/

simulated learning environment and simulated learners as a potential way of validating educational technologies designed to support doctoral learners.

In this paper, we first describe the nature and the metrics used by interested stakeholders to measure the success or lack therefore of a doctoral program. Following this, we briefly discuss the uses of simulation as it relates to learning environment. We then introduce the research questions we are interested in answering using simulation. We go on to describe the architectural design of our simulation model. Further, we show how data about the 'real world' target domain is used to inform the parameters and initial conditions for the simulation model. This provides the model with a degree of fidelity. Throughout this model development process, we illustrate some of the challenges in the design and development of simulated learning environments. We conclude the paper with a discussion of the expected contributions and our research plans going forward.

2 Understanding Doctoral Program

Graduate school is a very dynamic and complex social learning environment. A doctoral program in particular is a dynamic, semi-structured, and complex learning environment. Most doctoral programs have some structure in the sense that there are three distinct stages that doctoral learners must go through: admission stage, coursework stage, and dissertation stage. While coursework stage is fairly structured, the dissertation stage is not. Further, the dissertation stage have various milestones that include: comprehensive exam, thesis proposal, research, writing, and dissertation defense. As time passes, learners move from one stage to the next and their academic and social goals change. There is need for self-directed learning and individual doctoral learners are responsible for their own learning pace and choice of what to learn especially in the dissertation stage.

The dynamic nature of the program ensures that there is constant change; there are new learners joining the program, other learners leaving the program either through graduation or deciding to drop out, and still other learners proceeding from one stage to the next. There are two key aspects that influences learners to decide whether to persist or drop out of a learning institution: academic and social integration [2], [3] which are impacted by learner's initial characteristics and experiences during their duration in the program. The various stages of the doctoral program (e.g., coursework) and learning resources can be seen as factors that directly influence the academic integration of a doctoral learner. Peers and instructors/supervisors can be viewed as supporting the social aspects of the doctoral program and hence, directly impact the social integration of doctoral learners. As time passes, doctoral learners continually interact with both the academic and social facets of the doctoral program. As a result, there is constant change in learners' commitment to their academic goal and the social sides of the learning institution

Time-to-degree, completion rates, and attrition rates are important factors influencing the perception and experience of graduate education by interested stakeholders [4], [5]. Research on doctoral attrition and time-to-completion indicates that on aver-

age, the attrition rate is between 30% and 60% [5]–[8]. Long times to completion and a high attrition rate are costly in terms of money to the funding institution and the learning institution; and in terms of time and effort to the graduate student(s) and supervisor(s) [8]. Lack of both academic and social integration (isolation) have been shown to affect graduate learners decision to persist [2], [3], [9]. Learners facing academic and social integration challenges should be enabled to engage in a community of peers to foster interaction and hence, encourage peer help and personalized collaboration [10]. Understanding the nature of learner-institution interactions that foster doctoral learners' persistence to degree is important to both the learning institution and its learners. We use simulation to achieve this feat.

Simulation is an established third way of exploring research questions in addition to qualitative and quantitative methods [11], [12]. VanLehn [13] has identified three main uses of simulation in learning environments: 1) to provide an environment for human teachers to practise their teaching approaches; 2) to provide an environment for testing different pedagogical instructional design efforts; 3) to provide simulated learners who can act as companions for human learners. Our research is mainly focused on the first and the second uses – to enable deep insight into the complex interaction of the factors affecting doctoral learners' attrition and time-to-degree leading to a better design of an educational system. Therefore, our research questions are formulated around investigations of how various factors influence time-to-degree, completion rates, and dropout rates of doctoral students. We are interested in answering the following research questions:

1. How does the number of classes (as a platform for social integration with peers – potential helpers) offered by a program(s) or taken by a learner, influence learners' time-to-degree and their propensity to persist or drop out?
2. How does the average class size (as basis of learners' social integration) attended by learners, impact learners' time-to-degree and their inclination to persist or drop out? What is the optimum class size?
3. How does the overall population size of the learners (a few learners vs many learners) influence learners' time-to-degree and their likelihood to persist or drop out?
4. Does timely help affects doctoral learners' time-to-degree and their decision to persist or drop out? If so, how?
5. How does the level of reciprocation influence the formation of a 'helpful community' of learners and adaptive help seeking behavior of the learners?

Use of simulation enables us to explore the aforementioned issues in a fine-grained controlled environment. For example, it would be almost impossible in the 'real world' setting to examine the impact of different number of course to take or class size to attend. Two cohorts of learners will have different attributes. Simulation allows us to tweak the number of courses or class size without touching the other characteristics of learners. Hence, we are able to see the real impact of one variable at a time. Before any exploration and insight can be gained on these issues, there is need to design and implement the simulation model.

3 Building an Initial Prototype of *SimGrad*

In this section we demonstrate the steps we have taken in the development of our initial prototype of our simulated doctoral learning environment: *SimGrad*. We show how a designer of an educational technology can develop a model of their target learning environment and inform its initial condition with available 'real world' data.

3.1 *SimGrad* Design

We need to design a simulation model by addressing two key challenges. First, we need to consider issues related to the modeling of the learning environment: how do we design conceptual and computational models of a doctoral program and what stakeholders should be included in these models? The second concern is about modeling of simulated learners: what doctoral learners' features affect persistence and time-to-degree, what factors do we model, and can we inform these features with available 'real world' data?

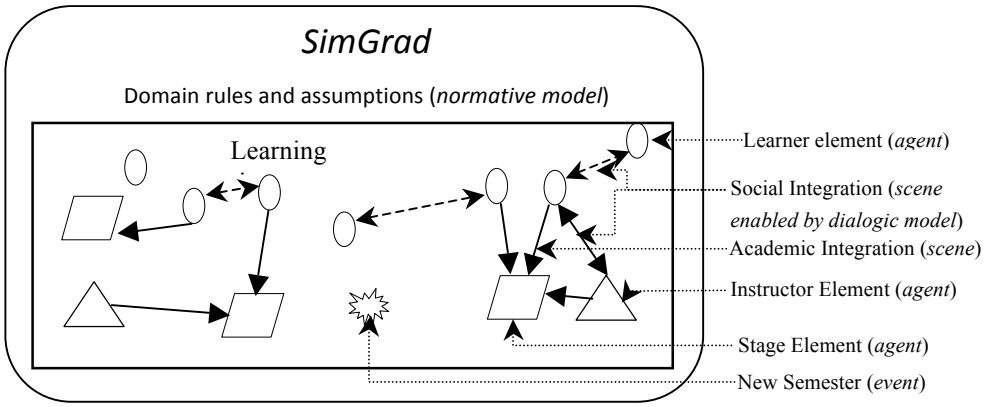

Fig. 1. SimGrad conceptual framework, its three elements, and the possible interaction between the elements

We have designed our conceptual model of the different aspects of simulated doctoral learners and doctoral learning environment based on the simulated learning environment specifications suggested by Koper et al. in [14], and features for building an electronic institution proposed by Esteva et al. [15]. We name our conceptual framework, *SimGrad*. Its core elements include: *normative model* - specifies requirements and constraints to guide agent actions and behavior; *dialogic model* – deals with interaction strategies and communication mechanism; *events* – refers to happenings in the model that trigger (re)action by agents; *scene* – description of an interaction between elements; *elements (agents)* – represent key stakeholders of the target domain that are modeled. Elements are modeled as agents. Each of the agents has attributes and behavior which are informed by our assumptions guided by our research questions and factors that influence learners' decision to persist. Every element of interest is to be

modeled within the learning environment and all possible interactions and operations within the learning setting is guided by domain rules represented by the normative model. See **Fig. 1**.

In our simulation model, we have chosen to model three types of elements: class, instructor, and learner. In this paper, in keeping with model simplicity, both the class and the instructor agents are passive while the learner agent is modeled to be active and reactive to its environment. Also, the only instructor's attributes we are interested in are related to classes (see **Table 1**). We modeled only one type of instructor agent. Another instructor type agent that can be modeled is the supervisor.

Each learner agent has the following properties: autonomy, social ability, reactivity, proactivity, and a degree of intentionality. We have also identified the following key attributes for our agent learner model: state – (busy, available), program, stages, course taken, peer interactions (pertaining challenges), academic integration, social integration, and motivation (see **Table 2**). In our model, peer interaction and state contribute to a learners' social integration, while research area, stage, course taken impact to their academic integration. Motivation combines both the social and academic integration and hence, is the main factor that determines whether an agent continues to persist or chooses to drop out of the program.

Table 1. Comparison of computed attributes of the three agent types

Attribute – data (value range)	*Agent learner*	*Agent instructor*	*Agent class*
Total number of classes take, taught, or frequency of offering within 10 years – *numeric (0-20)*	X	X	X
Grade obtained, average awarded, or average obtained by learners – *numeric (0,12)*	X	X	X
Take classes from, teach classes in, or class offered in various programs – *textual (program id)*	X	X	X
Instructors teaching a class – *array list (instructor id)*	X	-	X
What is the class size – *numeric (1-5)*	X		X
Number of classes taken or taught per year - *numeric (0,4)*	X	X	-
Which classes are taken or taught – *textual (class id)*	X	X	-

The main intentions of each agent is to persist through doctoral requirements to graduation and to do so in a timely manner. However, each of these agents reacts to the different challenges at various stages of graduate school in divergent and autonomous ways. At the coursework stage, agents have the goal of taking courses that are relevant to their field and that they will perform well. When facing a course choice challenge or any other particular challenge, we have modeled our agents to proactively associate with peers to seek help. Each peer makes individual choice on whether to or not to respond to a request for help from others. The dialogic model

handles the agent to agent interaction and communication through a message passing mechanism [16].

Table 2. Attributes and parameters considered for an agent learner model for learners, their description and how each of them changes.

Attribute	*Value - description*	*How it changes*
Enrolment	Date (MM/YYYY) Indicate the month a year an agent enrolled in the program	Does not change
Graduation	Date (MM/YYYY) Target graduation date	Evaluated whenever an agent completes a milestone
State	Textual (busy, available) Indicates an agent availability to help others, assigned based on the smallest time unit model	Changes whenever an agent experiences a challenge
Program	Textual (program id) Identify an agent's closer community within the larger community of learners	Does not change during a simulation run
Stage	Textual (admission, coursework, dissertation, timeout, dropout)	Admission stage is like an event. Learner move to the coursework immediately after admission. They more to dissertation after completing their course load.
Courses taken	Array [course, mark, instructor id](0-6) Record courses taken by an agent and the marks obtain in each course	Every end of semester that the student took classes, this array is updated
Peer interaction	Array [learner id, challenge, result], Keep track of an agent interactions with others and the outcome of the interaction	Changes whenever two agents interact
Academic integration	Numeric (-1,1) Measures the academic satisfaction	Changes whenever an agent learner interacts with agent stage (i.e., completes a milestone or experience a challenge)
Social integration	Numeric (-1,1) Measures a learners sense of belonging to the learning environment	Changes whenever an agent learner interacts with its peers or agent instructors
Motivation	Numeric (-1,1) Measures the propensity of an agent to still want to persist. A motivation value above 0.3 indicates persistence. A value between -0.3 and 0.3 indicate help seeking needed. A value below -0.3 means the agent drops out	Whenever there is a change in the social and academic integration values. Its value is the average of the integration values.

3.2 Informing *SimGrad* behavior and evaluation functions

Having identified the important agents and their key attributes, there are two sets of important functions for each element that need to be modelled: behaviour functions and evaluation functions [17]. Behaviour functions inform the decision making of the active elements and dictates the interaction patterns between them and the other modeled elements (e.g., how many classes a given agent takes). Evaluation functions indicate whether or not various interactions between the different agents in a simulation were successful (e.g., determine what grade a given agent attains in a class it took). Informing such functions with 'real world' data allows the simulation to behave in a way consistent with reality.

Simulation model fidelity is an issues that might arise when using simulation to study a target real world phenomenon. However, the most important issue to consider is the research question to be answered. While Champaign [18] used a very low fidelity model, Matsuda et al. [19] used a model with high cognitive fidelity to reach compelling conclusion. Further yet, Erickson et al. [17] also demonstrated that is possible to use a medium fidelity model and uncover interesting results. In some situations it might not be possible to have a high fidelity model because of lack of data. A case in point is our simulation scenario. Where possible, we inform our simulation functions with data received from the U of S on their doctoral program. An investigation into the U of S data showed that we will not be able to inform every aspect of our simulation model. It would be desirable to inform every initial aspects of our simulation model with 'real world' data but, we do not have data on the dissertation stage.

We are provided information on student id, years a student is registered, year of graduation (if graduated), student's program, classes taken and marks obtained, class instructor, and students instructional responsibilities. From this dataset we are able to inform the admission and coursework stages of our model (academic integration). However, there is no information concerning the dissertation stage and the social integration aspects. While it possible to inform various behaviour and evaluation functions for our simulation model, in this paper we focus on describing the steps we took to inform two functions of our simulation: learning environment admission behaviour function, and learners' class interactions behaviour function.

As already mentioned, admission is an important part of a doctoral program that contributes to it dynamic nature. The admission process is complex and involves a lot of stakeholders and processes, but we are concerned only with determining the year to year patterns in how many students are admitted. To provide some fidelity to our simulated learning environment admission, we analyzed data provided to us by the U of S University Data Warehouse[2]. The provided dataset contained information on doctoral learners registered in the 10 years 2005-2014. In this time there were 2291 doctoral learners with a total of 52850 data points on class registration. The 2005 registration included learners who had joined the program earlier than 2005. In order to get a clean admission pattern, we only considered learners who were registered from the year 2006 onwards. This reduced the population size to 1962.

[2] http://www.usask.ca/ict/services/ent-business-intelligence/university-data-warehouse.php

We were able to identify three admission periods, September, January, and May. We then obtained values for each of the admissions months for the years 2006-2014. This provided a distribution for each month that we used to generate a scatter plot of admission numbers. A sigmoidal pattern emerged. Next, we performed a non-linear curve fitting to the scatter plot so that the admission function can be represented in the form $Y = St*(c + x)$, where c is a constant, St is a variable dependent on the admission period, and x is the admission period. We then ran a regression to find values of each of these variables. This allowed us to model the admission patterns observed in the U of S dataset.

Next we derived the number of classes taken. To introduce some realism to the number classes taken behaviour, we had to further prune the data. We only considered data for students whose cohorts would have been registered for at least 3 years by the end of the year 2014 and hence, we considered class taking behaviour of 1466 U of S doctoral learners.

We obtained the number of classes each of the remaining learners we registered in and created a histogram. This histogram showed us the distribution of the number of students registered for a certain number of classes. Next, we transformed this distribution graph into a cumulative distribution function. We then took an inverse of the cumulative distribution function to achieve a quantile function. The quantile function, when run over many learners, assigns learners a class count that mimics the initial histogram. We use this quantile function to inform the number of classes a learner can take.

In this section we have described the importance of informing a simulation model with 'real world' data. We have described two functions that are informed with U of S dataset. Other examples of functions that can be informed using the U of S dataset include: class performance evaluation function, dropout behaviour function, time to degree behaviour function, and flow through behavior function (main as pertains to coursework stage). We have identified that missing data values is a major hindrance in this endeavor. There are possible ways of informing simulation attributes where there are no 'real world' data to derive from. A designer can either assign common sense values, generate and assign random values, or refer to the research literature to identify patterns that have been found by other researchers. Since we have the enrolment dates and the graduate dates for learners who graduate, we choose to derive common sense values with these two dates guiding the process and the value range.

4 Discussion, Expected Contributions, and Future Research Plans

Despite the growth in the use of simulation as a method for exploration and learning in many areas such as: engineering, nursing, medicine [20], and building design [21], research in the used of simulation within AIED is still at an early stage. There is need for more research to demonstrate that the outputs of simulation runs are desirable and informative to the AIED community. In this paper, we aim at contributing to this notion and by promoting the use of simulation in educational research and presenting

an agent based simulation conceptual framework for building simulated learning environment, with a focus on the semi-structured ones. Simulated learning environment and simulated learners are important in exploring and understanding a given learning domain. Further, it helps with the generation of system validation data.

The expected contributions to AIED include: providing a conceptual framework for simulated graduate school learning environment – an architecture that enables investigations into factors affecting doctoral learners progress through their program; shedding light on learner modeling issues in dynamic learning environments; and demonstrating the importance of simulation in exploring various AIED research domains, particularly semi-structured domains.

Current research work is focused on the implementation of the simulation model and the refinement of the various behaviour and evaluation functions. Once the implementation is done, we will validate our model against the dataset we have from the U of S before proceeding to explore the impact of various environmental factors. Since we are informing the simulation with both common sense assumptions and U of S dataset, the goal is to tweak the common sense assumptions such that when the model is run we get similar results as the U of S data in terms of class performance, dropout rate, and time-to-degree. Achieving this, would give us confidence that we have captured reality in some measurable way. We can then start exploring the various impact of measures we are interested in examining. As earlier indicated, we are interested in exploring the interactions of a number of variables: number of classes taken which will impact the availability of potential peer helpers, the effect of reciprocity on help seeking and help giving, and the effect of help seeking and other factors on doctoral learners' time-to-degree and attrition rates.

Acknowledgement

We would like to thank University of Saskatchewan University Data Warehouse team for giving us access to 10 year dataset. Specifically, we would like to thank Mathew Zip for processing and explaining to the first author the nature of the dataset. We also wish to acknowledge and thank the Natural Science and Engineering Research Council of Canada (NSERC) for funding our research.

References

[1] D. E. K. Lelei, "Supporting Lifelong Learning: Recommending Personalized Sources of Assistance to Graduate Students," in *Artificial Intelligence in Education*, 2013, pp. 912–915.

[2] V. Tinto, "Taking student success seriously: Rethinking the first year of college.," *Ninth Annu. Intersession Acad. Aff. Forum,* vol. 19, no. 2, pp. 1–8, 2005.

[3] V. Tinto, "Dropout from higher education: A theoretical synthesis of recent research," *Rev. Educ. Res.*, vol. 45, no. 1, pp. 89–125, 1975.

[4] H. Groenvynck, K. Vandevelde, and R. Van Rossem, "The Ph.D. Track: Who Succeeds, Who Drops Out?," *Res. Eval.*, vol. 22, no. 4, pp. 199–209, 2013.

[5] F. J. Elgar, “Phd Degree Completion in Canadian Universities,” Nova Scotia, Canada, 2003.
[6] M. Walpole, N. W. Burton, K. Kanyi, and A. Jackenthal, “Selecting Successful Graduate Students: In-Depth Interviews With GRE Users,” Princeton, NJ, 2002.
[7] L. Declou, “Linking Levels to Understand Graduate Student Attrition in Canada,” McMaster University, Hamilton, Ontario, Canada, 2014.
[8] B. E. Lovitts, *Leaving the Ivory Tower: The Causes and Consequences of Departure from Doctoral Study*, Illustrate., vol. 32. Rowman & Littlefield Publishers, 2001, p. 307.
[9] A. Ali and F. Kohun, “Dealing with isolation feelings in IS doctoral programs,” *Int. J. Dr. Stud.*, vol. 1, no. 1, pp. 21–33, 2006.
[10] Computing Research Association, “Grand research challenges in information systems,” Washington, D.C, 2003.
[11] R. Axelrod, “Advancing the Art of Simulation in the Social Sciences,” in *Proceedings of the 18th European Meeting on Cybernetics and Systems Research*, 2006, pp. 1–13.
[12] N. Gilbert and K. G. Troitzsch, “Simulation and Social Science,” in *Simulation for the Social Scientist*, McGraw-Hill International, 2005, pp. 1–14.
[13] K. VanLehn, S. Ohlsson, and R. Nason, “Applications of Simulated Students: An Exploration,” *J. Artif. Intell. Educ.*, vol. 5, no. 2, pp. 1–42, 1994.
[14] R. Koper and B. Olivier, “Representing the Learning Design of Units of Learning,” *Educ. Technol. Soc.*, vol. 7, no. 3, pp. 97–111, 2003.
[15] M. Esteva, J.-A. Rodriquez-Aguilar, C. Sierra, P. Garcia, and J. L. Arcos, “On the Formal Specification of Electronic Institutions,” in *Agent Mediated Eectronic Commerce*, 2001, pp. 126–147.
[16] H. J. C. Berendsen, D. Van Der Spoel, and R. Van Drunen, “GROMACS: A message-passing parallel molecular dynamics implementation,” *Comput. Phys. Commun.*, vol. 91, no. 1, pp. 43–56, 1995.
[17] G. Erickson, S. Frost, S. Bateman, and G. Mccalla, “Using the Ecological Approach to Create Simulations of Learning Environments,” in *In Artificial Intelligence in Education*, 2013, pp. 411–420.
[18] J. Champaign, “Peer-Based Intelligent Tutoring Systems: A Corpus-Oriented Approach,” University of Waterloo, 2012.
[19] N. Matsuda, W. W. Cohen, K. R. Koedinger, V. Keiser, R. Raizada, E. Yarzebinski, S. P. Watson, and G. Stylianides, “Studying the Effect of Tutor Learning Using a Teachable Agent that Asks the Student Tutor for Explanations,” in *2012 IEEE Fourth International Conference On Digital Game And Intelligent Toy Enhanced Learning*, 2012, pp. 25–32.
[20] A. L. Baylor and Y. Kim, “Simulating Instructional Roles Through Pedagogical Agents,” *Int. J. Artif. Intell. Educ.*, vol. 15, no. 2, pp. 95–115, 2005.
[21] G. Augenbroe, “Trends in Building Simulation,” *Build. Environ.*, vol. 37, no. 8, pp. 891–902, 2002.

Exploring the Role of Small Differences in Predictive Accuracy using Simulated Data

Juraj Nižnan, Jan Papoušek, and Radek Pelánek

Faculty of Informatics, Masaryk University Brno
{niznan,jan.papousek,xpelanek}@mail.muni.cz

Abstract. Research in student modeling often leads to only small improvements in predictive accuracy of models. The importance of such improvements is often hard to assess and has been a frequent subject of discussions in student modeling community. In this work we use simulated students to study the role of small differences in predictive accuracy. We study the impact of such differences on behavior of adaptive educational systems and relation to interpretation of model parameters. We also point out a feedback loop between student models and data used for their evaluation and show how this feedback loop may mask important differences between models.

1 Introduction

In student modeling we mostly evaluate models based on the quality of their predictions of student answers as expressed by some performance metric. Results of evaluation often lead to small differences in predictive accuracy, which leads some researchers to question the importance of model improvements and meaningfulness of such results [1]. Aim of this paper is to explore the impact and meaning of small differences in predictive accuracy with the use simulated data. For our discussion and experiments in this work we use a single performance metric – Root Mean Square Error (RMSE), which is a common choice (for rationale and overview of other possible metrics see [15]). The studied questions and overall approach are not specific to this metric.

Simulated students provide a good way to study methodological issues in student modeling. When we work with real data, we can use only proxy methods (e.g., metrics like RMSE) to evaluate quality of models. With simulated data we know the "ground truth" so we can study the link between metrics and the true quality of models. This enables us to obtain interesting insight which may be useful for interpretation of results over real data and for devising experiments. Similar issues are studied and explored using simulation in the field of recommender systems [7, 17].

We use a simple setting for simulated experiments, which is based on an abstraction of a real system for learning geography [12]. We simulate an adaptive question answering system, where we assume items with normally distributed difficulties, students with normally distributed skills, and probability of correct

answer given by a logistic function of the difference between skill and difficulty (variant of a Rasch model). We use this setting to study several interrelated question.

1.1 Impact on Student Practice

What is the impact of prediction accuracy (as measured by RMSE) on the behavior of an adaptive educational system and students' learning experience?

Impact of small differences in predictive performance on student under-practice and over-practice (7-20%) has been demonstrated using real student data [18], but insight from a single study is limited. The relation of RMSE to practical system behavior has been analyzed also in the field of recommender systems [2] (using offline analysis of real data). This issue has been studied before using simulated data is several studies [5, 6, 10, 13]. All of these studies use very similar setting – they use Bayesian Knowledge Tracing (BKT) or its extensions and their focus is on mastery learning and student under-practice and over-practice. They differ only in specific aspects, e.g., focus on setting thresholds for mastery learning [5] or relation of moment of learning to performance metrics [13]. In our previous work [16] have performed similar kind of simulated experiments (analysis of under-practice and over-practice) both with BKT and with student models using logistic function and continuous skill.

