The AAAI-17 Workshop on
Human-Machine Collaborative Learning
WS-17-11

Learning to Tutor from Expert Demonstrators via Apprenticeship Scheduling

Matthew Gombolay*
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, Massachusetts 02139
gombolay@csail.mit.edu

Reed Jensen, Jessica Stigile, Sung-Hyun Son
MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02420
{rjensen,jessica.stigile,sson}@ll.mit.edu

Julie Shah
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, Massachusetts 02139
julie_a_shah@csail.mit.edu

Abstract

We have conducted a study investigating the use of automated tutors for educating players in the context of serious gaming (i.e., game designed as a professional training tool). Historically, researchers and practitioners have developed automated tutors through a process of manually codifying domain knowledge and translating that into a human-interpretable format. This process is laborious and leaves much to be desired. Instead, we seek to apply novel machine learning techniques to, first, learn a model from domain experts' demonstrations how to solve such problems, and, second, use this model to teach novices how to think like experts. In this work, we present a study comparing the performance of an automated and a traditional, manually-constructed tutor. To our knowledge, this is the first investigation using learning from demonstration techniques to learn from experts and use that knowledge to teach novices.

Introduction

An increase in the sheer number and complexity of missile threats to national security have prompted researchers in the Department of Defense to develop innovative decision support tools that promote better decision-making for the warfighter. For the air and missile defense mission, initial research in this area began with simple Red/Blue wargaming exercises, where warfighters played against each other (i.e., red for offense and blue for defense) in order to solve challenging, unsolved tactical problems. Playing these games not only allowed the warfighter to discover and learn new tactics, techniques, and procedures, but also allowed the researchers to solicit feedback from the warfighter in order to refine the development of their decision support tools. While the data and feedback collected were invaluable, the training and educational aspects were static and limited by the sample size and update rate.

Limitations in conveying and collecting information across relevant sample sizes have motivated a data-driven, game-based simulation approach. For example, industry and academia alike are keenly interested in understanding player types and behaviors in games to better tailor the gameplay experience (Drachen et al. 2012; Nygren et al. 2011; Pirovano et al. 2012; Shaker, Yannakakis, and Togelius 2011; Thurau and Bauckhage 2010; van Lankveld et al. 2011). A key component of understanding player behavior is performance prediction. Performance prediction allows the educator to efficiently focus attention on those students who are struggling and need help. Further, performance prediction allows one to determine with less time spent on testing whether a student is actually proficient in a domain and ready to proceed to the next subject.

Still others within the field of education have thereby sought to develop methods for understanding why students, or players, drop out of educational programs (Clifton, Mandzuk, and Roberts 1994; Deslandes et al. 1999; Graunke and Woosley 2005; Mcinnis 2002). Students becoming disengaged in learning exercises is a chronic problem that greatly hampers the ability of educators to give students the tools they need to succeed (Clifton, Mandzuk, and Roberts 1994; Deslandes et al. 1999; Graunke and Woosley 2005; Mcinnis 2002). Researchers in artificial intelligence and machine learning have sought to develop methods for predicting student and player retention (Bauckhage et al. 2012; Mahlmann et al. 2010), which is a strong first step in correcting the problem of trainee dropout.

We have conducted a study investigating the use of automated tutors for educating players in the context of serious gaming (i.e., game designed as a professional training tool). Within the context of automated tutors, many researchers have sought methods to improve education (Albert and Thomas 2000; Kumar 2005; McLaren et al. 2004;

Distribution Statement: This material is based upon work supported by the Department of the Navy under Air Force Contract No. FA8721-05-C-0002 and/or FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Navy.