In this work we complement these studies by performing simulated experiments in slightly different setting. Instead of using BKT and mastery learning, we use (variants of) the Rasch model and adaptive question answering setting. We study different models and the relation between their prediction accuracy and the set of items used by the system.

1.2 Prediction Accuracy and Model Parameters

Can RMSE be used to identify good model parameters? What is the relation of RMSE to the quality of model parameters?

In student modeling we often want to use interpretable models since we are interested not only in predictions of future answers, but also in reconstructing properties of students and educational domains. Such outputs can be used to improve educational systems as was done for example by Koedinger at al. [9]. When model evaluation shows that model A achieves better prediction accuracy (RMSE) then model B, results are often interpreted as evidence that model A better reflects "reality". Is RMSE a suitable way to find robust parameters? What differences in metric value are meaningful, i.e., when we can be reasonably sure that the better model really models reality in better way? Is statistical significance of differences enough? In case of real data it is hard to answer these question since we have no direct way to evaluate the relation of a model to reality. However, we can study these questions with simulated data, where we have access to the ground truth parameters. Specifically, in our experiments we study the relation of metric values with the accuracy of reconstructing the mapping between items and knowledge components.

1.3 Feedback between Data Collection and Evaluation

Can the feedback loop between student models and adaptive choice of items influence evaluation of student models?

We also propose novel use of simulated students to study a feedback loop between student models and data collection. The data that are used for model evaluation are often collected by a system which uses some student model for adaptive choice of items. The same model is often used for data collection and during model evaluation. Such evaluation may be biased – it can happen that the used model does not collect data that would show its deficiencies. Note that the presence of this feedback loop is an important difference compared to other forecasting domains. For example in weather forecasting models do not directly influence the system and cannot distort collected data. In student modeling they can.

So far this feedback has not been thoroughly studied in student modeling. Some issues related to this feedback have been discussed in previous work on learning curves [6, 11, 8]. When a tutoring system uses mastery learning, students with high skill drop out earlier from the system (and thus from the collected data), thus a straightforward interpretation of aggregated learning curves may be misleading. In this work we report experiment with simulated data which illustrate possible impact of this feedback loop on model evaluation.

2 Methodology

For our experiments we use a simulation of a simplified version of an adaptive question answering systems, inspired by our widely used application for learning geography [12]. Fig. 1 presents the overall setting of our experiments. System asks students about items, answers are dichotomous (correct/incorrect), each student answers each item at most once. System tries to present items of suitable difficulty. In evaluation we study both the prediction accuracy of models and also sets of used items. This setting is closely related to item response theory and computerized adaptive testing, specifically to simulated experiments with Elo-type algorithm reported by Doebler et al. [3].

Simulated Students and Items We consider a set of simulated students and simulated items. To generate student answers we use logistic function (basically the Rasch model, respectively one parameter model from item response theory): $P(correct|\theta_s, d_i) = 1/(1 + e^{-(\theta_s - d_i)})$, where θ_s is the skill of a student s and d_i is difficulty of an item i.

To make the simulated scenarios more interesting we also consider multiple knowledge components. Items are divided into disjoint knowledge components and students have different skill for each knowledge component. Student skills and item difficulties are sampled from a normal distribution. Skills for individual knowledge components are independent from one another.

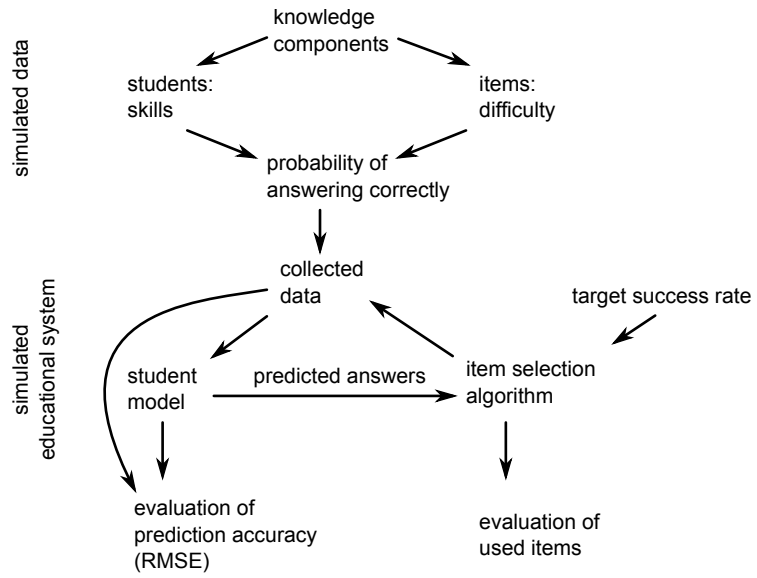

Fig. 1. Setting of our experiments

Item Selection Algorithm The item selection algorithm has as a parameter a target success rate t. It repeatedly presents items to a (simulated) student, in each step it selects an item which has the best score with respect to the distance of the predicted probability of correct answer p and the target rate t (illustrated by gray dashed line in Fig. 3). If there are multiple items with the same score, the algorithm randomly selects one of them.

Student Models Predictions used by the item selection algorithm are provided by a student model. For comparison we consider several simple student models:

- Optimal model – Predicts the exact probability that is used to generate the answer (i.e., a "cheating" model that has access to the ground truth student skill and item difficulty).
- Optimal with noise – Optimal model with added (Gaussian) noise to the difference $\theta_s - d_i$ (before we apply logistic function).
- Constant model – For all students and items it provides the same prediction (i.e., with this model the item selection algorithm selects items randomly).
- Naive model – Predicts the average accuracy for each item.
- Elo model – The Elo rating system [4, 14] with single skill. The used model corresponds to the version of the system as described in [12] (with slightly modified uncertainty function).
- Elo concepts – The Elo system with multiple skills with correct mapping of items to knowledge components.

– Elo wrong concepts – The Elo system with multiple skills with wrong mapping of items to knowledge components. The wrong mapping is the same as the correct one, but 50 (randomly choosen) items are classified incorrectly.

Data We generated 5,000 students and 200 items. Items are divided into 2 knowledge components, each user has 2 skills corresponding to the knowledge components and each item has a difficulty. Both skills and difficulties were sampled from standard normal distribution (the data collected from the geography application suggests that these parameters are approximately normally distributed). The number of items in a practice session is set to 50 unless otherwise noted.

3 Experiments

We report three types of experiments, which correspond to the three types of questions mentioned in the introduction.

3.1 Impact on Student Practice

Our first set of experiments studies differences in the behavior of the simulated system for different models. For the evaluation of model impact we compare the sets of items selected by the item selection algorithm. We make the assumption that the algorithm for item selection using the optimal model generates also the optimal practice for students. For each user we simulate practice of 50 items (each item is practiced at most once by each student). To compare the set of practiced items between those generated by the optimal model and other models we look at the size of the intersection. We assume that bigger intersection with the set of practiced items using the optimal model indicates better practice. Since the intersection is computed per user, we take the mean.

This is, of course, only a simplified measure of item quality. It is possible that an alternative model selects completely different set of items (i.e., the intersection with the optimal set is empty) and yet the items are very similar and their pedagogical contribution is nearly the same. However, for the current work this is not probable since we are choosing 50 items from a pool of only 200 items. For future work it would be interesting to try to formalize and study the "utility" of items.

Noise Experiment The optimal model with noise allows us to easily manipulate differences in predictive accuracy and study their impact on system behavior. Experiment reported in the left side of Fig. 2 shows both the predictive accuracy (measured by RMSE) and the impact on system behavior (measured by the size of the intersection with the optimal practiced set as described above) depending on the size of noise (we use Gaussian noise with a specified standard deviation). The impact of noise on RMSE is approximately quadratic and has a slow rise –

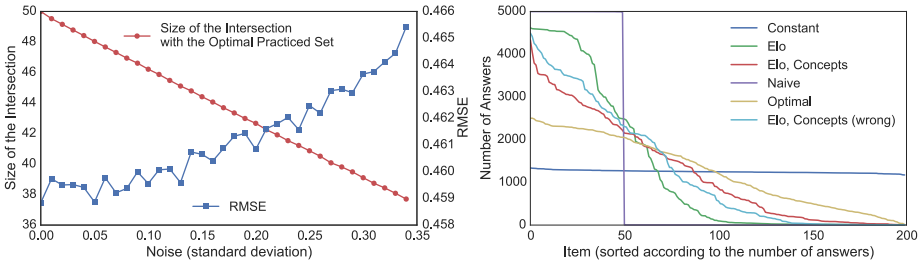

Fig. 2. Size of the intersection with the optimal practiced set of items and RMSE depending on Guassian noise in optimal model (left side). Distribution of answers over the items based on the given model (right side).

this is a direct consequence of the quadratic nature of the metric. The impact on used items is, however, approximately linear and rather steep. The most interesting part is for noise values in the interval $[0, 0.1]$. In this interval the rise in RMSE values is very small and unstable, but the impact on used items is already high.

Model Comparison Right side of the Fig. 2 shows the distribution of the number of answers per item for different models. The used models have similar predictive accuracy (specific values depend on what data we use for their evaluation, as discussed below in Section 3.3), yet the used model can dramatically change the form of the collected data.

When we use the optimal model, the collected data set covers almost fairly most items from the item pool. In the case of worse models the use of items is skewed (some items are used much more frequently than others). Obvious exception is the constant model for which the practice is completely random. The size of the intersection with the optimal practiced set for these models is – Constant: 12.5; Elo: 24.2; Elo, Concepts: 30.4; Elo, Concepts (wrong): 28.5; Naive: 12.0. Fig. 3 presents a distribution of answers according to the true probability of their correctness (given by the optimal model). Again there is a huge difference among the given models, especially between simple models and those based on Elo.

3.2 Prediction Accuracy and Model Parameters

Metrics of prediction accuracy (e.g., RMSE) are often used for model selection. Model that achieves lower RMSE is assumed to have better parameters (or more generally better "correspondence to reality"). Parameters of a selected model are often interpreted or taken into account in improvement of educational systems. We checked validity of this approach using experiments with knowledge components.

We take several models with different (random) mappings of items to knowledge components and evaluate their predictive accuracy. We also measure the

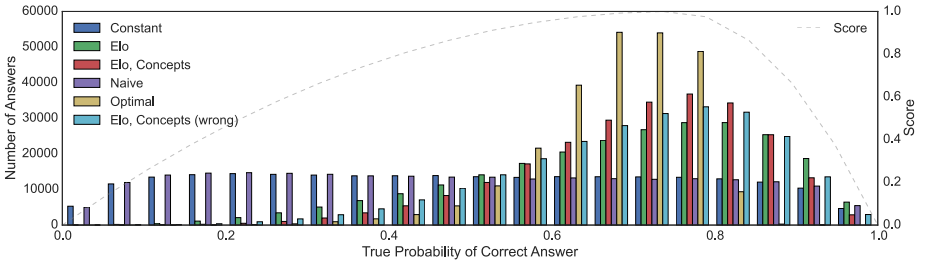

Fig. 3. Distribution of answers according to the true probability of correct answer. The gray dashed line stands for the score function used by the algorithm for item selection.

quality of the used mappings – since we use simulated data, we know the ground truth mapping and thus can directly measure the quality of each mapping. Quality is expressed as the portion of items for which the mapping agrees with the ground truth mapping. The names of the knowledge components are irrelevant in this setting. Therefore, we compute quality for each one-to-one mapping from the names of the components in the model to the names of the components in the ground truth. We select the highest quality as the quality of the model's item-to-component mapping. To focus only on quality of knowledge components, we simplify other aspects of evaluation, specifically each student answers all items and their order is selected randomly.

These experiments do not show any specific surprising result, so we provide only general summary. Experiments show that RMSE values correlate well with the quality of mappings. In case of small RMSE differences there may be "swaps", i.e., a model with slightly higher RMSE reflects reality slightly better. But such results occur only with insufficiently large data and are unstable. Whenever the differences in RMSE are statistically significant (as determined by t-test over different test sets), even very small differences in RMSE correspond to improvement in the quality of the used mappings. These results thus confirm that it is valid (at least in the studied setting) to argue that a model A better corresponds to reality than a model B based on the fact that the model A achieves better RMSE than the model B (as long as the difference is statistically significant). It may be useful to perform this kind of analysis for different settings and different performance metrics.

3.3 Feedback between Data Collection and Evaluation

To study feedback between the used student model and collected data (as is described in subsection 1.3) we performed the following experiment: We choose one student model and use it as an input for adaptive choice of items. At the same time we let all other models do predictions as well and log answers together with all predictions.

Fig. 4 shows the resulting RMSE for each model in individual runs (data collected using specific model). The figure shows several interesting results. When

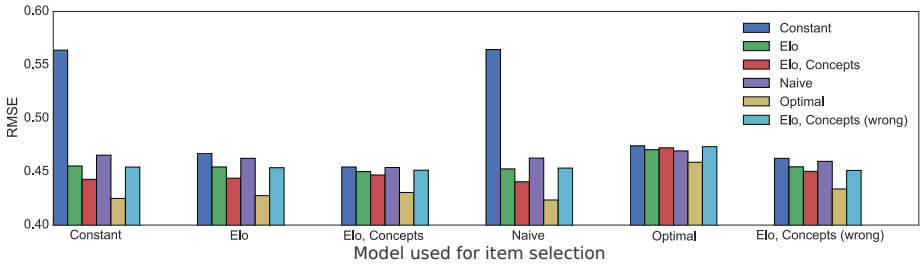

Fig. 4. RMSE comparison over data collected using different models.

the data are collected using the optimal model, the RMSE values are largest and closest together; even the ordering of models is different from other cases. In this case even the constant model provides comparable performance to other models – but it would be very wrong to conclude that "predictive accuracy of models is so similar that the choice of model does not matter". As the above presented analysis shows, different models lead to very different choice of items and consequently to different student experience. The reason for small differences in RMSE is not similarity between models, but characteristics of data ("good choice of suitable items"), which make predictions difficult and even a naive predictor comparatively good.

Another observation concerns comparison between the "Elo concepts" and "Elo concepts (wrong)" models. When data are collected by the "Elo concepts (wrong)" model, these two models achieve nearly the same performance, i.e., models seem to be of the same quality. But the other cases show that the "Elo concepts" model is better (and in fact it is by construction a better student model).

4 Conclusions

We have used simulated data to show that even small differences in predictive accuracy of student models (as measured by RMSE) may have important impact on behavior of adaptive educational systems and for interpretation of results of evaluation. Experiments with simulated data, of course, cannot demonstrate the practical impact of such small differences. We also do not claim that small differences in predictive accuracy are always important. However, experiments with simulated data are definitely useful, because they clearly illustrate mechanisms that could play role in interpretation of results of experiments with real student data. Simulated data also provide setting for formulation of hypotheses that could be later evaluated in experiments with real educational systems.

Simulated data also enable us to perform experiments that are not practical for realization with actual educational systems. For example in our experiment with the "feedback loop" we have used different student models as a basis for item selection. Our set of models includes even a very simple "constant model",

which leads to random selection of practiced item. In real setting we would be reluctant to apply such a model, as it is in contrary with the advertised intelligent behavior of our educational systems. However, experiments with this model in simulated setting provide interesting results – they clearly demonstrate that differences in predictive accuracy of models do not depend only on the intrinsic quality of used student models, but also on the way the data were collected.

Our analysis shows one particularly interesting aspect of student modeling. As we improve student models applied in educational systems, we should expect that evaluations of predictive accuracy performed over these data will show worse absolute values of performance metrics and smaller and smaller differences between models (even if models are significantly different), just because virtues of our models enable us to collect less predictable data.

References

1. Joseph E Beck and Xiaolu Xiong. Limits to accuracy: How well can we do at student modeling. In *Proc. of Educational Data Mining*, 2013.
2. Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.
3. Philipp Doebler, Mohsen Alavash, and Carsten Giessing. Adaptive experiments with a multivariate elo-type algorithm. *Behavior research methods*, pages 1–11, 2014.
4. Arpad E Elo. *The rating of chessplayers, past and present*, volume 3. Batsford London, 1978.
5. Stephen E Fancsali, Tristan Nixon, and Steven Ritter. Optimal and worst-case performance of mastery learning assessment with bayesian knowledge tracing. In *Proc. of Educational Data Mining*, 2013.
6. Stephen E Fancsali, Tristan Nixon, Annalies Vuong, and Steven Ritter. Simulated students, mastery learning, and improved learning curves for real-world cognitive tutors. In *AIED Workshops*. Citeseer, 2013.
7. Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.
8. Tanja Käser, Kenneth R Koedinger, and Markus Gross. Different parameters-same prediction: An analysis of learning curves. In *Proceedings of 7th International Conference on Educational Data Mining. London, UK*, 2014.
9. Kenneth R Koedinger, John C Stamper, Elizabeth A McLaughlin, and Tristan Nixon. Using data-driven discovery of better student models to improve student learning. In *Artificial intelligence in education*, pages 421–430. Springer, 2013.
10. Jung In Lee and Emma Brunskill. The impact on individualizing student models on necessary practice opportunities. *International Educational Data Mining Society*, 2012.
11. R Charles Murray, Steven Ritter, Tristan Nixon, Ryan Schwiebert, Robert GM Hausmann, Brendon Towle, Stephen E Fancsali, and Annalies Vuong. Revealing the learning in learning curves. In *Artificial Intelligence in Education*, pages 473–482. Springer, 2013.

12. Jan Papoušek, Radek Pelánek, and Vít Stanislav. Adaptive practice of facts in domains with varied prior knowledge. In *Proc. of Educational Data Mining*, pages 6–13, 2014.
13. Zachary A Pardos and Michael V Yudelson. Towards moment of learning accuracy. In *AIED 2013 Workshops Proceedings Volume 4*, page 3, 2013.
14. Radek Pelánek. Application of time decay functions and Elo system in student modeling. In *Proc. of Educational Data Mining*, pages 21–27, 2014.
15. Radek Pelánek. Metrics for evaluation of student models. *Journal of Educational Data Mining*, 2015. To appear.
16. Radek Pelánek. Modeling student learning: Binary or continuous skill? In *Proc. of Educational Data Mining*, 2015.
17. Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M McNee, Joseph A Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 127–134. ACM, 2002.
18. Michael V Yudelson and Kenneth R Koedinger. Estimating the benefits of student model improvements on a substantive scale. In *EDM 2013 Workshops Proceedings*, 2013.

Using Data from Real and Simulated Learners to Evaluate Adaptive Tutoring Systems

José P. González-Brenes[1], Yun Huang[2]

[1] Pearson School Research & Innovation Network, Philadelphia, PA, USA
jose.gonzalez-brenes@pearson.com
[2] Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA, USA
yuh43@pitt.edu

Abstract. Classification evaluation metrics are often used to evaluate adaptive tutoring systems— programs that teach and adapt to humans. Unfortunately, evidence suggests that existing convention for evaluating tutoring systems may lead to suboptimal decisions. In a companion paper, we propose Teal, a new framework to evaluate adaptive tutoring. In this paper we propose an alternative formulation of Teal using simulated learners. The main contribution of this novel formulation is that it enables approximate inference of Teal, which may useful on the cases that Teal becomes computationally intractable. We believe that this alternative formulation is simpler, and we hope it helps as a bridge between the student modeling and simulated learners community.

1 Introduction

Adaptive systems teach and adapt to humans and improve education by optimizing the subset of *items* presented to students, according to their historical performance [3], and on features extracted from their activities [6]. In this context, items are questions, or tasks that can be graded individually. Adaptive tutoring may be evaluated with randomized control trials. For example, in a seminal study [3] that focused on earlier adaptive tutors, a controlled trial measured the time students spent on tutoring, and their performance on post-tests. The study reported that the adaptive tutoring system enabled significantly faster teaching, while students maintained the same or better performance on post-tests

Unfortunately, controlled trials can become extremely expensive and time consuming to conduct: they require institutional review board approvals, experimental design by an expert, recruiting and often payment of enough participants to achieve statistical power, and data analysis. Automatic evaluation metrics improve the engineering process because they enable less expensive and faster comparisons between alternative systems.

The adaptive tutoring community has tacitly adopted conventions for evaluating tutoring systems [4]. Researchers often evaluate their models with classification evaluation metrics that assess the *student model* component of the tutoring system— student models are the subsystems that forecast whether a learner will answer the next item correctly. However, automatic evaluation metrics are

intended to measure an outcome of the end user. For example, the PARADISE [9] metric used in spoken dialogue systems correlates to user satisfaction scores. We are not aware of evidence that supports that classification metrics correlate with learning outcomes; yet there is a growing body of evidence [2, 5] that suggests serious problems with them. For example, classification metrics ignore that an adaptive system may not help learners— which could happen with a student model with a flat or decreasing learning curve [1, 8]. A decreasing learning curve implies that student performance decreases with practice; this curve is usually interpreted as a modeling problem, because it operationalizes that learners are better off with no teaching.

We study a novel formulation of the Theoretical Evaluation of Adaptive Learning Systems (Teal) [5] evaluation metric. The importance of evaluation metrics is that they help practitioners and researchers quantify the extent that a system helps learners.

2 Theoretical Evaluation of Adaptive Learning Systems

In this section, we just briefly summarize Teal and do not compare it with a related method called ExpOppNeed [7]. Teal assumes the adaptive tutoring system is built using a single-skill Knowledge Tracing Family model [3, 6]. Knowledge Tracing uses a Hidden Markov Model (HMM) per skill to model the student's knowledge as latent variables. It models whether a student applies a practice opportunity of a skill correctly. The latent variables are used to model the latent student proficiency, which is often modeled with a binary variable to indicated mastery of the skill.

To use Teal on data collected from students, we first train a model using an algorithm from the Knowledge Tracing family, then we use the learned parameters to calculate the effort and outcome for each skill.

- Effort: Quantifies how much practice the adaptive tutor gives to students. In this paper we focus on counting the number of items assigned to students but, alternatively, amount of time could be considered.
- Outcome: Quantifies the performance of students after adaptive tutoring. For simplicity, we operationalize performance as the percentage of items that students are able to solve after tutoring. We assume that the performance on solving items is aligned to the long-term interest of learners.

Algorithm 1 describes our novel formulation. Teal calculates the expected number of practice that an adaptive tutor gives to students. We assume that the tutor stops teaching a skill once the student is very likely to answer the next item correctly according to a model from the Knowledge Tracing Family [6]. The adaptive tutor teaches an additional item if two conditions hold: (i) it is likely that the student will get the next item wrong— in other words, the probability of answering correctly the next item is below a threshold τ; and (ii) the tutor has not decided to stop instruction already.

The inputs of Teal are:

– Real student performance data from m students practicing a skill. Data from each student is encoded into a sequence of binary observations of whether the student was able to apply correctly the skill at different points in time.
– A threshold $\tau \in \{0 \dots 1\}$ that indicates when to stop tutoring. We operationalize this threshold as the target probability that the student will apply the skill correctly.
– A parameter T that indicates the number of practice opportunities each of the simulated students will practice the skill.

Algorithm 1 Teal algorithm for models with one skill per item

Require: real student data $\mathbf{y}^{(1)} \dots \mathbf{y}^{(m)}$, threshold τ, # of simulated time steps T
function TEAL
  $\theta \leftarrow$ Knowledge_Tracing($\mathbf{y}^{(1)} \dots \mathbf{y}^{(m)}$)
  $e \leftarrow \{\ \}$
  $s \leftarrow \{\ \}$
  for $\hat{\mathbf{y}} \in$ get_simulated_student(θ, T) **do**:
    $e \leftarrow$ calculate_effort($\hat{\mathbf{y}}, \theta, \tau$)
    if $e < T$ **then**
      $s \leftarrow$ calculate_score($\hat{\mathbf{y}}$, e)
    else
      $s \leftarrow$ imputed_value
  return mean(e), mean(s)

Teal learns a Knowledge Tracing model from the data collected from real students interacting with a tutor. Our new formulation uses simulated learners sampled from the Knowledge Tracing parameters. This enables us to decide how many simulated students to generate. Our original formulation required 2^m sequences to be generated, which can quickly become computationally intractable. If an approximate solution is acceptable, our novel formulation allows more efficient calculations of Teal. Teal quantifies the effort and outcomes of students in adaptive tutoring. Even though measuring effort and outcomes is not novel by itself, Teal's contribution is measuring both without a randomized trial. Teal quantifies effort as how much practice the tutor gives. For this, we count the number of items assigned to students. For a single simulated student, this is:

$$\text{calculate\_effort}(y_1, \dots, y_T, \theta, \tau) \equiv \arg\min_t p(y_t | y_1 \dots y_{t-1}, \theta) > \tau \qquad (1)$$

The threshold τ implies a trade-off between student effort and scores and responds to external expectations from the social context. Teal operationalizes the outcome as the performance of students after adaptive tutoring as the percentage of items that students are able to solve after tutoring:

$$\text{calculate\_score}(y_1, \dots, y_T, e) \equiv \sum_{t=e} \frac{\delta(\mathbf{y}_t, \text{correct})}{T - e} \qquad (2)$$

Here, $\delta(\cdot, \cdot)$ is the Kronecker function that returns 1 iff its arguments are equal.

3 Discussion

Simulation enables us to measure effort and outcome for a large population of students. Previously, we required Teal to be computed exhaustively on all student outcomes possibilities. We relax the prohibitively expensive requirement of calculating all student outcome combinations. Our contribution is that Teal can be calculated with a simulated dataset size that is large yet tractable.

References

1. R. Baker, A. Corbett, and V. Aleven. More Accurate Student Modeling through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. In B. Woolf, E. Aïmeur, R. Nkambou, and S. Lajoie, editors, *Intelligent Tutoring Systems*, volume 5091 of *Lecture Notes in Computer Science*, pages 406–415. Springer Berlin / Heidelberg, 2008.
2. J. Beck and X. Xiong. Limits to accuracy: how well can we do at student modeling? In S. K. D'Mello, R. A. Calvo, and A. Olney, editors, *Proceedings of the 6th International Conference on Educational Data Mining, Memphis, Tennessee, USA, July 6-9, 2013*, pages 4–11. International Educational Data Mining Society, 2013.
3. A. Corbett and J. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1995.
4. A. Dhanani, S. Y. Lee, P. Phothilimthana, and Z. Pardos. A comparison of error metrics for learning model parameters in bayesian knowledge tracing. Technical Report UCB/EECS-2014-131, EECS Department, University of California, Berkeley, May 2014.
5. González-Brenes and Y. José P., Huang. Your model is predictive— but is it useful? theoretical and empirical considerations of a new paradigm for adaptive tutoring evaluation. In J. G. Boticario, O. C. Santos, C. Romero, and M. Pechenizkiy, editors, *Proceedings of the 8th International Conference on Educational Data Mining*, Madrid, Spain, 2015.
6. J. P. González-Brenes, Y. Huang, and P. Brusilovsky. General Features in Knowledge Tracing: Applications to Multiple Subskills, Temporal Item Response Theory, and Expert Knowledge. In M. Mavrikis and B. M. McLaren, editors, *Proceedings of the 7th International Conference on Educational Data Mining*, London, UK, 2014.
7. J. I. Lee and E. Brunskill. The impact on individualizing student models on necessary practice opportunities. In K. Yacef, O. R. Zaïane, A. Hershkovitz, M. Yudelson, and J. C. Stamper, editors, *Proceedings of the 5th International Conference on Educational Data Mining*, pages 118–125, Chania, Greece, 2012.
8. D. Rai, Y. Gong, and J. E. Beck. Using dirichlet priors to improve model parameter plausibility. In T. Barnes, M. Desmarais, C. Romero, and S. Ventura, editors, *Proceedings of the 2nd International Conference on Educational Data Mining*, Cordoba, Spain, 2009.
9. M. Walker, C. Kamm, and D. Litman. Towards developing general models of usability with PARADISE. *Natural Language Engineering*, 6(3):363–377, 2001.

Authoring Tutors with Complex Solutions: A Comparative Analysis of Example Tracing and SimStudent

Christopher J. MacLellan[1], Erik Harpstead[1], Eliane Stampfer Wiese[1], Mengfan Zou[2], Noboru Matsuda[1], Vincent Aleven[1], and Kenneth R. Koedinger[1]

[1] Carnegie Mellon University, Pittsburgh PA, USA,
{cmaclell, eharpste, stampfer, noboru.matsuda, aleven, koedinger}@cs.cmu.edu,
[2] Tsinghua University, Beijing, China,
zmf11@mails.tsinghua.edu.cn

Abstract. Problems with many solutions and solution paths are on the frontier of what non-programmers can author with existing tutor authoring tools. Popular approaches such as Example Tracing, which allow authors to build tutors by demonstrating steps directly in the tutor interface. This approach encounters difficulties for problems with more complex solution spaces because the author needs to demonstrate a large number of actions. By using SimStudent, a simulated learner, it is possible to induce general rules from author demonstrations and feedback, enabling efficient support for complexity. In this paper, we present a framework for understanding solution space complexity and analyze the abilities of Example Tracing and SimStudent for authoring problems in an experimental design tutor. We found that both non-programming approaches support authoring of this complex problem. The SimStudent approach is 90% more efficient than Example Tracing, but requires special attention to ensure model completeness. Example Tracing, on the other hand, requires more demonstrations, but reliably arrives at a complete model. In general, Example Tracing's simplicity makes it good for a wide range problems, a reason for why it is currently the most widely used authoring approach. However, SimStudent's improved efficiency makes it a promising non-programmer approach, especially when solution spaces become more complex. Finally, this work demonstrates how simulated learners can be used to efficiently author models for tutoring systems.

Keywords: Tutor Authoring, Intelligent Tutoring Systems, Cognitive Modeling, Programming-by-Demonstration

1 Introduction

Intelligent Tutoring Systems (ITSs) are effective at improving student learning across many domains– from mathematics to experimental design [10, 13, 5]. ITSs

also employ a variety of pedagogical approaches for learning by doing, including intelligent novice [7], invention [12], and learning by teaching [9]. Many of these approaches require systems that can model complex solution spaces that accommodate multiple correct solutions to a problem and/or multiple possible paths to each solution. Further, modeling complex spaces can be desirable pedagogically: student errors during problem solving can provide valuable learning opportunities, and therefore may be desirable behaviors. Mathan and Koedingers spreadsheet tutor provides experimental support for this view– a tutor that allowed exploration of incorrect solutions led to better learning compared to one that enforced a narrower, more efficient solution path [7]. However, building tutoring systems for complex solution spaces has generally required programming. What options are available to the non-programmer? Authoring tools have radically reduced the difficulties and costs of tutor building [2, 6], and have allowed authoring without programming. Through the demonstration of examples directly in the tutor interface, an author can designate multiple correct solutions, and many correct paths to each solution. Yet, the capabilities of these tools for authoring problems with complex solution spaces has never been systematically analyzed.

In this paper, we define the concept of solution space complexity and, through a case study, explore how two authoring approaches deal with this complexity. Both approaches (Example Tracing and SimStudent) are part of the Cognitive Tutor Authoring Tools (CTAT) [1]. Our case study uses the domain of introductory experimental design, as problems in this area follow simple constraints (only vary one thing at a time), but solutions can be arbitrarily complex depending on how many variables are in the experiment and how many values each can take.

2 Solution Space Complexity

Solution spaces have varying degrees of complexity. Our framework for examining complexity considers both how many correct solutions satisfy a problem and how many paths lead to each solution. Within this formulation, we discuss how easily a non-programmer can author tutors that support many solutions and/or many paths to a solution.

How might this formulation of complexity apply to an experimental design tutor? Introductory problems in this domain teach the control of variables strategy (only manipulating a single variable between experimental conditions to allow for causal attribution) [3]. Due to the combinatorial nature of experiments (i.e., multiple conditions, variables, and variable values), the degree of complexity in a particular problem depends on how it is presented. To illustrate, imagine that students are asked to design an experiment to determine how increasing the heat of a burner affects the melting rate of ice in a pot (see Figure 1). The following tutor prompts (alternatives to the prompt in Figure 1) highlight how different problem framings will affect the solution complexity:

One solution with one path Design an experiment to determine how increasing the heat of a Bunsen burner affects the rate at which ice in a pot will

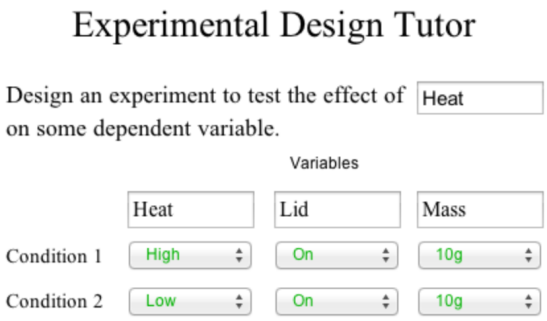

Fig. 1. Experimental design tutor interface

melt by assigning the first legal value to the variables in left to right, top down order as they appear in the table.

One solution and many paths Design an experiment to determine how increasing the heat of a Bunsen burner affects the rate at which ice in a pot will melt by assigning the first legal value to variables.

Many solutions each with one path Design an experiment to determine how increasing the heat of a Bunsen burner affects the rate at which ice in a pot will melt by assigning values to variables in left to right, top down order as they appear in the table.

Many solutions with many paths Design an experiment to determine how increasing the heat of a Bunsen burner affects the rate at which ice in a pot will melt.

While these examples show that solution space complexity can be qualitatively changed (i.e., one solution vs. many solutions) by reframing a problem, quantitative changes are also possible. For example, adding a fourth variable to the interface in Figure 1 would require two more steps per solution path (setting the variable for each condition), while adding another value to each variable increases the number of possible options at each step of the solution path. As this example illustrates, solution space complexity is not an inherent property of a domain, but rather arises from an authors design choices.

3 Tutor Authoring

Our analysis focuses on the Cognitive Tutor Authoring Tools (CTAT), as CTAT is the most widely used tutor authoring tool and the approaches it supports are representative of authoring tools in general [2]. CTAT supports non-programmers in building both tutor interfaces and cognitive models (for providing feedback). Cognitive models can be constructed with Example Tracing or SimStudent. In this section, we step through how Example-Tracing and SimStudent approaches would be applied by non-programmers to the experimental design task, using the

interface shown in Figure 1. Further, we discuss the features of each approach for handling solution space complexity in the context of this example.

3.1 Example Tracing

When building an Example-Tracing tutor in CTAT, the author demonstrates correct solutions directly in the tutoring interface. These demonstrated steps are recorded in a behavior graph. Each node in the behavior graph represents a state of the tutoring interface, and each link represents an action that moves the student from one node to another. In Example Tracing each link is produced as a result of a single action demonstrated directly in the tutor interface; many legal actions might be demonstrated for each state, creating branches in the behavior graph.

Figure 2 shows an example of our experimental design tutor interface and an associated behavior graph. The particular prompt chosen has 8 solutions and many paths to each solution. These alternative paths correspond to different orders in which the variables in the experimental design can be assigned. The Example-Tracing approach allows authors to specify that groups of actions can be executed in any order. In the context of our example, this functionality allows the author to demonstrate one path to each of the 8 unique solutions (these 8 paths are visible in Figure 2) and then specify that the actions along that path can be executed in any order. Unordered action groups are denoted in the behavior graph by colored ellipsoids.

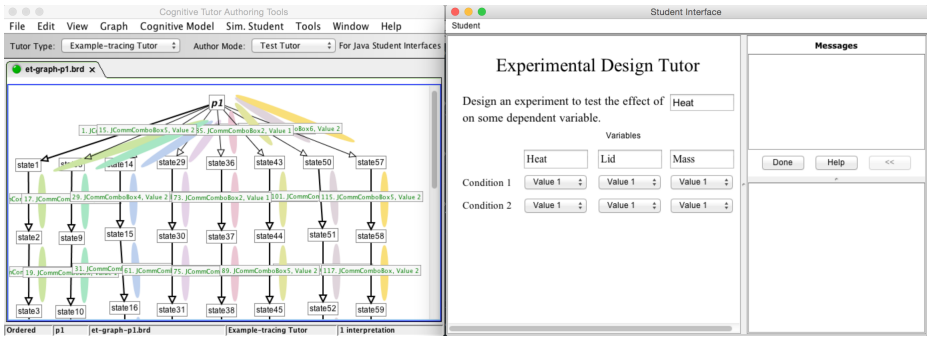

Fig. 2. An experimental design tutor (right) and its associated behavior graph (left). This tutor supports students in designing an experiment to test the effect of heat on a dependet variable. The correct answer is to pick two different values for the "Heat" variable and to hold the values constant for other variables.

Once a behavior graph has been constructed for a specific problem (e.g. determine the effect of heat on ice melting), that behavior graph can be generalized to other problems (e.g. determine the effect of sunlight on plant growth) using mass production. The mass production feature allows the author to replace specific values in the interface with variables and then to instantiate an arbitrary

number of behavior graphs with different values for the variables. This approach is powerful for supporting many different problems that have identical behavior graph structure, such as replacing all instances of "heat" with another variable, "sunlight". However, if a problem varies in the structure of its behavior graph, such as asking the student to manipulate a variable in the second column instead of the first (e.g., "lid" instead of "heat"), then a new behavior graph would need to be built to reflect the change in the column of interest.

How efficient is Example Tracing in building a complete cognitive model for the experimental design problem? The complete model consists of 3 behavior graphs (one for each of the three variable columns that could be manipulated). Each graph took 56 demonstrations and required 8 unordered action groups to be specified. Thus, the complete cognitive model required 168 demonstrations and 24 unordered group specifications. Using estimates from a previously developed Keystroke-Level Model [6], which approximates the time needed for an error-free expert to perform each interface action, we estimate that this model would take about 27 minutes to build using Example Tracing. Notably, the ability to specify unordered action groups offers substantial efficiency gains - without it, authoring would take almost 100 hours. Furthermore, with mass production, this model can generalize to any set of authored variables.

3.2 SimStudent

While the Example-Tracing behavior graph creates links from user demonstrations, the SimStudent system extends these capabilities by inducing production rule models from demonstrations and feedback (for details on this rule induction see [8]). In the experimental design tutor, SimStudent might learn a rule that sets one of the variables to an arbitrary value when no values for that variable have been assigned. Then, it might learn different rules for setting a variables second value based on whether or not it is being manipulated.

Authoring with SimStudent is similar to Example Tracing in that SimStudent asks for demonstrations when it does not know how to proceed. However, when SimStudent already has an applicable rule, it fires the rule and shows the resulting action in the tutor interface. It then asks the author for feedback on that action. If the feedback is positive, SimStudent may refine the conditions of its production rules before continuing to solve the problem. If the feedback is negative, SimStudent will try firing a different rule. When SimStudent exhausts all of its applicable rules, it asks the author to demonstrate a correct action. Figure 3 shows how SimStudent asks for demonstrations and feedback. When authoring with SimStudent, the author does not have to specify rule order - as long as a rule's conditions are satisfied, it is applicable. Authoring with SimStudent produces both a behavior graph (of the demonstrations and actions SimStudent took in the interface) and a production rule model.

To evaluate the efficiency of the SimStudent approach we constructed a complete model for the experimental design tutor. It can be difficult to determine when a SimStudent model is correct and complete from the authoring interactions alone. In most cases the SimStudent model is evaluated with set of held-out

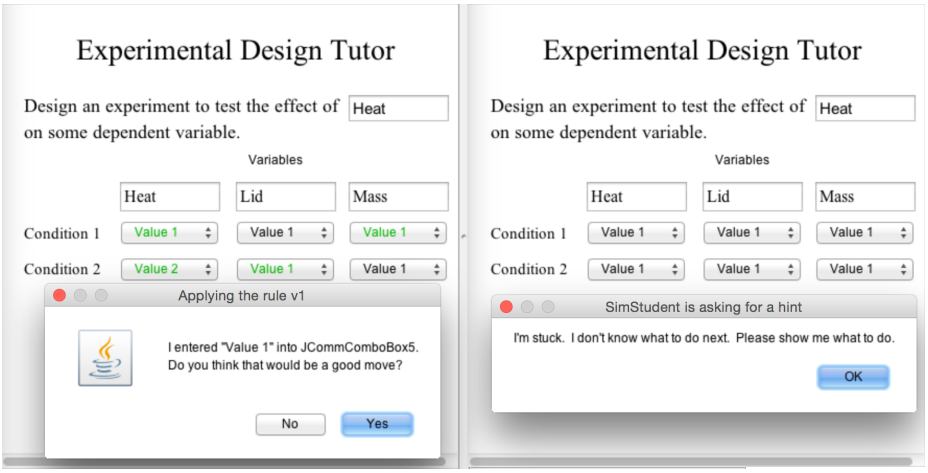

Fig. 3. SimStudent asking for feedback (left) and for a demonstration (right).

test problems (i.e., unit tests). However, in this case the learned rules were simple enough to evaluate by direct inspection. We noticed that SimStudent learned one correct strategy, but had not explored other solutions. This is typical of SimStudent - once it learns a particular strategy it applies it repeatedly. Therefore, authors must give it additional demonstrations of alternative paths. With the experimental design tutor, we noticed that SimStudent was always choosing the first value for non-manipulated variables, so we gave it additional demonstrations where non-manipulated variables took values besides those demonstrated on the initial run.

Ultimately, SimStudent acquired a complete model after 7 demonstrations and 23 feedback responses. Using the same Keystroke-Level Model from [6], we estimate that building a cognitive model using SimStudent would take an error-free expert about 2.12 minutes – much shorter than Example Tracing. Like Example Tracing, the model produced by SimStudent can work with arbitrary variables. Unlike Example Tracing, the learned model can work for unauthored variables; for example, students could define their own variables while using the tutor. This level of generality could be useful in inquiry-based learning environments [4]. Finally, if another variable column was added to the tutor, the SimStudent model would be able to function without modification. For Example Tracing, such a change would constitute a change to the behavior graph structure, so a completely new behavior graphs would need to be authored to support this addition.

4 Discussion

Both Example Tracing and SimStudent can create tutors for problems with complex solution spaces. However, our analysis shows that the two approaches

differ in terms of their efficiency and, as a result, how many solutions and paths they can handle in practice.

First, the Example-Tracing approach worked very well, even though the experimental design problems have a combinatorial structure. In particular, unordered action groups and mass production drastically reduced the number of demonstrations needed to cover the solution space, 168 vs. 40,362. The simplicity of Example Tracing combined with the power afforded by these features is likely why Example Tracing is the most widely used authoring approach today [2].

The SimStudent approach was more efficient than Example Tracing (approx. 2.12 vs. 27 minutes), but this comparison requires several caveats. The machine learning mechanisms of SimStudent generalize demonstrations and feedback into rules, which allows SimStudent to only model unique actions and the conditions under which they apply. However, this means SimStudent may not acquire a complete model. In the experimental design case study, SimStudent at first only learned that non-manipulated variables take their first value (rather than any value that is constant across conditions). In general, this problem arises when SimStudent acquires a model that can provide at least one correct solution for any problem. In these situations, it never prompts an author to provide alternative demonstrations; leading an unsuspecting author to create an incomplete model. A related complication is determining when the SimStudent model is complete. While determining the completeness of models in both Example Tracing and SimStudent can be difficult, authors must attempt to infer completeness from SimStudent's problem solving performance– a method that can be rather opaque at times. Thus, an open area for simulated learning systems is how best to evaluate the quality of learned models.

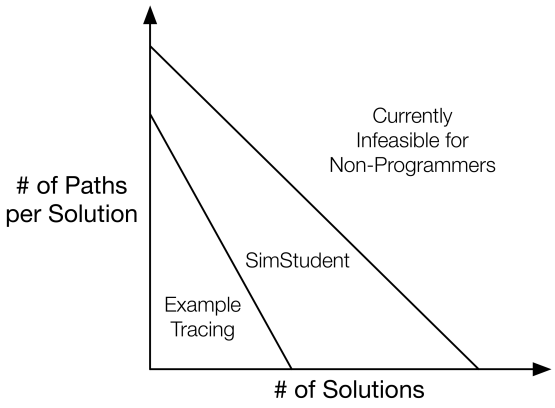

Fig. 4. How the space of solution space complexity is handled by existing non-programmer authoring approaches.

Overall our findings, when paired with those of previous work [6], suggest an interpretation depicted in Figure 4. In this figure the potential space of complexity is depicted in terms of number of unique solutions and number of paths per solution. The inner region denotes the area of the complexity space where we believe Example Tracing will maximize non-programmers' authoring utility. This region is skewed towards a higher number of paths, owing to Example Tracing's capacity to specify unordered actions. This portion of the complexity space contains many of the tutors that have already been built using Example Tracing [2]. As the complexity of a problem's solution space increases, Example Tracing becomes less practical (though still capable) and SimStudent becomes a more promising option, despite the caveats for using it. SimStudent's power of rule generalization gives it the ability to deal with more paths and unique solutions with less author effort, however, these capabilities come with the risk of producing incomplete models (without the author being aware).

Notably missing in the figure is any coverage of the upper right quadrant. This area would be a fruitful place to direct future work that supports non-programmers in authoring problems with many solutions with many paths. In particular, simulated learning systems might be extended to give non-programmers access to this portion of the space. One existing approach for dealing with highly complex solution spaces is to only model the aspects of the space that students are most likely to traverse. For example, work by Rivers and Koedinger [11] has explored the use of prior student solutions to seed a feedback model for introductory programming tasks. As it stands this area can only be reached using custom built approaches and would benefit from authoring tool research.

One limitation of our current approach is the assumption that there is a body of non-programmers that wants to build tutors for more complex problems. Our analysis here suggests that there is an open space for non-programming tools that support highly complex solution spaces, but it is less clear that authors have a desire to create tutors in this portion of the space. A survey of authors interested in building complex tutors without programming would help to shed light on what issues non-programmers are currently having in building their tutors. It is important that such a survey also include the perspective of those outside the normal ITS community to see if there are features preventing those who are interested from entering the space.

From a pedagogical point of view, it is unclear how much of the solution space needs to be modeled in a tutor. Waalkens et al. [16] have explored this topic by implementing three versions of an Algebra equation solving tutor, each with progressively more freedom in the number of paths that students can take to a correct solution. They found that the amount of freedom did not have an effect on students learning outcomes. However, there is evidence that the ability to use and decide between different strategies (i.e. solution paths) is linked with improved learning [14]. Further, subsequent work [15] has suggested that students only exhibit strategic variety if they are given problems that favor different strategies. Regardless of whether modeling the entire solution space is

pedagogically necessary, it is important that available tools support the ability to model complex spaces so that these research questions can be further explored.

5 Conclusion

The results of our analysis suggest that both the Example Tracing and SimStudent authoring approaches are promising methods for non-programmers to create tutors even for problems with many solutions with many paths. More specifically, we found that SimStudent was more efficient for authoring a tutor for experimental design, but authoring with SimStudent had a number of caveats related to ensuring that the authored model was complete. In contrast, Example Tracing was simple to use and it was clear that the authored models were complete. Overall, our analysis shows that Example Tracing is good for a wide range of problems that non-programmers might want to build tutors for (supported by its extensive use in the community [2]). However, the SimStudent approach shows great promise as an efficient authoring approach, especially when the solution space becomes complex. In any case, more research is needed to expand the frontier of non-programmers' abilities to author tutors with complex solution spaces.

Finally, this work demonstrates the feasibility and power of utilizing a simulated learning system (i.e., SimStudent) to facilitate the tutor authoring process. In particular authoring tutors with SimStudent took only 10% of the time that it took to author a tutor with Example-Tracing, a non-simulated learner approach. Educational technologies with increasingly complex solution spaces are growing in popularity (e.g. educational games and open-ended learning environments), but current approaches do not support non-programmers in authoring tutors for these technologies. Our results show that simulated learning systems are a promising tool for supporting these non-programmers. However, more work is needed to improve our understanding of how simulated learners can contribute to the authoring process and how the models learned by these systems can be evaluated.

6 Acknowledgements

We would like to thank Caitlin Tenison for her thoughtful comments and feedback on earlier drafts. This work was supported in part by a Graduate Training Grant awarded to Carnegie Mellon University by the Department of Education (#R305B090023) and by the Pittsburgh Science of Learning Center, which is funded by the NSF (#SBE-0836012). This work was also supported in part by National Science Foundation Awards (#DRL-0910176 and #DRL-1252440) and the Institute of Education Sciences, U.S. Department of Education (#R305A090519). All opinions expressed in this article are those of the authors and do not necessarily reflect the position of the sponsoring agency.

References

1. Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In: Ikeda, M., Ashley, K.D., Tak-Wai, C. (eds.) ITS '06. pp. 61–70. Springer (2006)
2. Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors. IJAIED 19(2), 105–154 (2009)
3. Chen, Z., Klahr, D.: All Other Things Being Equal: Acquisition and Transfer of the Control of Variables Strategy. Child Development 70(5), 1098–1120 (1999)
4. Gobert, J.D., Koedinger, K.R.: Using Model-Tracing to Conduct Performance Assessment of Students' Inquiry Skills within a Microworld. Society for Research on Educational Effectiveness (2011)
5. Klahr, D., Triona, L.M., Williams, C.: Hands on what? The relative effectiveness of physical versus virtual materials in an engineering design project by middle school children. Journal of Research in Science Teaching 44(1), 183–203 (Jan 2007)
6. MacLellan, C.J., Koedinger, K.R., Matsuda, N.: Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality. In: Trausen-Matu, S., Boyer, K. (eds.) ITS '14 (2014)
7. Mathan, S.A., Koedinger, K.R.: Fostering the Intelligent Novice: Learning From Errors With Metacognitive Tutoring. Educational Psychologist 40(4), 257–265 (2005), http://www.tandfonline.com/doi/abs/10.1207/s15326985ep4004_7
8. Matsuda, N., Cohen, W.W., Koedinger, K.R.: Teaching the Teacher: Tutoring SimStudent Leads to More Effective Cognitive Tutor Authoring. IJAIED 25(1), 1–34 (2014)
9. Matsuda, N., Yarzebinski, E., Keiser, V., Cohen, W.W., Koedinger, K.R.: Learning by Teaching SimStudent – An Initial Classroom Baseline Study Comparing with Cognitive Tutor. IJAIED (2011)
10. Pane, J.F., Griffin, B.A., McCaffrey, D.F., Karam, R.: Effectiveness of Cognitive Tutor Algebra I at Scale. Tech. rep., RAND Corporation, Santa Monica, CA (2013)
11. Rivers, K., Koedinger, K.R.: Automating Hint Generation with Solution Space Path Construction. In: ITS '14, pp. 329–339. Springer (2014)
12. Roll, I., Aleven, V., Koedinger, K.R.: The Invention Lab : Using a Hybrid of Model Tracing and Constraint-Based Modeling to Offer Intelligent Support in Inquiry Environments. In: ITS '10. pp. 115–124 (2010)
13. Sao Pedro, M.A., Gobert, J.D., Heffernan, N.T., Beck, J.E.: Comparing Pedagogical Approaches for Teaching the Control of Variables Strategy. In: Taatgen, N., van Rijn, H. (eds.) CogSci '09. pp. 1–6 (2009)
14. Schneider, M., Rittle-Johnson, B., Star, J.R.: Relations among conceptual knowledge, procedural knowledge, and procedural flexibility in two samples differing in prior knowledge. Developmental Psychology 47(6), 1525–1538 (2011)
15. Tenison, C., MacLellan, C.J.: Modeling Strategy Use in an Intelligent Tutoring System: Implications for Strategic Flexibility. In: ITS '14, pp. 466–475. Springer (2014)
16. Waalkens, M., Aleven, V., Taatgen, N.: Computers & Education. Computers & Education 60(1), 159–171 (Jan 2013)

Methods for Evaluating Simulated Learners: Examples from SimStudent

Kenneth R. Koedinger[1], Noboru Matsuda[1], Christopher J. MacLellan[1], and Elizabeth A. McLaughlin[1]

[1] Carnegie Mellon University, Pittsburgh, PA
koedinger@cmu.edu

Abstract. We discuss methods for evaluating simulated learners associated with four different scientific and practical goals for simulated learners. These purposes are to develop a precise theory of learning, to provide a formative test of alternative instructional approaches, to automate authoring of intelligent tutoring systems, and to use as a teachable agent for students to learn by teaching. For each purpose, we discuss methods for evaluating how well a simulated learner achieves that purpose. We use SimStudent, a simulated learner theory and software architecture, to illustrate these evaluation methods. We describe, for example, how SimStudent has been evaluated as a theory of student learning by comparing, across four domains, the cognitive models it learns to the hand-authored models. The SimStudent-acquired models yield more accurate predictions of student data in the three of the four domains. We suggest future research into more directly evaluating simulated learner predictions of the process of student learning.

Keywords: simulated learners, cognitive models, learning theory, instructional theory

1 Introduction

When is a simulated learner a success? We discuss different approaches to evaluating simulated learners. Some of these evaluation approaches are technical in nature, whether or how well a technical goal has been achieved, and some are empirical, whereby predictions from the simulated learner are compared against data. These approaches can be framed within the different goals and uses for simulated learners. Table 1 summarizes four purposes for developing simulated learners.

Table 1. Scientific and Practical Goals for Simulated Learners (SLs)

1. *Precise Theory*. Use SLs to develop and articulate precise theory of student learning in a replicable and unambiguous computational form.
   a. *Cognitive Model*. Create theories of domain expertise
   b. *Error Model*. Create theories of student domain misconceptions
   c. *Prior Knowledge*. Create theories of how different prior knowledge changes the nature and effectiveness of learning
   d. *Learning Process*. Create theories of change in student knowledge and performance
2. *Instructional Testing*. Use SLs as a "crash test" to evaluate and compare different instructional approaches on how well they facilitate learning.
3. *Automated Authoring*. Use SLs to automate the development of the expert component or cognitive model of an intelligent tutoring system.
4. *Teachable Agent*. Use SLs as a teachable agent or peer learner inside an instructional system to directly aid student learning.

Some of these goals have been pursued in prior simulated learner research. For example, [1] proposed the use of a simulated learner for instructional testing (#2 in Table 1). More specifically, he used "pseudo-students" during the design process for a formative evaluation of instruction to detect design defects.

Different evaluation approaches are appropriate for the different goals indicated in Table 1. In later sections, we discuss these evaluation approaches for each of the four main goals. But first, we introduce, SimStudent, the simulated learner system we have developed.

1.1 SimStudent: A Simulated Learner Theory and Software Architecture

We use SimStudent [2,3] as a running example to illustrate the evaluation techniques we discuss. SimStudent is a simulated learner system and theory in the class of adaptive production systems as defined by [4]. As such, it is similar to cognitive architectures such as ACT-R [5], Soar [6], and Icarus [7]. It is distinctive in its focus on modeling learning of complex academic topics, such as math, science, and language learning, and in its focus on inductive knowledge level learning [8]. SimStudent learns from a few primary forms of instruction, including examples of correct actions, skill labels on similar actions (which cue, but do not guarantee, learning and use of the same production rule), clues for what information in the interface to focus on to infer a next action, and finally yes-or-no feedback on actions performed by SimStudent.

To tutor SimStudent, the author first enters a problem in the tutoring interface (e.g., the "2x = 8" in the first row of Figure 1). SimStudent then attempts to solve the problem by applying productions learned so far. If an applicable production is found, the production application is visualized as a step in the behavior recorder represented as a new state-edge pair like the one shown in the bottom left of Figure 1. The author then provides correctness *feedback* on the step performed by SimStudent. When there are multiple applicable productions SimStudent shows the author all corresponding production applications and obtains correctness feedback on each.

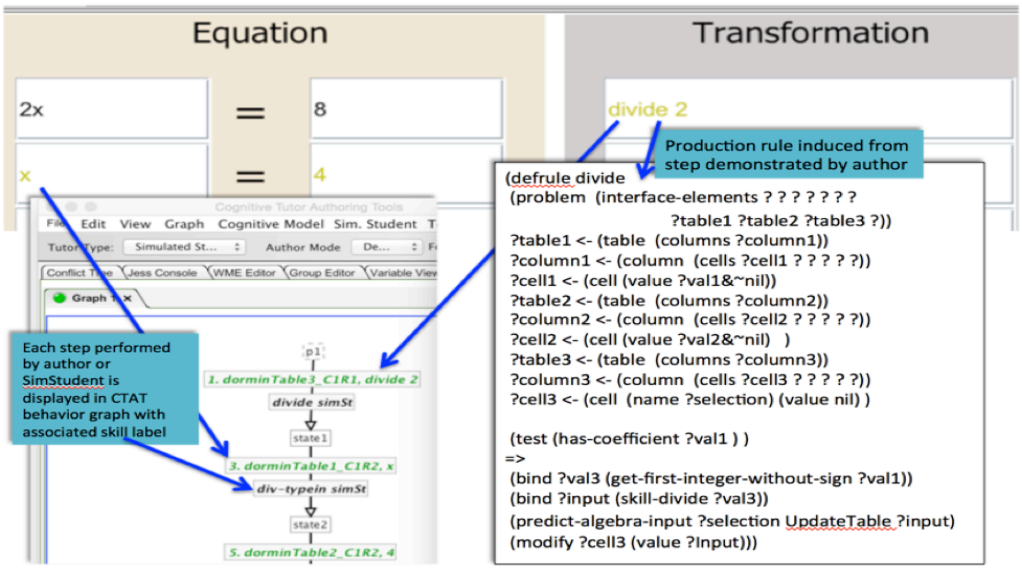

Fig. 1. After creating an interface (shown at top) and entering a problem ("2x=8"), teaching of SimStudent occurs either by giving yes-or-no feedback when SimStudent attempts a step or by demonstrating a correct step when SimStudent cannot (e.g., "divide 2"). SimStudent induces production rules from demonstrations (example shown on right) for each skill label (e.g., "divide" or "div-typein" shown on left). It refines productions based on subsequent positive (demo or yes feedback) or negative (no feedback) examples.

If no correct production application is found, then SimStudent asks the author to demonstrate the next step directly in the interface. When providing a demonstration, the author first specifies the *focus of attention* (i.e. input fields relevant to the current step) by double-clicking the corresponding interface elements, for example, the cells containing "2x" and "8" in Figure 1. The author then takes action using the relevant information (e.g., entering "divide 2" in Figure 1). The demonstrated step is visualized in the behavior graph. Finally, the *author specifies the skill name* by clicking on the newly added edge of the behavior graph. A small dialogue box appears to enter a skill name. This skill label is used to guide SimStudent's learning and to make the models acquired by SimStudent more interpretable (i.e., to give them human read-

able names). Generally speaking, authors could model a single observable step (i.e., an edge in the behavior graph) with a chain of production-rule applications. However, when using SimStudent to author an expert model, SimStudent generates a single production rule per each observable step.

SimStudent learns production rules from the author's demonstrations and feedback using three machine-learning mechanisms: *how*, *where*, and *when* learning. When given a new demonstration (i.e., a positive example of a rule), SimStudent uses its *how* learner to explain the demonstration and produces a general composition of functions that replicate the demonstrated steps and ones like it. For example, in Figure 1, when given the demonstration "divide 2" for the problem 2x=8, SimStudent induces (or "guesses") that the result of the "get-first-integer-without-sign" function when applied to left side of the problem and appended to the word "divide" explains the demonstration. This general sequence can then be used on novel problems (e.g., 4x=12 -> divide 4).

After an action sequence has been discovered, SimStudent uses its *where* learner to identify a generalized path to the focuses of attention in the tutor interface; e.g., to the left and right sides of the equation or to the next step input field. For the example in Figure 1, the *where* learner discovers retrieval paths for the three cells in the first column. These paths are generalized as more positive examples are acquired for a given rule. For example, when the author demonstrates the application of the divide rule shown in Figure 1 to the second row of the equation table, then the production's retrieval path might be generalized to function over any row in the equation table (rather than just the first two rows).

Finally, after learning an action sequence and general paths to relevant information, SimStudent uses its *when* learning to identify the conditions under which the learned production rule produces correct actions. For the example in Figure 1, SimStudent learns that this rule can only be correctly applied when one side of the equation has a coefficient. In situations when SimStudent receives positive and negative feedback on its rule applications, it uses the *when* learner to update the conditions on the rules. Thus, the *how* and *where* learners primarily use positive examples and the *when* learner uses both positive and negative examples.

SimStudent is also capable of learning the representation of the chunks (object attribute structures) that make up the production system working memory and are the informational basis on which productions are learned. It does so using an unsupervised grammar induction approach [3]. This feature particularly sets it apart from the other production rule learning systems mentioned above.

2 Evaluating Simulated Learners as Theories of Learning

It is helpful to distinguish a general theory of learning from a human-specific theory of *student* learning. We focus primarily on a theory of human student learning as it is most relevant to the education goals of the field of AI in Education. However, it is worth mentioning that there are evaluation criteria for a general learning theory, such as how quickly (e.g., in number of examples or time) and independently (e.g., with less supervision) learning takes and how general and accurate is the performance (e.g., problem solving or inference capability) of the resulting expert system. These criteria, then, provide guidance for comparative evaluations of general theories of learning. It is reasonable to consider as a better theory of learning one that produces improvements over a competing theory on any of speed, independence, generality, or accuracy without harming any of the others. In [9], for instance, Tenenbaum, Griffiths and Kemp have argued that hierarchical Bayesian models are better models of learning than other classification or neural network models because they can learn as well with fewer examples. (Note: If one suggests that a model is better because humans are able to learn with fewer examples, then one is moving into the realm of a human-specific theory of learning.)

While not necessary to evaluate general theories of learning, to evaluate the validity of theories of student learning, it is critical that the simulated learner be compared with student data. This data may involve student correct performance, incorrect performance, performance across tasks, and changes in performance over time. The data may be qualitative or quantitative.

2.1 Good Student Learning Theory Should Generate Accurate Cognitive Models

A student learning theory should produce the kind of expertise that human students acquire. In other words, the result of teaching a simulated learner should be a cognitive model of what a human student

knows after instruction and should behave as a human student does after instruction. For this kind of evaluation of a simulated learner, the issue reduces to the question of how to evaluate a cognitive model. In [10], we proposed six constraints to evaluate the quality of a cognitive model. These were labeled 1) solution sufficiency, 2) step sufficiency, 3) choice matching, 4) computational parsimony, 5) acquirability, and 6) transfer. The first two are empirical and qualitative: Is the cognitive model that the SL acquires able to solve tasks in the domain of interest and does it do so with steps that are consistent with human students? The third is quantitative: Does the frequency of strategy use and common error categories generated by the cognitive model on different tasks correspond with the frequency of strategy use and common error categories exhibited by human students on those tasks? The last three are rational in character, involving inspection of the cognitive model (or contrasting models) to judge whether it is (they are) not unnecessarily complex (#4), can be plausibly learned (#5), and implies transfer through overlap in knowledge components that apply across tasks (#6).

These constraints were designed to evaluate cognitive models developed by hand (e.g., by a scientist writing an expert system), but in the case that the model is generated by a simulated learner, the acquirability constraint (#5) is naturally achieved. The components of the cognitive models can be plausible because the SL does, in fact, learn them. If the cognitive model that is produced can solve tasks in the domain, for example, a simulated learner trained on algebra equations can solve equations, then the solution sufficiency constraint (#1) is met. If it solves them using the kinds of intermediate steps that match the kinds of steps in student solutions, for example, it performs its solution in a step-based tutoring system interface, then the step sufficient constraint (#2) is met.

How, then, can the remaining choice matching (#3), parsimony (#4), and transfer (#6) constraints be evaluated? In [11], we employed an approach that provides much of what is needed. This approach employs educational data mining and, in particular, evaluates the accuracy of a cognitive model by the so-called "smooth learning curve" criteria [cf., 12,13]. Using a relatively simple statistical model of how instructional opportunities improve the accuracy of knowledge, this approach can measure and compare cognitive models in terms of their accuracy in predicting learning curve data. To employ the statistical model fit, the cognitive model is simplified into a "Q matrix", which encodes a mapping from each observed task students perform (e.g., answering a question or entering a step in a problem solving) to the knowledge components that are hypothesized to be needed to successfully perform that task. Different cognitive models produce different Q matrices and different levels of predictive accuracy in fitting student learning curve data (measured, for example, by the root mean squared error on held-out data in cross validation). For any appropriate dataset uploaded into DataShop (learnlab.org/DataShop), the website allows users to edit and upload alternative cognitive models (in the Q matrix format), automatically performs statistical model fits, renders learning curve visualizations, and displays a ranking ordering of the models in terms of their predictive accuracy [14].

In [11], this approach was used to evaluate the empirical accuracy of the cognitive models that SimStudent learns as compared to hand-authored cognitive models. SimStudent was tutored in four domains: algebra, fractions, chemistry, and English grammar, in which we had existing human data and existing hand-authored cognitive models (see Figure 2). In each domain SimStudent induced, from examples and from practice with feedback, both new chunk structures to represent the organization (or "grammar") of the perceptual input in each domain and new production rules that solve problems (e.g., add two fractions) or make decisions (e.g., select when to use "the" or "a" in English sentences) in each domain. In each case, the production rules that SimStudent acquired were converted into the Q matrix format whereby a production (the columns of the Q matrix) is indicated as needed (entering a 1 rather than a 0 in the matrix) for a task (the rows of the Q matrix) if SimStudent uses that production to succeed on that task. Then the DataShop cognitive model comparison was employed to compare whether these models fit student learning curve data better than the hand-authored cognitive models do.

In all four domains, the SimStudent-acquired cognitive models made distinctions not present in the hand-authored models (e.g., it had two different production rules across tasks for which the hand-authored model had one) and thus it tended to produce models with more knowledge components (as shown in Table 2). For example, SimStudent learned two different production rules for the typical last step in equation solving where one production covered typical cases (e.g., from $3x = 12$ the student should "divide by 3") and another covered a perceptually distinct special case (e.g., from $-x = 12$ the student should divide by -1).

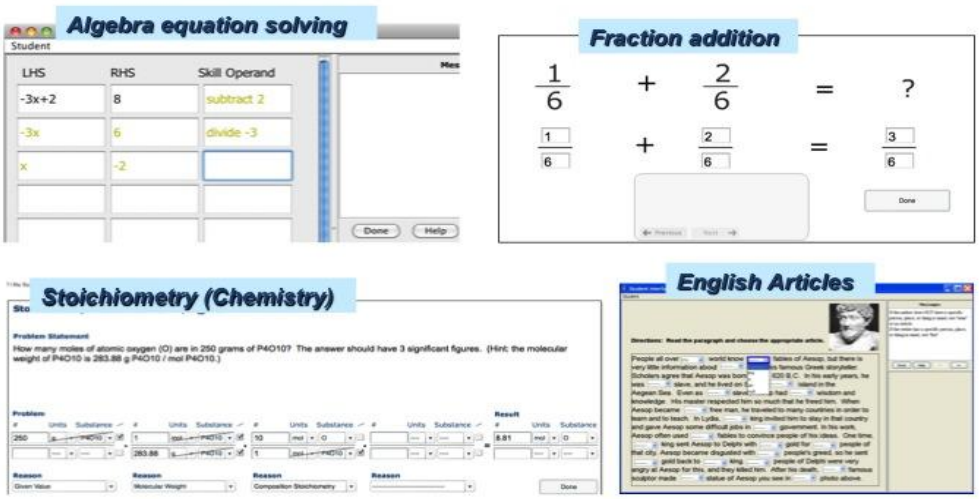

Fig. 2. Four domains in which SimStudent has been tutored and for which we have student learning data.

In all four domains, at least some of these distinctions improved the accuracy of fit to the learning curve data for the relevant tasks. Continuing the example, the SimStudent-acquired cognitive model in algebra leads to better accuracy because real students had a much higher error rate on tasks like -x=12 (where the coefficient, -1, is implicit) than on tasks like 3x=12 (where the coefficient, 3, is explicitly visible). More generally, in one domain (Fraction Addition, see Table 2), the SimStudent-acquired cognitive model failed to make a key distinction present in the hand-authored model and thus, while better in some cases, its overall fit was worse. In the three other domains (see Table 2), the SimStudent-acquired cognitive models were all found to be more accurate than the hand-authored cognitive models.

Table 2. A comparison of human-generated and SimStudent-discovered models. The columns on the left show the number of productions rules in the cognitive models and the columns on the right show the root mean squared error of the models for predicting held-out student learning data in cross validation.

	Number of Production Rules		Cross-Validated RMSE	
	Human-Generated Model	SimStudent Discovered Model	Human-Generated Model	SimStudent Discovered Model
Algebra	12	21	0.4024	**0.3999**
Stoichiometry	44	46	0.3501	**0.3488**
Fraction Addition	8	6	**0.3232**	0.3343
Article selection	19	22	0.4044	**0.4033**

In other words, this "smooth learning curve" method of evaluation can provide evidence that a simulated learner, SimStudent in this case, is a reasonable model of student learning in that it acquires knowledge at a grain size (as represented in the components of the cognitive model) that is demonstrably consistent with human data.

One limitation of this approach is that it *indirectly* compares a simulated learner to human learners through the process of fitting a statistical model. In the case of algebra, for example, SimStudent's acquisition of two different productions for tasks of the form Nx=N versus tasks of the form -x=N gets translated into a prediction that student performance will be *different* in these situations, but the not direction of the difference. It is process of estimating the parameters of the statistical model that yields the prediction for which of these task categories (Nx=N or -x=N) will be harder. A more *direct* comparison would not use an intermediate statistical model fit. It would require the simulated learner to not only produce a relevant distinction, but to make a prediction of student performance differences, such as whether it takes longer to successfully learn some kinds of tasks than others. Such an evaluation approach is discussed in section 2.3.

2.2 Good Student Learning Theory Should Generate Errors that Match Student Errors and Test Prior Knowledge Assumptions

As a model of student learning, a good simulated learner should not only produce accurate performance with learning, but should also produce the kinds of errors that students produce [cf.,15]. Thus, another way to evaluate a simulated learner is compare the errors it generates with student errors.

One theory of student errors is that students learn incorrect knowledge (e.g., incorrect production rules or schemas) from *correct example-based instruction* due to the necessary fallibility of inductive learning processes. A further hypothesis is that inductive learning errors are more likely when students have "weak" (i.e., more domain general) rather than "strong" (i.e., more domain specific) prior knowledge. With weak prior knowledge, students may interpret examples shallowly, paying attention to more immediately perceived surface features, rather than more deeply, by making domain-relevant inferences from those surface features. Consider example-based instruction where a student is given the equation "3x+5 = 7" and told that "subtract 5" from both sides is a good next step. A novice student with weak prior knowledge might interpret this example shallowly, as subtracting a number (i.e., 5) instead of more deeply, as subtracting a term (i.e., +5). As a consequence, the student may induce knowledge that produces an error on a subsequent problem, such as "4x-2=5" where they subtract 2 from both sides. Indeed, this error is a common one among beginning algebra students.

In [16], we evaluated SimStudent by comparing induction errors it makes with human student errors. More specifically, we evaluated the weak prior knowledge hypothesis expressed above. We conducted a simulation study by having multiple instances of SimStudent get trained by Cognitive Tutor Algebra I. We compared SimStudent behaviors with actual student data from the Cognitive Tutor's logs of student interactions with the system. When SimStudent starts with weak prior knowledge rather than strong prior knowledge, we found that it learns more slowly, that is, the accuracy of learned skills is lower given the same amount of training or the amount of training needed to reach the same level of accuracy is greater. More importantly, we found that SimStudent's ability to predict student errors increased significantly when given weak rather than strong prior knowledge. In fact, the errors generated by SimStudent with strong prior knowledge were almost never the same kinds of errors commonly made by real students.

In addition to illustrating how a simulated learner can be evaluated by comparing its error generation to human errors, the above example illustrates how a simulated learner can be used to test assumptions about what prior knowledge students bring to their learning environments. The study above showed that novice algebra students do not have strong prior knowledge, particularly of the grammatical elements of equations, such as terms and coefficients. Thus, the SimStudent provides a theoretical explanation not only of common student error patterns, but also of empirical results (e.g., Booth and Koedinger, 2008) showing correlations between tasks measuring prior knowledge (e.g., identify the negative terms in "3x - 4 = -5 - 2x") and subsequent learning of target skills (e.g., solving algebra equations).

Some previous studies of students' errors focus primarily on a descriptive theory to explain why students made particular errors, for example, repair theory [15], the theory of bugs [18], and the theory of extrapolation technique [19]. With simulated learners [cf., 15], we can better understand the process of acquiring the incorrect skills that generate errors. The precise understanding that computational modeling of learning facilitates provides us with insights into designing better learning environments that anticipate and prevent or quickly remedy error formation.

2.3 Good Student Learning Theory Should Match Learning Process Data

Matching a simulated learners performance to learning process data is similar to the cognitive model evaluation discussed above in section 2.1. However, as indicated above, that approach has the limitation of being an indirect comparison with human data whereby there the fit to human data is, in a key sense, less challenging because it is mediated by a separate step parameter estimation of a statistical model. A more direct comparison is, in simple terms, to match the behavior of multiple instances of a simulated learner (i.e., a whole simulated class) with the behavior of multiple students. The simulated learners interact with a tutoring system (like one shown in Figure 2) just as a class of human students would and their behavior is logged just as human student data is. Then the simulated and human student data logs can be compared, for example, by comparing learning curves that average across all (simulated and human) student participants.

3 Evaluating Simulated Learners as Instruction Testers

A number of projects have explored the use of a simulated learner to compare different forms of instruction. VanLehn was perhaps the first to suggest such a use of a "pseudo student" [1]. MacLaren & Koedinger used a version of ACT-R's utility learning mechanism to show that the simulated learner was often successful when given error feedback not only on target performance tasks (e.g., solving two-step equations), but also on shorter subtasks (e.g., one-step equations) [10]. Matsuda, Cohen & Koedinger used SimStudent to show better learning from giving it a combination of examples and problems to solve, than just giving it examples [2]. Li, Cohen & Koedinger showed that interleaving problem types is as good, or better, for learning than blocking problem types because interleaving provides better opportunities for detecting and correcting generalization errors [20].

Recall the distinction mentioned above between general learning theory and human learning theory. This distinction can be extended to separate a general theory of instruction from a theory of instruction relevant to human students. For a general theory of instruction, it is of scientific interest to understand the effectiveness of different forms of instruction for different kinds of SL systems even if the SL is not (or not known to be) an accurate model of student learning. Such understanding may be relevant to advancing applications of AI and is directly relevant to the issue of how an SL can be easily trained for purposes of automated ITS authoring, the topic of the next section. Such theoretical demonstrations may also have relevance to a theory of *human* instruction as they may 1) provide theoretical explanations for instructional improvements that have been demonstrated with human learners or 2) generate predictions for what may (or may not) work (but has not yet been tried) with human students.

However, these instructional conclusions can only be reliably extended to human learners when there is existing evidence that the simulated learner is an accurate model of student learning (see the prior section 2). And ideally, the most reliable evaluation of a simulated learner as instructional tester is a follow-up random assignment experiment with human learners that demonstrates that the instructional form that was better for the simulated learners is also better for students. In the examples given above, there is some reasonable evidence that the simulated learners (or learning mechanisms) applied are accurate models of student learning. In many cases, there are past relevant human experiments. However, in none of these cases was the ideal follow-up experiment performed.

4 Evaluating Simulated Learners as ITS Authoring Tools

In addition to their use as theories of learning and for testing instructional content, simulated learning systems can also be used to facilitate the authoring of Intelligent Tutoring Systems (ITS). In particular, once a simulated learner has been sufficiently trained, the cognitive model it learns can then be used directly as an expert model. Previous work, such as Example Tracing tutor authoring [21], has explored how models can be acquired by demonstration. However, by using a simulated learning system to induce general rules form the demonstrations more general models can be acquired more efficiently. For example, the use of SimStudent as authoring tool is still experimental, but there is evidence that it may accelerate the authoring process and that it may produce more accurate cognitive models than hand authoring. In one demonstration, [2] explored the benefits of a traditional programming by demonstration approach to authoring in SimStudent versus a programming by tutoring approach, whereby SimStudent asks for demonstrations only at steps in a problem/activity where it has no relevant productions and otherwise it performs a step (firing a relevant production) and asks the author for feedback as to whether the step is correct/desirable or not. They found that programming by tutoring is much faster, 13 productions learned with 20 problems in 77 minutes versus 238 minutes in programming by demonstration. They also found that programming by tutoring produced a more accurate cognitive model whereby there were fewer productions that produced over-generalization errors. Programming by tutoring is now the standard approach used in SimStudent and its improved efficiency and effectiveness over programming by demonstration follow from having SimStudent start performing its own demonstrations. Better efficiency is obtained because the author need only respond to each of SimStudent's step demonstrations with a single click, on a yes or no button, which is much faster than demonstrating that step. Better effectiveness is obtained because these demonstrations better expose over-

generalization errors to which the author responds "no" and the system learns new IF-part preconditions to more appropriately narrow the generality of the modified production rule.

In a second demonstration of SimStudent as an authoring tool, [22] compared authoring in SimStudent (by tutoring) with authoring example-tracing tutors in CTAT. Tutoring SimStudent has considerable similarity with creating an example-tracing tutor except that SimStudent starts to perform actions for the author, which can be merely checked as desirable or not, saving the time it otherwise takes for an author to perform those demonstrations. That study reported a potential savings of 43% in authoring time by using SimStudent to aid in creating example-tracing tutors. As mentioned before, the work by [11] has shown that the models acquired using SimStudent better fit the student data. Thus, using the SimStudent system to author a tutoring system allows for the efficient construction of empirically better models.

5 Evaluating a Simulated Learner as a Teachable Agent

Simulated learner systems can be more directly involved in helping students learn when they are used as a teachable agent whereby students learn by teaching [cf., 23]. Evaluating the use of a simulated learner in this form ideally involves multiple steps. One should start with a simulated learner that has already received some positive evaluation as a good model of student learning (see section 2). Then incorporate it into a teachable agent architecture and, as early and often as possible, perform pilot students with individual students [cf., 24 on think aloud user studies) and revise the system design. Finally, for both formative and summative reasons, use random assignment experiments to compare student learning from the teachable agent with reasonable alternatives.

Using SimStudent, we built a teachable agent learning environment, called APLUS, in which students learn to solve linear equations by teaching SimStudent [25] . To evaluate the effectiveness of APLUS and advance the theory of learning by teaching, we conducted multiple *in vivo* experiments each with a specific hypotheses to test [25,26,27,28].

Each of the classroom studies have been randomized controlled trials with two conditions controlling a single study variable. For example, in one study [25], the self-explanation hypothesis was tested by having students justify their tutoring activities and decision making. To test this hypothesis, we developed a version of APLUS in which SimStudent occasionally asked "why" questions. For example, when a student provided negative feedback to a step SimStudent performed, SimStudent asked, "Why do you think adding 3 here on both sides is incorrect?" Students were asked to respond to SimStudent's questions either by selecting pre-specified menu items or entering a free text response. The results showed that the amount and the level of elaboration of the response had a reliable correlation with students' learning measured by online pre- and post-tests.

We also compared learning by teaching with other forms of instruction [28]. In this *in vivo* study, half of the students used APLUS and half used Cognitive Tutor Algebra I [29]. The results showed an aptitude-treatment interaction such that students scoring in the low half on the pre-test may not be ready to benefit from learning by teaching -- they learned better using the Cognitive Tutor than using APLUS -- whereas students scoring in the high half of the pre-test learned more from APLUS (i.e., by teaching) than from the Cognitive Tutor (i.e., by being tutored).

6 Conclusion

We outlined four general purposes for simulated learners (see Table 1) and reviewed methods of evaluation that align with these purposes. To evaluate a simulated learner as a precise theory of learning, one can evaluate the cognitive model that results from learning, evaluate the accuracy of error predictions as well as prior knowledge assumptions needed to produce those errors, or evaluate the learning process, that is, the opportunity by opportunity changes in student performance over time. To evaluate a simulated learner as an instructional test, one should not only evaluate the systems accuracy as a precise theory of student learning, but should also perform human experiment to determine whether the instruction that works best for simulated learners also works best for human students. To evaluate a simulated learner as an automated authoring tool, one can evaluate the speed and precision of rule production, the frequency of over-

generalization errors and the fit of the cognitive models it produces. More ambitiously, one can evaluate whether the resulting tutor produces as good (or better!) learning than an existing tutor. Similarly, to evaluate a simulated learner as a Teachable Agent, one can not only evaluate the features of a Teachable Agent system, but also perform experiments on whether students learn better with that system than with reasonable alternatives.

Simulated learner research is still in its infancy so most evaluation methods have not been frequently used. There have been very few studies that have evaluated a simulated learner as an instructional tester by following up a predicted difference in instruction with a random assignment experiment using the same forms of instruction with real students. We know of just one such study in which [29] first used an extension of the ACT-R theory of memory to simulate positive learning effects of an optimized practice schedule over a (highly recommended) spaced practice schedule. Next, he ran the same experiment with human students and confirmed the benefits of the optimized practice schedule. Such experiments are more feasible when the instruction involved is targeting simpler learning processes, such as memory, but will be more challenging as they target more complex learning processes, such as induction or sense making [cf., 31]

As far as we know, there have been no studies evaluating the learning process of a simulated learner as we recommended in section 2.3. Such evaluations would be particularly compelling demonstrations of the power of the simulated learner approach!

As we argued in [32], the space of instructional choices is just too large, over 200 trillion possible forms of instruction, for a purely empirical science of learning and instruction to succeed. We need parallel and coordinated advances in theories of learning *and* instruction and efforts to develop and evaluate simulated learners are fundamental to such advancement.

References

1. VanLehn, K. (1991). Two pseudo-students: Applications of machine learning to formative evaluation. In R. Lewis & S. Otsuki (Eds.), *Advanced Research on Computers in Education* (pp. 17-26). Amsterdam: Elsevier.
2. Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2015). Teaching the Teacher: Tutoring SimStudent leads to more Effective Cognitive Tutor Authoring. *International Journal of Artificial Intelligence in Education, 25*, 1-34.
3. Li, N., Matsuda, N., Cohen, W., & Koedinger, K.R. (2015). Integrating representation learning and skill learning in a human-like intelligent agent. *Artificial Intelligence, 219,* 67-91.
4. Anzai, Y. & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review, 86 (2)*, 124-140.
5. Anderson, J. R., & Lebiere, C. (1998). *The Atomic Components of Thought*. Hillsdale, NJ: Erlbaum.
6. Laird, J. E., Newell, A., & Rosenbloom, P.S. (1987). Soar: an architecture for general intelligence. *Artificial Intelligence, 33*(1), 1–64.
7. Langley, P. & Choi, D. (2006). A unified cognitive architecture for physical agents. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, Boston.
8. Newell, Allen. 1990. *Unified Theories of Cognition*. Cambridge, MA: Harvard U. Press.
9. Tenenbaum, J. B., Griffiths, T. L., & Kemp.C. (2006). Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, 10, 309-318.
10. MacLaren, B. & Koedinger, K. R. (2002). When and why does mastery learning work: Instructional experiments with ACT-R "SimStudents". In S.A. Cerri, G. Gouarderes, & F. Paraguacu (Eds.), *Proceedings of the 6th International Conference on Intelligent Tutoring Systems*, 355-366. Berlin: Springer-Verlag.
11. Li, N., Stampfer, E., Cohen, W., & Koedinger, K.R. (2013). General and efficient cognitive model discovery using a simulated student. In M. Knauff, N. Sebanz, M. Pauen, I. Wachsmuth (Eds.), *Proceedings of the 35th Annual Conference of the Cognitive Science Society*. (pp. 894-9) Austin, TX: Cognitive Science Society.
12. Martin, B., Mitrovic, T., Mathan, S., & Koedinger, K.R. (2011). Evaluating and improving adaptive educational systems with learning curves. *User Modeling and User- Adapted Interaction: The Journal of Personalization Research (UMUAI), 21*(3), 249-283. [2011 James Chen Annual Award for Best UMUAI Paper]
13. Stamper, J.C. & Koedinger, K.R. (2011). Human-machine student model discovery and improvement using data. In G. Biswas, S. Bull, J. Kay, & A. Mitrovic (Eds.), *Proceedings of the 15th International Conference on Artificial Intelligence in Education*, pp. 353-360. Berlin: Springer.
14. Koedinger, K.R., Baker, R.S.J.d., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J. (2010). A Data Repository for the EDM community: The PSLC DataShop. In Romero, C., Ventura, S., Pechenizkiy, M., Baker, R.S.J.d. (Eds.) *Handbook of Educational Data Mining*. Boca Raton, FL: CRC Press.

15. Brown, J. S., & VanLehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science, 4,* 379-426.
16. Matsuda, N., Lee, A., Cohen, W. W., and Koedinger, K. R. (2009). A computational model of how learner errors arise from weak prior knowledge. In *Proceedings of Conference of the Cognitive Science Society*. pp. 1288-1293. Amsterdam, The Netherlands.
17. Booth, J. L., & Koedinger, K. R. (2008). Key misconceptions in algebraic problem solving. In B. C. Love, K. McRae & V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (pp. 571-576). Austin, TX: Cognitive Science Society.
18. VanLehn, K. (1982). Bugs are not enough: Empirical studies of bugs, impasses and repairs in procedural skills. *Journal of Mathematical Behavior, 3(2)*, 3-71.
19. Matz, M. (1980). Towards a process model for high school algebra errors. In D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems* (pp. 25-50). Orlando, FL: Academic Press.
20. Li, N., Cohen, W. W., & Koedinger, K. R. (2012). Problem Order Implications for Learning Transfer (pp. 185–194). Presented at the *Proceedings of the Eleventh International Conference on Intelligent Tutoring Systems*, Springer Berlin Heidelberg. doi:10.1007/978-3-642-30950-2_24
21. Aleven, V., McLaren, B., Sewall, J., & Koedinger, K. R. (2009). Example-tracing tutors: A new paradigm for intelligent tutoring systems. *International Journal of Artificial Intelligence in Education, 19,* 105-154.
22. MacLellan, C.J., Koedinger, K.R., Matsuda, N. (2014) Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality. *Proceedings of the 12th International Conference on Intelligent Tutoring Systems,* 551-560. Honolulu, HI.
23. Biswas, G., Schwartz, D., Leelawong, K., Vye, N. (2005) Learning by Teaching: A New Agent Paradigm for Educational Software. *Applied Artificial Intelligence, 19*, 363-392.
24. Gomoll, K., (1990). Some techniques for observing users. In Laurel B. (ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley, Reading, MA, pp. 85-90.
25. Matsuda, N., Yarzebinski, E., Keiser, V., Raizada, R., William, W. C., Stylianides, G. J., & Koedinge, K. R. (2013). Cognitive anatomy of tutor learning: Lessons learned with SimStudent. *Journal of Educational Psychology, 105(4)*, 1152-1163. doi: 10.1037/a0031955
26. Matsuda, N., Cohen, W. W., Koedinger, K. R., Keiser, V., Raizada, R., Yarzebinski, E., Watson, S. P., & Stylianides, G. J. (2012). Studying the Effect of Tutor Learning using a Teachable Agent that asks the Student Tutor for Explanations. In M. Sugimoto, V. Aleven, Y. S. Chee & B. F. Manjon (Eds.), *Proceedings of the International Conference on Digital Game and Intelligent Toy Enhanced Learning* (DIGITEL 2012) (pp. 25-32). Los Alamitos, CA: IEEE Computer Society.
27. Matsuda, N., Griger, C. L., Barbalios, N., Stylianides, G., Cohen, W.W., & Koedinger, K. R. (2014). Investigating the Effect of Meta-Cognitive Scaffolding for Learning by Teaching. In S. Trausen-Matu, K. Boyer, M. Crosby & K. Panourgia (Eds.), *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 104-113). Switzerland: Springer.
28. Matsuda, N., Keiser, V., Raizada, R., Tu, A., Stylianides, G. J., Cohen, W. W., & Koedinger, K. R. (2010). Learning by Teaching SimStudent: Technical Accomplishments and an Initial Use with Students. In V. Aleven, J. Kay & J. Mostow (Eds.), *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 317-326). Heidelberg, Berlin: Springer.
29. Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. (2007). Cognitive tutor: Applied research in mathematics education. *Psychonomic Bulletin &Review, 14(2)*, 249-255.
30. Pavlik, P.I. & Anderson, J. R. (2008). Using a model to compute the optimal schedule of practice. *Journal of Experimental Psychology: Applied, 14*, 101-117.
31. Koedinger, K.R., Corbett, A.C., & Perfetti, C. (2012). The Knowledge-Learning-Instruction (KLI) framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science, 36 (5),* 757-798. ISSN: 0364-0213 print / 1551-6709 online DOI: 10.1111/j.1551-6709.2012.01245.x
32. Koedinger, K.R., Booth, J.L., & Klahr, D. (2013). Instructional complexity and the science to constrain it. *Science, 342*, 935-937.

Simulated learners in peers assessment for introductory programming courses

Alexandre de Andrade Barbosa[1,3] and Evandro de Barros Costa[2,3]

[1] Federal University of Alagoas - Arapiraca Campus, Arapiraca - AL, Brazil
[2] Federal University of Alagoas - Computer Science Institute, Maceio - AL, Brazil
[3] Federal University of Campina Grande, Campina Grande - PB, Brazil

{*alexandre.barbosa@arapiraca.ufal.br*, *ebc.academico@gmail.com*}

Abstract. Programming is one of the basic competences in computer science, despite its importance, it is easy to find students with difficulties to understand the concepts required to use this skill. Several researchers report that the impossibility to achieve a quick and effective feedback, is one of the motivators for the problematic scenario. The professor, even when helped by the TAs, is not able to perform the reviews quickly, for this activity requires a huge amount of time. Fast feedback is extremely important to enable the learning of any concept. Some researches suggest the use of peer assessment as a means of providing feedback. However, it is quite common that the feedback provided by peers is not adequate. In this paper, we propose the use of simulated learners in a peer assessment approach as part of the teaching and learning processes of programming. Currently a software tool is being developed to include the proposal described in this paper.

1 Introduction

Programming is one of the basic competences in computer science, it is the basis for the development of several other competences required for professionals in the area. However, despite its importance, it is easy to find students who are demotivated and with difficulties to understand the concepts required to use this skill [7]. These difficulties causes a large number of failures, dropouts or the approval of students without the required level of knowledge [14] [6] [5].

Many factors are identified in literature as causing the problematic scenario related to programming courses. Several researchers report that the impossibility to achieve a quick, effective and individualized feedback, is one of the motivators for the problematic scenario [10] [12]. An individual follow up is impossible due to many students enrolled in the courses. In addition, there is a great complexity involved in the evaluation of a program, for it is necessary to understand how the programmer has developed the algorithm, so the professor needs to comprehend the line of reasoning adopted by the student. In this way, the professor, even when helped by the TAs, cannot provide an adequate and fast feedback about the solutions created by the students. This activity will require a huge amount

of time to manually open the code, compile, run and verify the output of every students solution for programming assignment. If the grading depends on the structure and the quality of code, in addition to program output correctness, the situation is a lot worse. Traditionally the real comprehension state of the contents of a programming course is known only months after the beginning of the course, when an evaluation activity is performed. After an evaluation it may be too late to make any intervention.

Fast feedback is of extreme importance to enable the learning of any concept [12]. Thus, some researches have been developed with the aim to propose methods and tools to facilitate the monitoring of the activities of students in programming courses. Some of these researches, such as [9][11][13], suggests the use of peer assessment as a means of providing fast and effective feedback. This solution is broadly used in Massive Open Online Courses (MOOCs), as described in [3][8], where the courses are applied to hundreds or thousands of people enrolled in them, and just as occurs in the context of programming, it is impossible for the professor to evaluate each solution. However, the peer assessment approach as a means of providing feedback has some problems. Many times the feedback provided by peers is not adequate, because the results are often not similar to the analysis of an expert [8]. It is quite common to find comments that are summarized to a phrase of congratulation or critique.

The reasons related to lack of effectiveness of feedback provided are quite distinct, these may occur due to poor understanding of the content of the activity, because of the student's low motivation, or due to the short time that one has available for the activities.

In [2] paper, it was observed the impact of learning was observed when a student is influenced by the performance of their peers, the authors describe that some students are encouraged to perform better, but others experiencing the same situations end up discouraged to perform better.

In this paper is proposed the use of simulated learners in a peer assessment approach used as part of the teaching and learning processes of programming. Two concerns are explored in this proposal: the first is related to the search of methods that enable a positive influence between students; the second concern is related to an approach that allows a less costly way of testing any proposal of applicability of peer assessment approach.

This paper is divided in five sections. In Section 2 the concept of peer assessment is presented. Observations on the implementation of peer assessment in a programming course context are shown in Section 3. The proposal of using simulated learners in the context of peer assessment for introductory programming is presented in Section 4. Finally the conclusions and future work are shown in the last section.

2 Peer Assessment

Peer assessment, or peer review, is an evaluation method where students have responsibilities that traditionally belong to professors only. Among these respon-

sibilities there are the review and the critique of the solutions proposed by their peers. This way, they can experience the discipline as students and also from the perspective of a TA. Usually in a peer assessment environment, students also conduct self assessment. This way, they can reflect on their solution when compared to other solutions, develop their critical thinking skills and improve understanding of the concepts covered in the course.

In traditional approach, the professor, even when helped by TAs, can not provide fast and adequate feedback for each solution proposed by the students. The comments provided by the professor are generic observations based on observation of all students solutions.

In accordance with [9], peer review is a powerful pedagogical method, because once students need to evaluate the work of their peers, they begin to teach and learn from each other. Thus, the learning process becomes much more active, making the learning qualitatively better than the traditional approach. Sstudents can spend more time on analysis and construction of their comments, creating more particular descriptions on a given solution and enriching discussion about the topic studied.

Thus, the use of peer review can reduce the workload on the professor, permitting the professor to focus on other pedagogical activities [9][3]. This evaluation approach can also enable the evaluation of large-scale complex exercises, which can not be evaluated in a automatically or semi-automatic fashion [3][8].

The success of peer assessment approach is strongly influenced by the quality of feedback provided. However, this feedback if often not adequate, the results are often not similar to the analysis of an expert [8]. In [8] is described that in many cases the evaluations of the students are similar to the TAs evaluation, however there are situations where the evaluations are graded 10% higher than the TAs evaluation, in extreme cases the grades could be 70% higher than the TAs evaluation. In [3] is mentioned that in general, there is a high correlation between the grades provided by students and TAs, but often in the evaluations from students the grades are 7% higher than the grades given by TAs.

Thus, we can conclude that peer assessment approach is a promising evaluation method, however there are improvements and adjusts to be applied to obtain richer discussions and more accurate assessments.

3 Peer Assessment in introductory programing courses

Human interaction is described as an essential feature for learning in many domains, including the introductory programming learning [13]. In classroom programming courses the contact between students occurs on a daily basis, allowing, for example, the discussion of the problems presented in the exercise lists, the developed solutions and the formation of groups for the projects of the course. This contact is many times inexistent in online programming courses, interactions in this environment are the human-machine type. Thus, using the peer assessment approach may enable human interaction on online courses, or enhance the interaction between humans in presencial classroom courses.

To encourage the assimilation of the topics, the use of practical exercises is quite common in programming courses, the practice of programming skills is crucial for learning. Many researchers also argue that the programming learning involves the reading and understanding of third-party code. Through peer assessment approach both characteristics can be obtained. The professor can develop new exercises, or choose problems proposed by others, while students will have to observe, understand and evaluate the codes of their peers, as well to compare these codes with their solution.

In [11] the use of a peer assessment approach to the context of programming courses is described, this approach is supported by a web application. The results described on the paper have a high correlation between the evaluations of the TAs and students, the correlation is lowest when the complexity of the exercise is higher.

An approach of peer assessment evaluation for the context of programming learning, also supported by a web application is presented in [9]. Five activities where graded using peer assessment, the occurrence of conflicts ranged from 61 % to activity with a lower incidence of conflict, up to 80 % for the activity with the highest occurrence of conflicts. The system considers that a conflict occurs when the student does not agree with the assessment provided.

In [9] the authors describes that if the peer reviews are conducted in an inadequate way, the failure rates can increase. For the teaching approach used at the programming course described in [13] there are two types of activities that require assessment, quizzes and mini projects. Among these activities only the mini projects are evaluated through peer assessment. Thus, students are not overloaded and the approach can be used appropriately.

Another problem that can emerge with the use of peer review in a programming context, is the increase of plagiarism. Once the assessment activity will be distributed among the students, the similarities of self-identification of codes can become more complicated. However, solutions are widely used to carry out the detection automatically similarities, such as MOSS [1] e GPLAG [4].

4 Simulated learners as peers in a peer assessment environment for introductory programing courses

In previous sections the advantages and disadvantages associated with the use of peer assessment in a general context, and when applied to the context of programming courses have been described. In both cases, the success of the approach is strongly influenced by the quality of the feedback given. Therefore, it is necessary to identify situations where there is inadequate feedback as well as conflict situations. Situations where inadequate feedback occurs are when, for any reason, the feedback does not help in the learning process. Conflict situations occur when the student does not agree with the assessment provided, or when there are huge variations on the evaluations provided. To perform a validation of this proposal or of any proposal involving peer assessment, it is necessary to

allocate the resources of time, physical space and adequate human resources. Thus, it can be said that the test of this approach is a costly activity.

Two concerns are explored in this proposal: how we can achieve methods that enable a positive influence between students in peer assessment environments, in other words, how a student can give a high quality feedback to their peers; and how a peer assessment approach can be tested with a lower cost, since any validation of these assessment approaches requires a huge amount of resources.

4.1 A scenario of use of peer assessment with simulated learners

Traditionally in a peer assessment environment, the professor must create the assignment and a set of assessment criteria. Then students develop their solutions observing the assessment criteria and submitting the solution to be evaluated by their peers. Each student's evaluation must meet the assessment criteria. The students should provide comments to peers explaining the reasons associated to the outcome and a grade or an evaluation concept (eg. A-, B+, C). Each student will have their code evaluated by their peers, and should assess the codes of other students. In Figure 1, it is illustrated the scenario previously described. There are variations in ways peer assessment approach is used, the scenario just mentioned has many characteristics which are similar to all the variations.

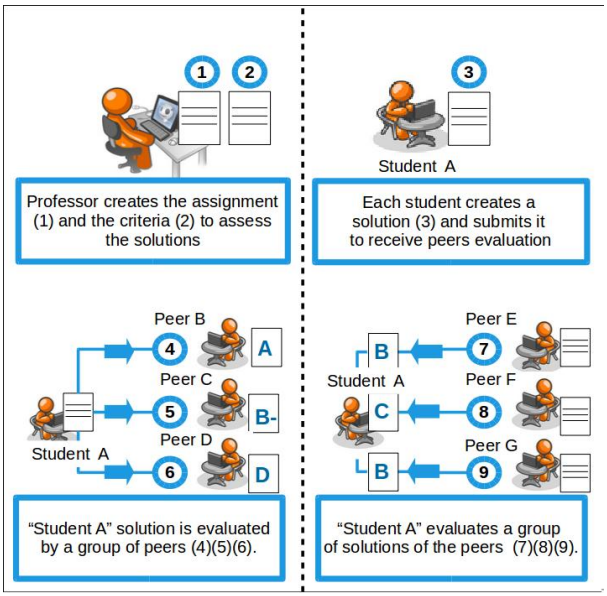

Fig. 1. A traditional peer assessment environment

In any peer assessment approach, it is possible to adopt pairing algorithms. Thereby, it is assured that evaluations are conducted by students with different

levels of knowledge. A student with low understanding of the subject will not be allocated to evaluate the work of another student in the same situation. Students with difficulties can clarify their doubts, while students with good understanding of the content should provide a good argumentation about their knowledge. However, it is not possible to ensure that a student evaluates the code that is the ideal for his/her learning and level of knowledge. As an example, in Figure 1, it is not possible to know if student "A" code is the best for peers "B", "C" and "D".

When a student does not agree with the evaluation provided by their peers, he/she will be able to request the intervention of the professor. This conflict situations are identified in [9]. However, in traditional peer assessment approach is not possible to identify incorrect evaluations provided by a student, or students that create biased evaluations only to help their fellows. As an example, in Figure 1, it is possible to see that different grades were given, but it is not possible to determine if the correct evaluations were given by peer "B", "C" or "D".

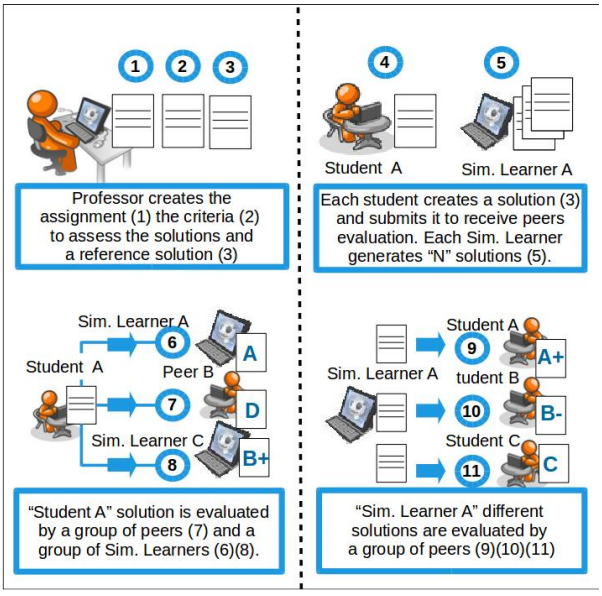

Fig. 2. A peer assessment environment using simulated learners

In a peer assessment environment that uses simulated learners, it is possible to solve the previous problems. As in traditional approach, the professor must create the assignment and a set of assessment criteria; in addition to that, he/she should provide a reference solution. Then, students develop their solutions, observing the assessment criteria and submitting the solution to be evaluated by their peers. At the same time, once a pairing algorithm can perform pairing of

the evaluators, each simulated learner must generate a code that is ideal for the learning and appropriate to the level of knowledge of each one of their peers, in this case, real students. Each student will have their code evaluated by their peers and by simulated students, and they should assess codes of other students, and codes of simulated students. In Figure 2 it is illustrated peer assessment environment with simulated learners. As an example, in Figure 2, it is possible to see that the simulated learner "A" generates a set of codes that are ideal for each student: "A", "B" and "C".

The identification of incorrect evaluations provided by a student, as well as students, who perform biased evaluations, could be carried out through the comparison of student's evaluations and the simulated student's evaluations. As an example, in Figure 2, it is possible to see that student "B", made an evaluation that is very different from the simulated learner's evaluations. In this way, it is possible to verify if the student did not understand the solution, or if the evaluation was created to help their fellows only.

Providing useful solutions to student learning A useful solution for the student learning does not always match the presentation of a correct and efficient code. Within the context of peer review may be more useful to display an incorrect code, as a way to make students to provide a set of review observations. To identify which type of code is best for a student; simulated learners can consult the representation of their cognitive status. In that way, it will be possible to the simulated learner identify the student misconceptions and errors in previous assignments, and generate variations of the reference solution that suits best for the student. Since multiple simulated students will be used, the codes that will be shown to students can range from efficient, correct and complete solutions to incorrect and/or incomplete solutions. Like that, it will be possible to check if students have different skills related to the content. To generate the variations from the reference solution, it is possible to combine testing techniques, such as mutant generation. Each code can be generated through the use of data related to the most common student's mistakes, emulating these behaviors and creating codes that are useful to learning. Once the research is in a preliminary stage, it is still not clear which artificial intelligence approaches should be used on the implementation of simulated students behaviors.

Assessment of students solutions Unlike what occurs in other contexts, for programming the evaluation of a solution can be automated or semi-automated. Typically a set of unit tests is applied to the code proposed by a student, who receives an indication that his/her code may be correct or incorrect, but no hint or comment is provided. Some researchers have investigated the use of different techniques to help assessment of codes and provide some guidance; these techniques usually employ software engineering metrics. Thus, simulated learners must be able to identify which subset of metrics can be used to perform the evaluation of the proposed solution for a student. The simulated learner should select the set of metrics that fits best to the objectives of the assignment and

the level of understanding that the student has at that moment. For each level of learning the same student can learn better if the set of metrics is properly selected. Each simulated student will use different strategies to evaluate the solutions provided by real students. Therefore, a variation between evaluations of simulated students is expected to occur. If an evaluation provided by a student has a very large variation in relation to the set of evaluations of simulated students, it will be necessary to investigate the motivation of this disparity. An acceptable variation threshold can be used to identify incorrect evaluations provided by students.

Discussing assessment criterias Once software engineering metrics were used in the evaluation, the explanation given by the simulated learner throughout the presentation of a set of metrics, is associated to the explanation of the metric choice and, possibly, of the snippet of the code where the observation is pertinent. Thereby, the simulated learner can help the professor to identify inadequate feedback, whenever an evaluation of a student is very different from the evaluation of a simulated learner, the professor and his tutors can then intervene.

4.2 Validation of peer assessent using simulated learners

Any validation of peer assessment approaches requires lots of physical space and a huge amount of human resources. As an example, if a validation of a pairing algorithm has to be done, it will be necessary to use a set of N students; this set must allow the creation of different profiles for evaluation of the pairing alternatives. The greater the possibilities of matching, the greater the amount of students required. Through the use of simulated learners any operational proposal of peer assessment can be tested at a much lower cost, since the physical space and human resources are drastically reduced. The researcher can determine how much of human resource will be available, replacing the students with simulated students. The researcher can also specify the desired behavior of students; the simulated students should emulate students with a high degree of understanding of the contents or with low understanding. After obtaining initial results with the use of simulated learners, the number of human individuals participating in an experiment can be increased, since it may be interesting to obtain a greater statistical power associated with the conclusions.

5 Conclusions and further work

In this paper, we have proposed the use of simulated learners in a peer assessment approach adopted as a support part of a programming course. The use of simulated learners as presented in this proposal aims to two goals: influence the students to provide better quality feedback; and allow for a less costly validation for peer assessment applied to programming contexts.

The research associated with the proposal presented in this paper is in a preliminary stage. Thus, the effectiveness of this proposal will be further evaluated in controlled experiments executed in the future. An open source software tool is being developed to include all aspects described throughout this proposal.

References

1. Bowyer, K., Hall, L.: Experience using "moss" to detect cheating on programming assignments. In: Frontiers in Education Conference, 1999. FIE '99. 29th Annual. vol. 3, pp. 13B3/18–13B3/22 (Nov 1999)
2. Frost, S., McCalla, G.I.: Exploring through simulation the effects of peer impact on learning. In: Proc. of the Workshops at the 16th International Conference on Artificial Intelligence in Education AIED 2013, Memphis, USA, July 9-13, 2013 (2013), http://ceur-ws.org/Vol-1009/0403.pdf
3. Kulkarni, C., Wei, K.P., Le, H., Chia, D., Papadopoulos, K., Cheng, J., Koller, D., Klemmer, S.R.: Peer and self assessment in massive online classes. ACM Trans. Comput. Hum. Interact. 20(6), 33:1–33:31 (Dec 2013)
4. Liu, C., Chen, C., Han, J., Yu, P.S.: Gplag: Detection of software plagiarism by program dependence graph analysis. In: Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 872–881. KDD '06, ACM, New York, USA (2006)
5. Mason, R., Cooper, G.: Introductory programming courses in australia and new zealand in 2013 - trends and reasons. In: Proc. of the Sixteenth Australasian Computing Education Conference - Volume 148. pp. 139–147. ACE, Darlinghurst, Australia (2014)
6. Mason, R., Cooper, G., de Raadt, M.: Trends in introductory programming courses in australian universities: Languages, environments and pedagogy. In: Proc. of the Fourteenth Australasian Computing Education Conference - Volume 123. pp. 33–42. ACE, Darlinghurst, Australia (2012)
7. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.D., Laxer, C., Thomas, L., Utting, I., Wilusz, T.: A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. In: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education. pp. 125–180. ITiCSE-WGR, New York, USA (2001)
8. Piech, C., Huang, J., Chen, Z., Do, C.B., Ng, A.Y., Koller, D.: Tuned models of peer assessment in moocs. CoRR abs/1307.2579 (2013)
9. de Raadt, M., Lai, D., Watson, R.: An evaluation of electronic individual peer assessment in an introductory programming course. In: Proc. of the Seventh Baltic Sea Conference on Computing Education Research - Volume 88. pp. 53–64. Koli Calling, Darlinghurst, Australia (2007)
10. Singh, R., Gulwani, S., Solar-Lezama, A.: Automated feedback generation for introductory programming assignments. In: Proc. of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 15–26. PLDI, New York, USA (2013)
11. Sitthiworachart, J., Joy, M.: Computer support of effective peer assessment in an undergraduate programming class. Journal of Computer Assisted Learning 24(3), 217–231 (2008)
12. Stegeman, M., Barendsen, E., Smetsers, S.: Towards an empirically validated model for assessment of code quality. In: Proc. of the 14th Koli Calling Int. Conf. on Computing Education Research. pp. 99–108. Koli Calling, New York, USA (2014)

13. Warren, J., Rixner, S., Greiner, J., Wong, S.: Facilitating human interaction in an online programming course. In: Proc. of the 45th ACM Technical Symposium on Computer Science Education. pp. 665–670. SIGCSE, New York, USA (2014)
14. Yadin, A.: Reducing the dropout rate in an introductory programming course. ACM Inroads 2(4), 71–76 (2011)

Simulated Learners for Testing Agile Teaming in Social Educational Games

Steeve Laberge and Fuhua Lin

School of Computing and Information Systems, Athabasca University, Edmonton, Canada
slaberge@acm.org, oscarl@athabascau.ca

Abstract. This paper proposes an approach for creating and testing an multiagent systems based adaptive social educational game (SEG), QuizMASter, using the concept of simulated learners to overcome experimentation complexity and unpredictable student availability, as is typical with online learning environments. We show that simulated learners can play two roles. First, it can be used for testing the game planning, scheduling and adaptive assessment algorithms. With some degree of success met with our initial experimentation with QuizMASter, advanced planning and coordination algorithms are now needed to allow the game-based assessment platform to realize its full potential. The multi-agent system approach is suitable for modeling and developing adaptive behaviour in SEGs. However, as we have found with our early prototypes, verifying and validating such a system is very difficult in an online context where students are not always available. MAS-based assessment game planning and coordination algorithms are complex and thus need simulated learners for testing purposes. Second, to overcome unpredictable student availability, we modeled QuizMASter as a new class of socio-technical system, human-agent collective (HAC). In the system, human learners and simulated learners (smart software agents) engage in flexible relationship in order to achieve both their individual and collective goals, while simulated learners are selected for serving as virtual team members.

Keywords: social educational agents, multiagent systems, simulated learners

1 Introduction

For decades, educational games have proven to be an effective means to motivate learners and enhance learning. Social (multi-player) educational games (SEGs) offer many opportunities to improve learning in ways that go beyond what a single-player game can achieve because SEGs allow players to be social, competitive, and collaborative in their problem solving. The presence of other players can be used to increase playability and to help teach team-work and social skills. SEGs promote intragroup cooperation and intergroup competition [1]. However, existing SEGs share many of the shortcomings of classroom role-playing. Setting

up existing SEGs is logistically challenging, expensive, and inflexible. Furthermore, players become bored after going through existing SEGs once or twice.

To test such a social educational game, we face two difficulties. One is how to test the planning and scheduling algorithms. Another is how to meet the need of agile team formation. In SEGs, group formation has big impact on group learning performance. Poor group formation in social games can result to homogeneity in student characteristic such that the peer learning is ineffective. Thus, there is a need to constitute a heterogeneous group SEGs that constitutes students with different collaborative competencies and knowledge levels. However, without empirical study it becomes difficult to conclude which group characteristics are desirable in the heterogeneity as different game-based learning needs may require different group orientations. Previous research has focused on various group orientation techniques and their impact on group performance like different learning styles in group orientation [2–4]. However, there is need to investigate the impact of other group orientation techniques on group performance like grouping students based on their collaboration competence levels. Furthermore, most of the previous research in group-formation focuses on classroom based learning. Also, it lacks the true experiment design methodology that is recommended when investigating learning outcomes from different game-based learning strategies. Simulated learners methodology [5] has shown a promising way to solve these challenges.

In this paper, we show that simulated learners can play two roles. First, it can be used for testing the game planning, scheduling and adaptive assessment algorithms. Second, working with human learners and forming human-agent collectives (HAC), simulated learners serve as virtual team members to enable asynchronous game-based learning in a context where student availability is unpredictable. This paper is structured as follows: In Section 2 we discuss recent advancements and related work. Section 3 describes QuizMASter. Section 4 presents the proposed architecture for development of QuizMASter. Section 5 explains how we intend to use simulated learners for testing QuizMASter. Finally, Section 6 concludes.

2 Related Work

Researchers have found that learning can be more attractive if learning experiences combine challenge and fun [6]. As social networks have become popular applications, they have given rise to social games. This kind of game is played by users of social networks as a way to interact with friends [7] and has become a part of the culture for digital natives. Social games have unique features that distinguish them from other video games. Those features are closely linked with the features of social networks [8]. Social games can make a contribution to social learning environments by applying game mechanics and other design elements, 'gamifying' social learning environments to make them more fun and engaging. For games to be effective as a learning tool, a delicate balance must be maintained between playability and educational value [9, 10], and between

game design and learning principles. Methods have been proposed for making valid inferences about what the student knows, using actions and events observed during gameplay. Such methods include evidence-centered-design (ECD) [11, 12]; the learning progressions model [13], the ecological approach to design of e-learning environments [14], stealth assessment [15], game analytics [16], and learning analytics [17]. Most of the new concepts target an ever-changing learning environment and learner needs, as today's education moves toward a digital, social, personalized, and fun environment. Moreover, as is the case for all competitive games, an equal match between players is essential to self-esteem and to maintain a high degree of player interest in the game. Hence, we need mechanisms and models that can aggregate the current performance and preferences of players, and accurately predict student performance in the game. Software agents have been used to implement consistent long-term intelligent behaviour in games [18], multi-agent collaborative team-based games [19], and adaptive and believable non-player character agents simulating virtual students [20]. The use of agent technologies leads to a system characterized by both autonomy and a distribution of tasks and control [21]. This trend has two aspects. First, game-based learning activities should be carefully orchestrated to be social and enjoyable. Second, game scheduling and coordination should be highly adaptive and flexible. However, nobody has yet developed models, algorithms, and mechanisms for planning, scheduling, and coordination that are suitable for creating and testing SEGs.

3 QuizMASter

QuizMASter is designed to be a formative assessment tool that enables students to be tested within a multi-player game [22]. Two or more students simultaneously log in remotely to the system via a Web-based interface. Each student is represented by one avatar in this virtual world. Students are able to view their own avatar as well as those of their opponents.

Each game has the game-show host who is also represented by an avatar visible to all contestants [22]. The game-show host poses each of the game questions to all the contestants. The students hear the voice of the host reading each question and view them displayed on their screens. They individually and independently from one another answer each question by, for instance, selecting an answer from available choices in a multiple-choice format. Each correct answer would receive one mark. Figure 1 shows a screen shot of QuizMASter.

3.1 Characteristics of QuizMASter

The environment for QuizMASter has the following characteristics:

Flexibility. The environment for QuizMASter needs flexibility for game enactment, to be able to cope with dynamic changes of user profiles, handle fragmentation of playing and learning time needed to accomplish activities and tasks,

Fig. 1. QuizMASter in Open Wonderland

adequately handle exceptional situations, predict changes due to external events, and offer sufficient interoperability with other software systems in educational institutions. Individual learners have particular interests, proficiency levels, and preferences that may result in conflicting learning goals.

Social ability and interactivity. The environment for QuizMASter should encourage interaction and collaboration among peers, and should be open to participation of students, teachers, parents, and experts on the subjects being taught. Web 2.0 has had a strong influence on the ways people learn and access information, and schools are taking advantage of this trend by adopting social learning environments. One way to engage learners in a collaborative production of knowledge is to promote social rewards.

User control. One of the most desirable features of social education games is to empower players with control over the problems that they solve. For example, in QuizMASter, students, parents, and teachers can design new rules to create their own games and modify the game elements to fit different knowledge levels.

Customization. Customization is a core principle that helps accommodate differences among learners [23]. Teachers could build a QuizMASter that has its own style and rules to determine the game's level of difficulty, to gear the game for specific goals or a specific group of learners. Some teachers may be interested in sharing collections of rules to fit the learning and play styles of their students. Like teachers, learners/players can be co-creators of their practice space through building new game scenarios, creating their own rules, sharing their strategies and making self-paced challenges [23].

4 The Proposed Architecture

Multi-agent technologies are considered most suitable for developing SEGs as it will lead to systems that operate in a highly dynamic, open, and distributed environment. In an MAS-based SEG, each learner/player is represented as an autonomous agent, called learner agent. MAS technologies, such as goal orientation and the Belief-Desire-Intention (BDI) paradigm, is used as the foundation for the agent architecture. These learner agents are able to reason about the learning goals, the strengths and weaknesses of learners and update the learner models.

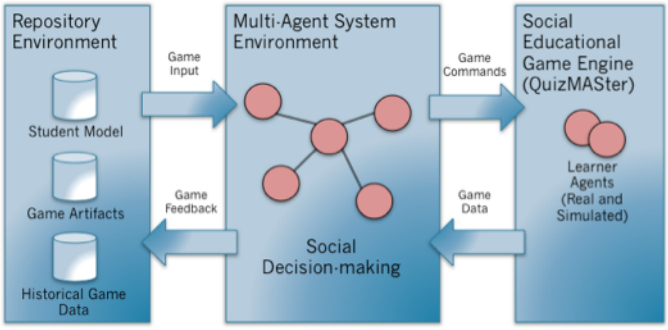

Fig. 2. Architecture for MAS-Based Social Educational Game Environment

Whenever a learner enters the system to play a social educational game, the learner agent will retrieve her/his learner model and acquire preferences about the current game-playing, and then send to a game management agent (GMA) of the system. The GMA is designed for setting up and maintaining teams for the system. The GMA will assign the learner to participate in a most suitable team that is undermanned according to the profile and preferences of the learner. The team will be configured in accordance with the game model by the GMA. Once the team has been completely formed, the GMA will create a game scheduling agent (GSA), a game host agent (GHA), and an assessment agent (AA) for each team. The GSA will continuously generate a game sequence dynamically adapted to the team's knowledge level (represented as a combined learner model [24]. The GHA will receive the game sequence from the scheduling agent and execute game sequence with the learners in the team. It will also be responsible for capturing data about learner/player performance. The AA will receive and interpret game events and communicate with the learner agents to update the learner model as necessary.

The GSA will dynamically schedule the game on the fly through interacting with other agents with a coordination mechanism, considering both the current world state and available resources, and solving conflicts in preferences and learning progression between the agents. The goal of the GSA is to optimize

the playability and educational values. We will model the game elements as resources. To solve the distributed constraint optimization problem, we are developing multiagent coordination mechanisms and scheduling algorithms to be used by the GSA.

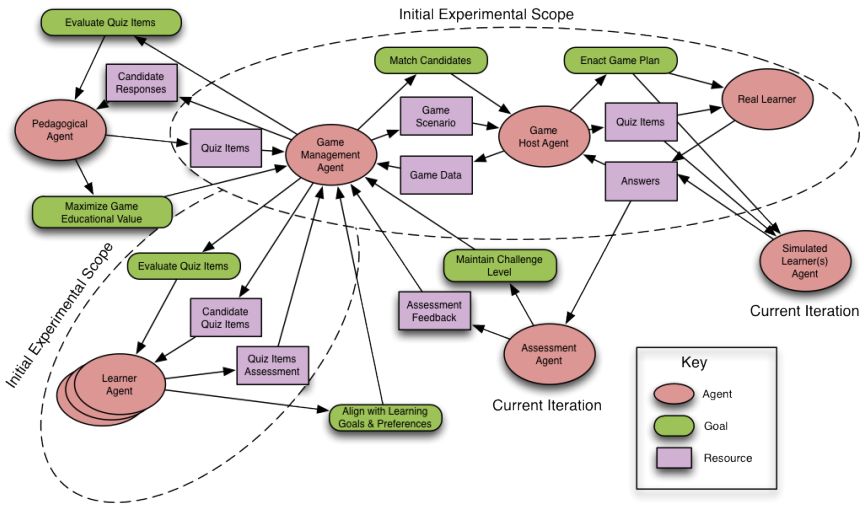

Fig. 3. MAS-Based SEG Agent Interaction Model

4.1 Planning and Scheduling Algorithms

The planning algorithms refer to the (local) planning algorithm of learner agents. To develop planning algorithms for learner agents, the following supporting models have been taken into consideration: (i) Learner models that accumulate and represent beliefs about the targeted aspects of skills. They are expressed as probability distributions for competency-model variables (called nodes) describing the set of knowledge and skills on which inferences are to be based. (ii) Evidence models that identify what the learner says or does, and provide evidence about those skills that express how the evidence depends on the competency-model variables in a psychometric model. (iii) Task/action models that express situations that can evoke required evidence. To design an action model, we adopt a model called Fuzzy Cognitive Goal Net [25] as the planning tool by combining the planning capability of Goal Net and reasoning ability of Fuzzy Cognitive Maps (FCMs). These FCMs give the learner agent a powerful reasoning ability for game context and player interactions, giving the task model accurate context awareness and learner awareness. We are developing coordination mechanisms

for the GMA and the GSA to solve the problem of team formation, scheduling and coordination in a highly flexible and dynamic manner. We considered the following concepts or methods:

(i) Contract-net protocols (CNPs) are used as a coordination mechanism by the GMA with a game model repository to timely form a team from all available players, using mutual selection and exchanging information in a structured way to converge on assignments. Each involved learner can delegate the negotiation process to its agent. These agents will strive to find a compromise team-joining decision obeying hard learning constraints while simultaneously resolving individual conflicts of interest.

(ii) The problem of scheduling and customizing a social educational game can be solved through social-choice-based customization. We view the SEG gameplay design as an optimization problem. Resources must be allocated through strategically scheduling, and coordinating a group of players according to their preferences and learning progressions. The constraints include key learning principles that inform the design of mechanics: challenge, exploration, risk taking, agency, and interactions [26-27]. The objective of the GSA is to maximize the learnability and engagement of the learners in the group. Social choice theory in MAS concerns the design and formal analysis of methods for aggregating preferences of multiple agents and collective decision-making and optimizing for preferences [28-29]. For example, we use a voting-based group decision-making approach such as Single Transferable Voting [30] to aggregate learner preferences and learning progression because it is computationally resistant to manipulation [31]. The purpose is to take information from individuals and combine it to produce the optimal result.

(iii) To support the need for dynamic decision making in the MAS-based SEG architecture, our current line of investigation is the concept of social choice Markov Decision Process (MDP) as recently proposed by Parkes and Procaccia [32]. In a social choice MDP, each state is defined by "preference profiles", which contain the preferences of all agents against a set of alternatives for a given scenario. The course of action from any given state is determined by a deterministic social choice function (the policy, in the context of the MDP) that takes into account the likelihood of transitions and their rewards. However, a preference profile is subject to change over time, especially in a live SEG context. For example, a learner that unexpectedly answers a question initially deemed beyond the learner's perceived level of comprehension would likely trigger a change of belief in the agents and potentially alter their ranking of alternatives. And since the number of alternatives in a SEG can be very large, the state space for any given SEG is huge, making the computation of optimal decision-making policies excessively difficult. We solve this problem by exploiting symmetries that exist in certain game types (e.g. in a quiz game SEG format, using a reduced set of question types that share common characteristics as a basis for alternatives as opposed to individual questions).

5 Simulated Learners

It is our view that the Belief-Desire-Intention (BDI) model is ideally suited for modeling and simulating learner behaviour. According to Jaques and Vicari (2007) [33], intelligent agents based on Bratman's Belief-Desire-Intention model, or BDI agents, are commonly used in modeling cognitive aspects, such as personality, affect, or goals. Píbil et al. (2012) claim BDI agent architecture is "a currently dominant approach to design of intelligent agents" [34]. Wong et al. (2012) describes the suitability of the BDI agent model for applications where both reactive behavior and goal-directed reasoning are required [35]. Soliman and Guetl (2012) suggest that BDI maps well onto models for pedagogically based selection of sub plans within a hierarchical planning strategy – "apprenticeship learning model" given as example [36]. They also talks about advantage of breaking plans down into smaller plans to allow for different "pedagogical permutations" allowing the agent to adapt to different learning styles, domain knowledge, and learning goals. Norling (2004) attributes the successful use of BDI agents for modeling human-like behavior in virtual characters to BDI's association to "folk psychology" [37]. This allows for an intuitive mapping of agent framework to common language that people use to describe the reasoning process. Of particular importance to this study is the way that implementations of the BDI architecture model long-term or interest goals. We have selected the Jason^{TM} [38] platform for providing multi-agent BDI programming in AgentSpeak.

A shortcoming of the BDI paradigm is that although it is intended to be goal-driven, in most implementations this means/amounts to using goals to trigger plans, but does not support the concept of long-term goals or preferences [39], such as a student's long term learning goals, or the pedagogical goals of a CA. They feel that these types of goals are difficult to represent in most BDI systems because they signify an ongoing desire that must be maintained over a long period of time compared to relative short goal processing cycles. It is left to the developer to implement this type of preference goal through the belief system of the agent, modifications to the platform or environment, or other methods of simulating long-term goals.

Hübner, Bordini, and Wooldridge (2007) describe plan patterns for implementing declarative goals, with varying levels of commitment in AgentSpeak [40]. Bordini et al. (2007) expand on this in their chapter on advanced goal-based programming [38]. While AgentSpeak and Jason support achievement goals, these patterns are intended to address the lack of support for "richer goal structures", such as declarative goals, which they feel are essential to providing agents with rational behaviour. Pokahr et al. (2005) point out that the majority of BDI interpreters do not provide a mechanism for deliberating about multiple and possibly conflicting goals [41]. It is worth noting that there are "BDI inspired" systems that are more goal-oriented, such as Practionist and GOAL [42]. The Jason multi-agent platform for BDI agents was selected for this project because it is a well-established open-source project that is being actively maintained. It supports both centralized and distributed multi-agent environments. Píbil et

al. (2012) describes Jason as "one of the popular approaches in the group of theoretically-rooted agent-oriented programming languages" [34]. A major advantage of Jason is that it is easy to extend the language through Java based libraries and other components. Internal actions can allow the programmer to create new internal functionality or make use of legacy object-oriented code [38]. However, Píbil et al. (2012) caution that the use of such extensions, if used too heavily, can make the agent program difficult to comprehend without understanding the functionality of the Java code [34]. They raise the concern that novice programmers have few guidelines for choosing how much to program in AgentSpeak, and how much too program in Java. The usefulness of being able to extend Jason can be demonstrated by two examples of current research into integrating BDI with Bayesian Networks. Modeling of some student characteristics requires a probabilistic model; Bayesian Networks (BN) being a popular choice in recent years [43-44]. Recent work by Kieling and Vicari (2011) describes how they have extended Jason to allow a BDI agent to use a BN based probabilistic model. Similarly, Silva and Gluz (2011) extend the AgentSpeak(L) language to implement AgentSpeak(PL) by extending the Jason environment. AgentSpeak(PL) integrates probabilistic beliefs into BDI agents using Bayesian Networks [45]. Experimentation with QuizMASter to date has enabled the modelling of simulated learners in virtual worlds with an initial focus on their appearance, gestures, kinematics, and physical properties [46]. Recent related research work in that area has been on the creation of engaging avatars for 3D learning environments [47]. Employing the theory of Transformed Social Interaction (TSI) [48], simulated learners were designed with the following abilities:

(i) Self-identification: The self-identification dimension of TSI was implemented using facial-identity capture with a tool called FAtiMA. Each of the users' face were morphed with their default avatar agent's face to capitalize on human beings' disposition to prefer faces similar to their own and general preference of appearing younger (see Fig. 4).

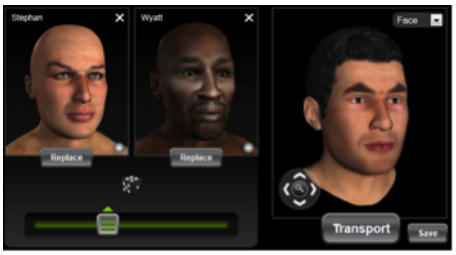

Fig. 4. Transformed Social Interaction – Image Morphing Technique

(ii) Sensory-abilities: Sensory-abilities dimension of TSI were implemented using a movement and visual tracking capability. The general challenge of sensory abilities implementation lies in two areas: the complexity of human senses and

the processing of sensory data of different modality and historicity. For the reason of simplicity, only visual tracking capability was exploited.

(iii) Situational-context: The situational-context dimension of TSI was implemented by using the best-view feature of Open Wonderland, whereby the temporal structure of a conversation can be altered.

The main idea of this research has been to explore the methodology for developing simulated learners for simulating and testing SEGs. That is, behind a simulated learner is an agent. Or we can say a simulated learner is an agent's avatar. All avatars, including real students' avatars and agent-based simulated learners, live in the virtual worlds, while the agents live in the multi-agent system. The integration of multi-agent systems with virtual worlds adds intelligence to the SEG platform and opens a number of extremely interesting and potentially useful research avenues concerning game-based learning. However, the advanced algorithms that support game planning, coordination and execution are difficult to test with real subjects considering the overhead involved is seeking authorization and the unpredictable availability of real life subjects in an online environment. This where an expanded view of simulated learners comes into play. The advantages of a simulated environment that closely approximates human behaviour include: (1) It allows for rapid and complete testing of advanced algorithms for game based adaptive assessment as well as SEG planning, coordination and execution in a simulated environment. The efficiency of the algorithms can be measured without first securing the availability of students; (2) With proper learner modeling and adaptive behaviour, simulated learners can engage with real life learners in friendly competitive games for the purpose of formative assessment, again working around the issue of availability of real students in an online learning environment.

6 Conclusions

As our recent experimentation suggests, many outstanding challenges must be addressed in developing intelligent SEGs. As we get closer to real world testing of our experimental game based assessment framework, we are faced with the complexity of enrolling real life learners in an e-learning environment and the variability that human interactions introduce in the measurement of adaptive algorithm efficiency. This is where we see the value of simulated learners. At this stage of our research, simulated learners have been rendered as Non Person Characters (NPCs) controlled by BDI agent running in the multi-agent system based virtual world. Our medium term goal is to extend the existing system to a particular learning subject (e.g., English language learning) to verify the effectiveness of the proposed virtual assessment environment and the benefit that students perceive from interacting with the proposed NPCs.

For simulated learners to be successful in our experimental framework, they must closely approximate the performance of real learners. The simple, pre-encoded behaviour we have implemented so far in the NPCs for QuizMASter will not suffice to demonstrate the efficiency of our adaptive algorithms and

allow for simulated learner agents to act as virtual players in our game based assessment framework. Current outstanding research questions within our group are:

1. How do we add intelligence and adaptive behaviour to the simulated learner agents while preserving our ability to obtain predictable and repeatable test results from our adaptive MAS framework?
2. How much autonomy can we afford to give to simulated learners in terms of independent thought and action, and to which degree should a simulated learner be able to adjust its behaviour as a function of its interactions with other agents, including real life learners?
3. How do we incorporate modern game, learning and assessment analytics in the supporting adaptive MAS framework in order to maximize the value of simulated learners as a means to perform non-intrusive, formative assessment?

References

1. Romero, M., Usart, M., Ott, M., Earp, J., de Freitas, S., and Arnab, S.: Learning through playing for or against each other. Promoting Collaborative Learning in Digital Game Based Learning. ECIS 2012 Proceedings. Paper 93 (2012)
2. Alfonseca, E., Carro, R. M., Martin, E., Ortigosa, A., and Paredes, P.: The impact of learning styles on student grouping for collaborative learning: a case study. User Modeling and User-Adapted Interaction, 16(3–4), 377–401 (2006)
3. Deibel, K.: Team formation methods for increasing interaction during in-class group work. In ACM SIGCSE Bulletin (Vol. 37, 291–295). Caparica, Portugal (2005)
4. Grigoriadou, M., Papanikolaou, K. A., and Gouli, E.: Investigating How to Group Students based on their Learning Styles. In In ICALT, 2006 1139–1140 (2006)
5. McCalla, G. and Champaign J.: AIED Workshop on Simulated Learners. AIED 2013 Workshops Proceedings, Volume 4 (2013)
6. Vassileva, J.: Toward social learning environments. IEEE Transactions on Learning Technologies, 1(4), 199–213 (2008)
7. Klopfer, E., Osterweil, S., and Salen, K.: Moving learning games forward: obstacles, opportunities and openness, the education arcade. MIT (2009)
8. Jarvinen, A.: Game design for social networks: Interaction design for playful dispositions. In Stephen N. Spencer (Ed.), Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games (Sandbox '09), 95–102. ACM, New York, NY, USA (2009)
9. Van Eck, R.: Building Artificially Intelligent Learning Games. In V. Sugumaran (Ed.), Intelligent Information Technologies: Concepts, Methodologies, Tools and Applications, 793-825 (2008)
10. Augustin, T., Hockemeyer, C., Kickmeier-Rust, M. D., and Albert, D.: Individualized Skill Assessment in Digital Learning Games: Basic Definitions and Mathematical Formalism. IEEE Trans. on Learning Technologies, 4(2), 138-147 (2011)
11. Mislevy, R. J., and Haertel, G. D.: Implications of evidence-centered design for educational testing. Educational Measurement: Issues and Practice, 25(4), 6–20 (2006)
12. Mislevy, R. J., Steinberg, L. S., and Almond, R. G.: On the structure of educational assessments. Measurement: Interdisciplinary Research and Perspectives, 1, 3-67 (2003)

13. Corcoran, T., Mosher, F. A., and Rogat, A.: Learning Progressions in Science: An Evidence-Based Approach to Reform (Research Report No. RR-63). Center on Continuous Instructional Improvement, Teachers College—Columbia University (2009)
14. McCalla, G.: The Ecological Approach to the Design of E-Learning Environments: Purpose-based Capture and Use of Information About Learners. Journal of Interactive Media in Education. (7), 1-23 (2004)
15. Shute, V. J.: Stealth assessment in computer-based games to support learning. Computer games and instruction. Charlotte, NC: Information Age Publishers, 503-523 (2011)
16. Long, P., and Siemens, G.: Penetrating the fog: analytics in learning and education. Educause Review Online 46 (5): 31–40 (2011)
17. El-Nasr, M. S., Drachen, A., and Canossa, A.: Game Analytics: Maximizing the Value of Player Data, Springer (2013)
18. Oijen, J., and Dignum, F.: Scalable Perception for BDI-Agents Embodied in Virtual Environments, Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on, Vol. 2, 46-53, 22-27 (2011)
19. Patel, P., and Hexmoor, H.: Designing BOTs with BDI agents, Collaborative Technologies and Systems, 2009. CTS '09. International Symposium on, 180-186 (2009)
20. Lim, M., Dias, J., Aylett, R., and Paiva, A.: Creating adaptive affective autonomous NPCs, Autonomous Agents and Multi-Agent Systems (AAMAS), Springer, 24(2), 287-311 (2012)
21. Sterling, L. S. and Taveter, K.: The art of agent-oriented modeling, MIT Press (2009)
22. Dutchuk, M., Muhammadi, K. A., and Lin, F.: QuizMASter - A Multi-Agent Game-Style Learning Activity, EduTainment 2009, Aug 2009, Banff, Canada, Learning by Doing, (eds.), M Chang, et al., LNCS 5670, 263-272 (2009)
23. de Freitas, S. and Oliver, M.: How can exploratory learning with game and simulation within the curriculum be most effectively evaluated? Computers and Education. 46(3), 249-264 (2006)
24. Shabani, S., Lin, F. and Graf, S.: A Framework for User Modeling in QuizMASter. Journal of e-Learning and Knowledge Society, 8(3), 1826-6223 (2012)
25. Jennings N. R., L. Moreau, D. Nicholson, S. Ramchurn, S. Roberts, T. Rodden, and A. Rogers: Human-agent collectives. Commun. ACM 57, 12 (2014), 80-88 (2014)
26. Cai, Y., Miao, C., Tan, A.-H., and Shen, Z.: Fuzzy cognitive goal net for interactive storytelling plot design. In Proceedings of the 2006 ACM SIGCHI Int. conf. on Advances in comput. entertainment technology. ACM, NY, USA, Article 56 (2006)
27. Gee, James P.: Good Video Games + Good Learning. New York: Peter Lang (2008)
28. Gee, James P.: Learning by Design: Games as Learning Machines, Interactive Educational Multimedia, Vol. 8, April Ed. 2004, 13-15 (2004)
29. Brandt, F., Conitzer, V., and Endriss, U.: Computational Social Choice, Chapter 6 of book edited by Weiss, G., Multiagent Systems (2nd edition), 213-283 (2013)
30. Conitzer, V.: Making Decisions Based on the Preferences of Multiple Agents, Communications of the ACM, 53(3), 84-94 (2010)
31. Bartholdi, J. J. and Orlin, J. B: Single Transferable Vote Resists Strategic Voting. Social Choice and Welfare, 8, 341-354 (1991)
32. Parkes, D. C. and Procaccia, A. D.: Dynamic Social Choice with Evolving Preferences. In M. desJardins and M. L. Littman (eds.), AAAI : AAAI Press (2013)
33. Jaques, P. A., Vicari, R. M.: A BDI approach to infer student's emotions in an intelligent learning environment. Computers & Education, 49(2), 360–384 (2007)

34. Píbil, R., Novák, P., Brom, C., Gemrot, J.: Notes on Pragmatic Agent-Programming with Jason. In L. Dennis, O. Boissier, & R. H. Bordini (Eds.), Programming Multi-Agent Systems, 58–73. Springer Berlin Heidelberg (2012)
35. Wong, W., Cavedon, L., Thangarajah, J., Padgham, L.: Flexible Conversation Management Using a BDI Agent Approach. In Y. Nakano, M. Neff, A. Paiva, & M. Walker (Eds.), Intelligent Virtual Agents, 7502, 464–470). Springer (2012)
36. Soliman, M., Guetl, C.: Experiences with BDI-based design and implementation of Intelligent Pedagogical Agents. In 2012 15th International Conference on Interactive Collaborative Learning (ICL) (1–5). Presented at the 2012 15th International Conference on Interactive Collaborative Learning (ICL) (2012)
37. Norling, E.: Folk Psychology for Human Modelling: Extending the BDI Paradigm. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1 (202–209). Washington, DC, USA: IEEE (2004)
38. Bordini, R. H., Hübner, J. F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. John Wiley & Sons (2007)
39. Bellotti, F., Berta, R., De Gloria, A., Lavagnino, E.: Towards a conversational agent architecture to favor knowledge discovery in serious games. In Proc. of the 8th Int. Conf. on Advances in Comput. Entertainment Technology, 17:1–17:7 (2011)
40. Hübner, J. F., Bordini, R. H., Wooldridge, M.: Programming Declarative Goals Using Plan Patterns. In M. Baldoni & U. Endriss (Eds.), Proceedings on the Fourth International Workshop on Declarative Agent Languages and Technologies, held with AAMAS 2006 (123–140). Springer Berlin Heidelberg (2007)
41. Pokahr, A., Braubach, L., Lamersdorf, W. (2005). A BDI architecture for goal deliberation. In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (1295–1296). New York, NY, USA: ACM (2005)
42. Braubach, L., Pokahr, A.: Representing Long-Term and Interest BDI Goals. In L. Braubach, J.-P. Briot, & J. Thangarajah (Eds.), Programming Multi-Agent Systems (pp. 201–218). Springer Berlin Heidelberg (2010)
43. Conati, C., Maclaren, H.: Empirically building and evaluating a probabilistic model of user affect. User Modeling and User-Adapted Interaction, 19(3), 267–303 (2009)
44. Chrysafiadi, K., Virvou, M.: Student modeling approaches: A literature review for the last decade. Expert Systems with Applications, 40(11), 4715–4729 (2013)
45. Silva, D. G., Gluz, J. C.: AgentSpeak(PL): A New Programming Language for BDI Agents with Integrated Bayesian Network Model. In 2011 International Conference on Information Science and Applications (ICISA) (pp. 1–7). Presented at the 2011 International Conference on Information Science and Applications (ICISA) (2011)
46. McClure, G., Chang, M., Lin, F.: MAS Controlled NPCs in 3D Virtual Learning Environment, The International Workshop on Smart Learning Environments at the 9th International Conference on Signal-Image Technology and Internet-Based Systems, Japan, Kyoto, 2013-12-02, 1026 – 1033, DOI: 10.1109/SITIS.2013.166 (2013)
47. Walsh, T.: An Empirical Study of the Manipulability of Single Transferable Voting, Proceedings of the 2010 conference on ECAI, 257-262 (2010)
48. Leung, S., Virwaney, S., Lin, F., Armstrong, A J, Dubbelboer, A.: TSI-enhanced Pedagogical Agents to Engage Learners in Virtual Worlds", International Journal of Distance Education Technologies, 11(1), 1-13 (2013)

Is this model for real? Simulating data to reveal the proximity of a model to reality

Rinat B. Rosenberg-Kima[1], Zachary A. Pardos[2]

[1] Tel-Aviv University
rinat.rosenberg.kima@gmail.com
[2] University of California, Berkeley
pardos@berkeley.edu

Abstract. Simulated data plays a central role in Educational Data Mining and in particular in Bayesian Knowledge Tracing (BKT) research. The initial motivation for this paper was to try to answer the question: given two datasets could you tell which of them is real and which of them is simulated? The ability to answer this question may provide an additional indication of the goodness of the model, thus, if it is easy to discern simulated data from real data that could be an indication that the model does not provide an authentic representation of reality, whereas if it is hard to set the real and simulated data apart that might be an indication that the model is indeed authentic. In this paper we will describe analyses of 42 GLOP datasets that were performed in an attempt to address this question. Possible simulated data based metrics as well as additional findings that emerged during this exploration will be discussed.

Keywords: Bayesian Knowledge Tracing (BKT), simulated data, parameters space.

1 Introduction

Simulated data has been increasingly playing a central role in Educational Data Mining [1] and Bayesian Knowledge Tracing (BKT) research [1, 4]. For example, simulated data was used to explore the convergence properties of BKT models [5], an important area of investigation given the identifiability issues of the model [3]. In this paper, we would like to approach simulated data from a slightly different angle. In particular, we claim that the question "*given two datasets could you tell which of them is real and which of them is simulated?"* is interesting as it can be used to evaluate the goodness of a model and may potentially serve as an alternative metric to RMSE, AUC, and others. In a previous work [6] we started approaching this problem by contrasting two real datasets with their corresponding two simulated datasets with Knowledge Tracing as the model. We found a surprising close to identity between the real and simulated datasets. In this paper we would like to continue this investigation by expanding the previous analysis to the full set of 42 Groups of Learning Opportunities (GLOPs) real datasets generated from the ASSISTments platform [7].

Knowledge Tracing (KT) models are widely used by cognitive tutors to estimate the latent skills of students [8]. Knowledge tracing is a Bayesian model, which assumes that each skill has 4 parameters: two knowledge parameters include initial (prior knowledge) and learn rate, and two performance parameters include guess and slip. KT in its simplest form assumes a single point estimate for prior knowledge and learn rate for all students, and similarly identical guess and slip rates for all students. Simulated data has been used to estimate the parameter space and in particular to answer questions that relate to the goal of maximizing the log likelihood (LL) of the model given parameters and data, and improving prediction power [7, 8, 9].

In this paper we would like to use the KT model as a framework for comparing the characteristics of simulated data to real data, and in particular to see whether it is possible to distinguish between the real and simulated datasets.

2 Data Sets

To compare simulated data to real data we started with 42 Groups of Learning Opportunities (GLOPs) real datasets generated from the ASSISTments platform[1] from a previous BKT study [7]. The datasets consisted of problem sets with 4 to 13 questions in linear order where all students answer all questions. The number of students per GLOP varied from 105 to 777. Next, we generated two synthetic, simulated datasets for each of the real datasets using the best fitting parameters that were found for each respective real datasets as the generating parameters. The two simulated datasets for each real one had the exact same number of questions, and same number of students.

3 Methodology

The approach we took to finding the best fitting parameters was to calculate LL with a grid search of all the parameters (prior, learn, guess, and slip). We hypothesized that the LL gradient pattern of the simulated data and real data will be different across the space. For each of the datasets we conducted a grid search with intervals of .04 that generated 25 intervals for each parameter and 390,625 total combinations of prior, learn, guess, and slip. For each one of the combinations LL was calculated and placed in a four dimensional matrix. We used fastBKT [12] to calculate the best fitting parameters of the real datasets and to generate simulated data. Additional code in Matlab and R was generated to calculate LL and RMSE and to put all the pieces together[2].

[1] Data can be obtained here: http://people.csail.mit.edu/zp/

[2] Matlab and R code will be available here: www.rinatrosenbergkima.com/AIED2015/

4 What are the Characteristics of the Real Datasets Parameters Space?

Before we explored the relationships between the real and sim datasets, we were interested to explore the BKT parameter profiles of the real datasets. We calculated the LL with a grid search of 0.04 granularity across the four parameters resulting in a maximum LL for each dataset and corresponding best prior, learn, guess, and slip. Figure 1 present the best parameters for each datasets, taking different views of the parameters space. The first observation to be made is that the best guess and slip parameters fell into two distinct areas (see figure 1, guess x slip).

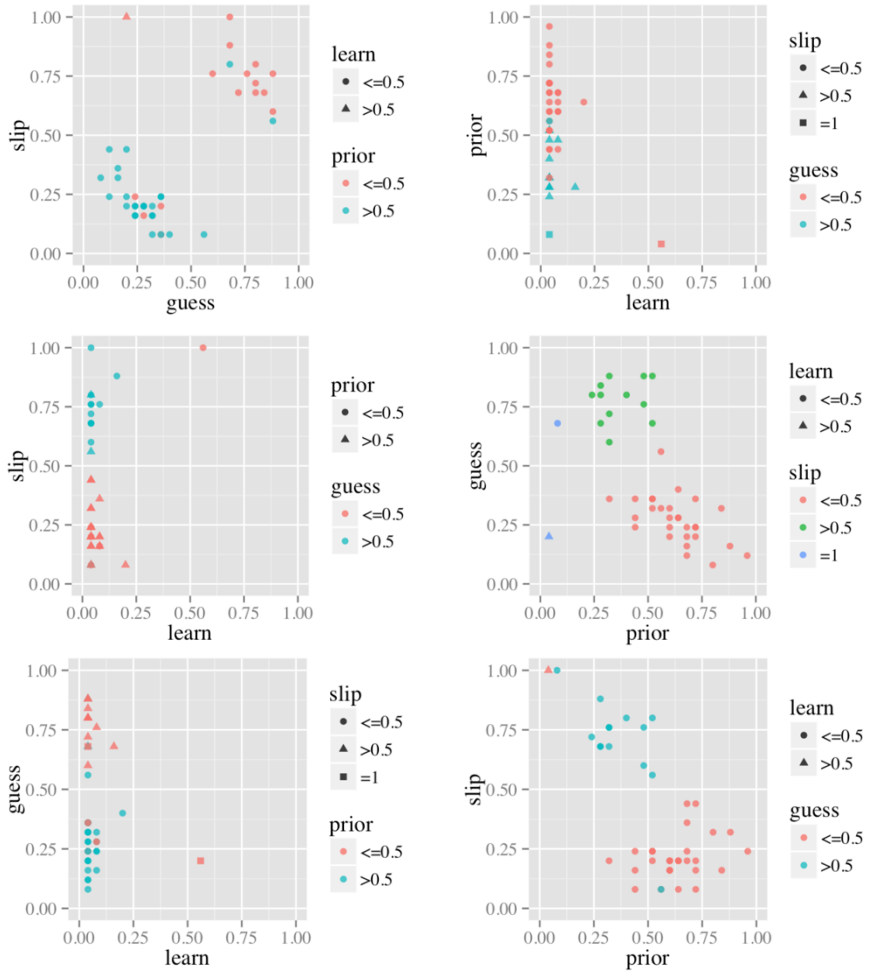

Figure 1. Best parameters across the 42 GLOP real datasets.

Much attention has been given to this LL space, which revealed the apparent co-linearity of BKT with two primary areas of convergence, the upper right area being a false, or "implausible" converging area as defined by [3]. What is interesting in this figure is that real data also converged to these two distinct areas. To further investigate this point, we looked for the relationships between the best parameters and the number of students in the dataset (see figure 2). We hypothesized that perhaps the upper right points were drawn from datasets with small number of students; nevertheless, as figure 2 reveals, that was not the case. Another interesting observation is that while in the upper right area (figure 1, guess x slip) most of the prior best values were smaller that 0.5, in the lower left area most of the prior best values were bigger than 0.5, thus revealing interrelationships between slip, guess, and prior that can be seen in the other views. Another observation is that while prior is widely distributed between 0 and 1, most of best learn values are smaller than 0.12.

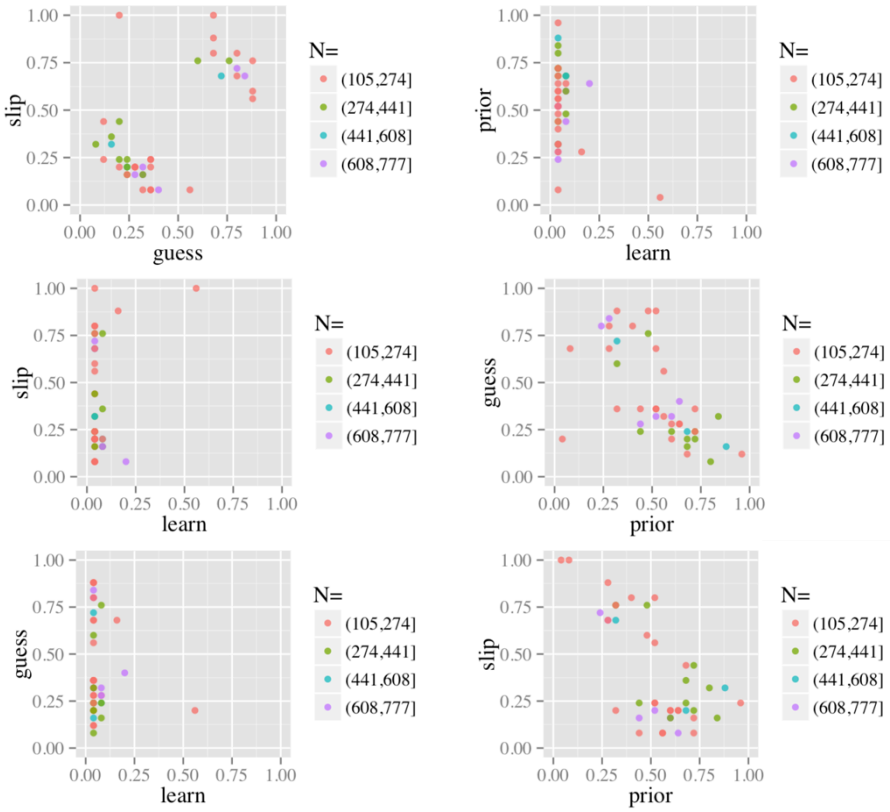

Figure 2. Best parameters across the 42 GLOP real datasets by number of students.

5 Does the LL of Sim vs. Real Datasets Look Different?

Our initial thinking was that as we are using a simple BKT model, it is not authentically reflecting reality in all its detail and therefore we will observe different patterns of LL across the parameters space between the real data and the simulated data. The LL space of simulated data in [5] was quite striking in its smooth surface but the appearance of real data was left as an open research question. First, we examined the best parameters spread across the 42 first set of simulated data we have generated. As can be seen in figure 3, the results are very similar (although not identical) to the results we received with the real data (see figure 1). This is not surprising, after all, the values of learn, prior, guess, and slip were inputs to the function generating the simulated data.

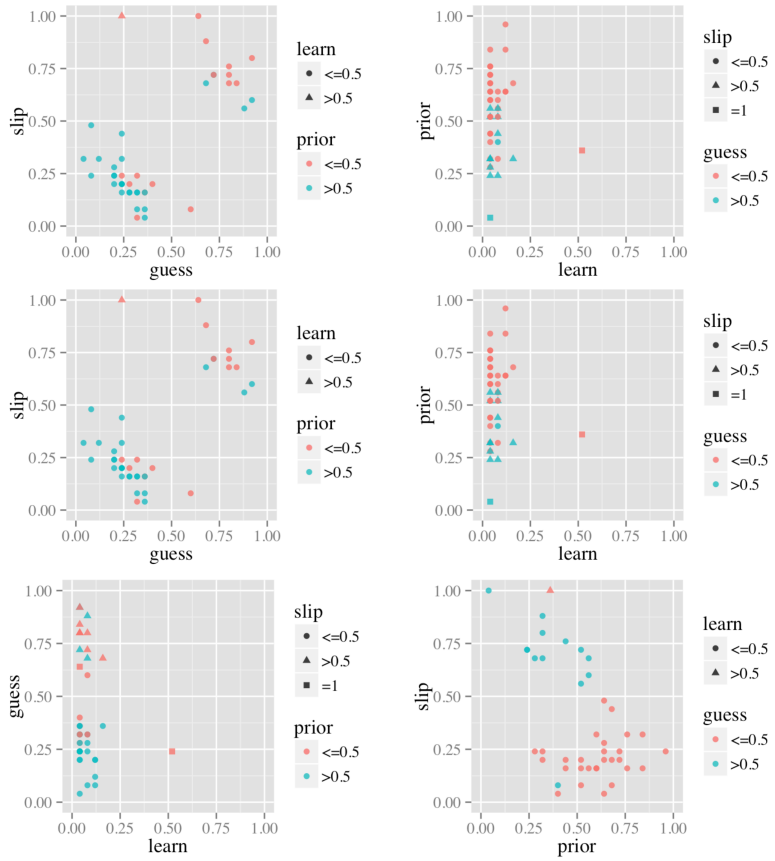

Figure 3. Best parameters across 42 GLOP simulated datasets.

In order to see if the differences between real and sim were more than just the difference between samples from the same distribution, we generated *two* simulated versions of each real dataset (sim1 and sim2) using the exact same number

of questions, number of students, generated with the best fitting parameters from the real dataset. We then visualized 2D LL heatmaps looking at two parameter plots at a time where the other two parameters were fixed to the best fitting values. For example, when visualizing LL heatmaps for the combination of guess and slip, we fixed learn and prior to be the best learn and the best prior from the real data grid search. To our surprise, when we plotted heatmaps of the LL matrices of the real data and the simulated data (the first column in figure 4 represents the real datasets, the second column represents the corresponding sim1, and the third column the corresponding sim2) we received what appears to be extremely similar heatmaps. Figure 4 and 5 displays a sample of 4 datasets, for each one displaying the real dataset heatmap and the corresponding two simulated datasets heatmaps.

The guess vs. slip heatmaps (see figure 4) prompted interesting observations. As mentioned above, the best guess and slip parameters across datasets fell into two areas (upper right and lower left). Interestingly, these two areas were also noticeable in the individual heatmaps. While in some of the datasets they were less clear (e.g., G5.198 in figure 4), most of the datasets appear to include two distinct global maxima areas. In some of the datasets the global maxima converged to the lower left expected area, as did the corresponding simulated datasets (e.g., G4.260 in figure 4), in other datasets the global maxima converged to the upper right "implausible" area, as did the corresponding simulated datasets (e.g., G6.208 in figure 4). Yet in some cases, one or more of the simulated dataset converged to a different area than that of the real dataset (e.g., G4.205 in figure 4). The fact that so many of the real datasets converged to the "implausible" area is surprising and may be due to small number of students or to other limitations of the model.

The learn vs. prior heatmaps were also extremely similar within datasets and exhibited a similar pattern also across datasets (see figure 5), although not all datasets had the exact pattern (e.g., G5.198 is quite different than the other 3 datasets in figure 5). While best learn values were low across the datasets, the values of best prior varied. As with guess vs. slip, in some cases the two simulated datasets were different (e.g., G4.205 had different best parameters also with respect to prior). Similar patterns of similarities within datasets and similarities with some clusters across datasets were also noticeable in the rest of the parameters space (learn vs. guess, learn vs. slip, prior vs. guess, prior vs. slip not displayed here due to space considerations).

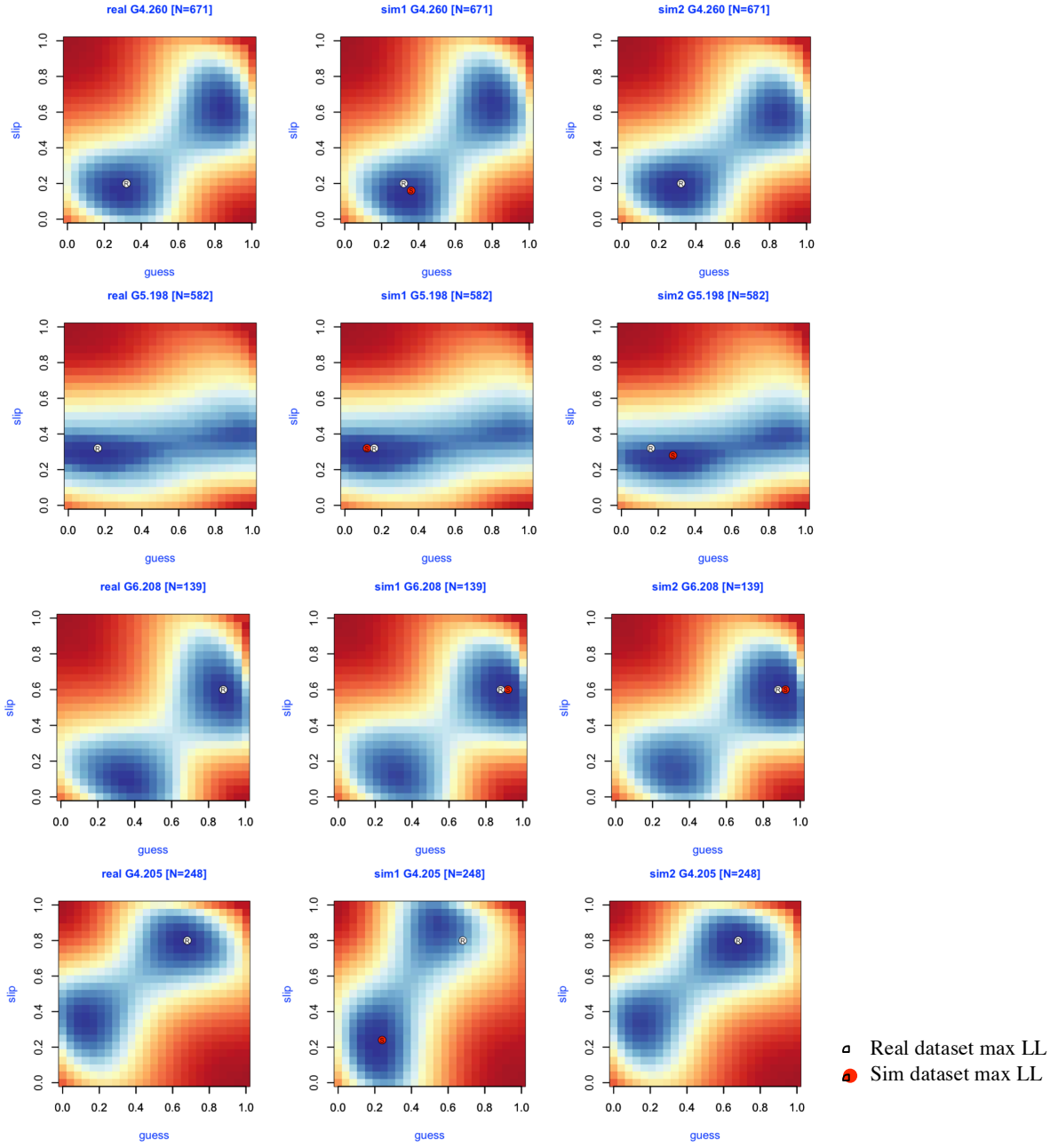

Figure 4. Heatmaps of (guess vs. slip) LL of 4 sample real GLOP datasets and the corresponding two simulated datasets that were generated with the best fitting parameters of the corresponding real dataset.

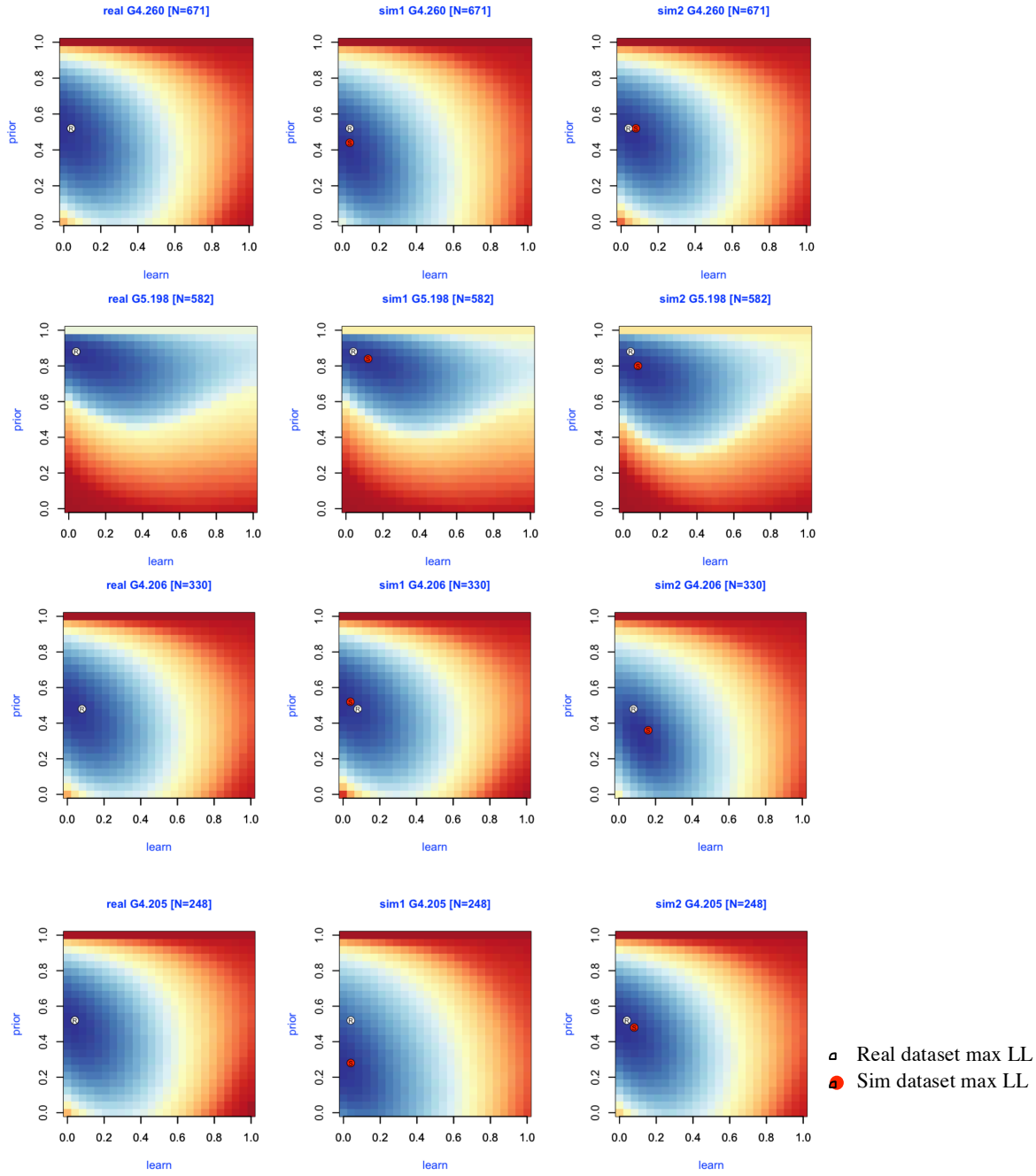

Figure 5. Heatmaps of (learn x prior) LL of 4 sample real GLOP datasets and the corresponding two simulated datasets that were generated with the best fitting parameters of the corresponding real dataset.

6 Exploring Possible Metrics Using the Real and Sim Datasets

In natural science domains, simulated data is often used as a mean to evaluate its underlying model. For example, simulated data is generated from a hypothesized model of the phenomena and if the simulated data appears to be similar to the real data observed in nature, it serves as evidence for the accuracy of the model. Then, if the underlying is validated, simulated data is used to make predictions (e.g., in the recent earthquake in Nepal a simulation was used to estimate the number of victims). Can this approach be used in education as well? What would be an indication of similarity between real and simulated data?

Figure 5 displays two preliminary approaches for comparing the level of similarity between the simulated and real data. First, the Euclidean distance between the real dataset parameters and the simulated data parameters was compared to the Euclidean distance between the two simulated datasets parameters. The idea is that if the difference between the two simulated datasets is smaller than the difference between the real and the simulated dataset this may be an indication that the model can be improved upon. Thus, points on the right side of the red diagonal indicate good fit of the model to the dataset. Interestingly, most of the points were on the diagonal and a few to the left of it. Likewise the max LL distance between the real and simulated datasets was compared to the max LL distance of the two simulated datasets. Interestingly, datasets with larger number of students did not result in higher similarity between the real and simulated dataset. Also, here we *did* find distribution of the points to the left and to the right of the diagonal.

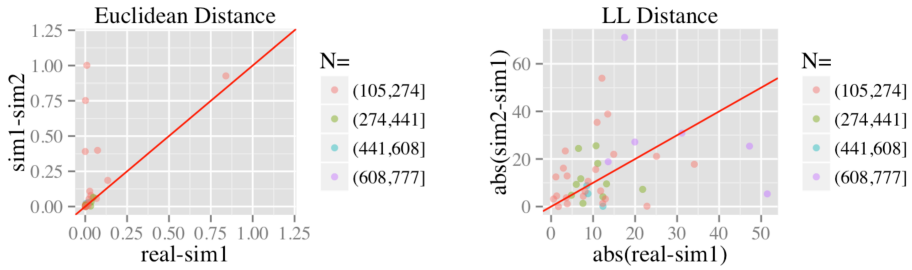

Figure 5. Using Euclidean distance and LL distance as means to evaluate the model.

7 Contribution

The initial motivation of this paper was to find whether it is possible to discern a real dataset from a simulated dataset. If for a given model it is possible to tell apart a simulated data from a real dataset then the authenticity of the model can be questioned. This line of thinking is in particular typical of simulation use in Science contexts, where different models are used to generate simulated data, and then if a simulated data has a good fit to the real phenomena at hand, then it may be possible to claim that the model provides an authentic explanation of the system [13]. We believe

that finding such a metric can serve as the foundation for evaluating the goodness of a model by comparing a simulated data from this model to real data and that such a metric could provide much needed substance in interpretation beyond that which is afforded by current RMSE and AUC measures. This can afford validation of the simulated data, which can than be used to make predictions on learning scenarios; decreasing the need to test them in reality, and at minimum, serving as an initial filter to different learning strategies.

References

[1] R. S. Baker and K. Yacef, "The state of educational data mining in 2009: A review and future visions," *J. Educ. Data Min.*, vol. 1, no. 1, pp. 3–17, 2009.

[2] M. C. Desmarais and I. Pelczer, "On the Faithfulness of Simulated Student Performance Data.," in *EDM*, 2010, pp. 21–30.

[3] J. E. Beck and K. Chang, "Identifiability: A fundamental problem of student modeling," in *User Modeling 2007*, Springer, 2007, pp. 137–146.

[4] Z. A. Pardos and M. V. Yudelson, "Towards Moment of Learning Accuracy," in *AIED 2013 Workshops Proceedings Volume 4*, 2013, p. 3.

[5] Z. A. Pardos and N. T. Heffernan, "Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence of the Expectation Maximization algorithm.," in *EDM*, 2010, pp. 161–170.

[6] R. B. Rosenberg-Kima and Z. Pardos, "Is this Data for Real?," in *Twenty Years of Knowledge Tracing Workshop*, London, UK, pp. 141–145.

[7] Z. A. Pardos and N. T. Heffernan, "Modeling individualization in a bayesian networks implementation of knowledge tracing," in *User Modeling, Adaptation, and Personalization*, Springer, 2010, pp. 255–266.

[8] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Model. User-Adapt. Interact.*, vol. 4, no. 4, pp. 253–278, 1994.

[9] S. Ritter, T. K. Harris, T. Nixon, D. Dickison, R. C. Murray, and B. Towle, "Reducing the Knowledge Tracing Space.," *Int. Work. Group Educ. Data Min.*, 2009.

[10] R. S. d Baker, A. T. Corbett, S. M. Gowda, A. Z. Wagner, B. A. MacLaren, L. R. Kauffman, A. P. Mitchell, and S. Giguere, "Contextual slip and prediction of student performance after use of an intelligent tutor," in *User Modeling, Adaptation, and Personalization*, Springer, 2010, pp. 52–63.

[11] R. S. Baker, A. T. Corbett, and V. Aleven, "More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing," in *Intelligent Tutoring Systems*, 2008, pp. 406–415.

[12] Z. A. Pardos and M. J. Johnson, "Scaling Cognitive Modeling to Massive Open Environments (in preparation)," *TOCHI Spec. Issue Learn. Scale*.

[13] U. Wilensky, "GasLab—an Extensible Modeling Toolkit for Connecting Micro- and Macro-properties of Gases," in *Modeling and simulation in science and mathematics education*, Springer, 1999, pp. 151–178.