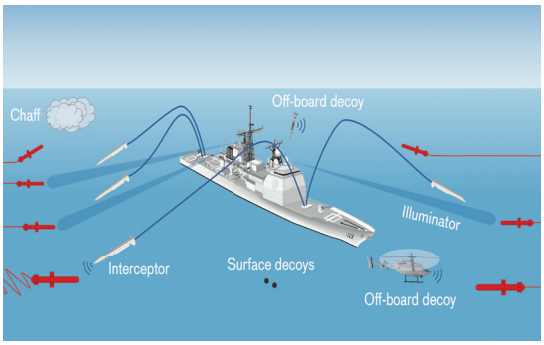


Figure 1: This figure depicts the problem of ASMD.

Mostow et al. 2003; Rahman, Sanghvi, and El-Moughny 2009; Remolina et al. 2009; that Listens 1997). Kumar proposes an automated tutor that can generate problems and answers to those problems; however, this tutor is manually programmed by an expert (Kumar 2005). Rahman et al. manually develop an automated tutor for learning Braille (Rahman, Sanghvi, and El-Moughny 2009). These techniques have in common that the tutor designer must solicit manually codify domain experts' knowledge within the tutor's software.

However, we know from prior work, that while domain experts are readily able to provide you with the important features of their problem, they are less able to tell you how they use those features to solve their problem. Thus, it is imperative we can learn from demonstration. One of the few works we are aware of that uses a learning from demonstration paradigms is that by McLaren et al., (McLaren et al. 2004). In their work, McLaren et al. propose a technique in which student interaction log data is used to create a skeleton model of a tutor for how to use a software tool. With this skeleton, users can improve the model with their own data or data of other users in a semi-autonomous fashion. However, this algorithm is only semi-autonomous, requiring an expert to manually parse data, incorporate new modules, and adjust the models representation.

We have developed an an automated tutor, Claire, which can fully autonomously learn an accurate model of how experts solve resource allocation problems and can use that knowledge to teach novices to do the same. Claire relies on a state-of-the-art technique in learning from demonstration, which we call apprenticeship scheduling (Gombolay et al. 2016). While this technique is useful for autonomous control, there will always be domains in which a human must be ultimately responsible for acting. These humans must be educated to make those decisions. Thus, extracting domain expert knowledge to teach novices to be proficient is critical. In this work, we present a study comparing the performance of an automated and manually-constructed tutor. To our knowledge, this is the first investigation using learning from demonstration techniques to learn from experts and use that knowledge to teach novices.

Problem Domain

For out investigation, we study the problem of anti-ship missile defense (ASMD), which is a complex variant of the weapon-to-target assignment problem (Lee, Su, and Lee 2003). As depicted in Figure 1, the problem of ASMD entails defending one's ship from a raid of enemy, anti-ship missiles through the use of hard- and soft-kill weapons. Hard-kill weapons (e.g., missile interceptors) disable enemy missiles with kinetic or chemical energy. On the other hand, soft-kill weapons (e.g., decoys and countermeasures) fall under the class of electronic warfare, in which the aim is to seduce, distract, or confuse enemy missiles. These soft-kill weapons mimic certain characteristics of naval vessels in order to cause the enemy missile to divert its attack away from those vessels. Because of the relatively high cost and limited availability of hard-kill weapons, the navy is particularly interested in the development of tactics for the use of soft-kill weapons.

The development of tactics for ASMD is extremely challenging. Operators may face situations where a single missile can only be defeated by the use of multiple, differing decoys. Alternatively, a single decoy may be able to defeat multiple enemy missiles. When faced with multiple missiles at the same time (i.e., a raid), countermeasures must be deployed to defeat all of those missiles, each of which may be using differing targeting characteristics. Further, the deployment of a single decoy could cause "fratricide," a condition in which the decoy may save one's own ship but now cause another ship to fall into harms way. AMSD falls into the hardest class of scheduling problems defined in the Korsah et al. taxonomy (Korsah, Stentz, and Dias 2013).

Investigative Platform

We have developed a game-based simulation, called Strike Group Defender (SGD), to emulate ASMD exercises. SGD, shown in Figure 2, provides users across a variety of locations and platforms with both single- and multi-player training experiences in the context of relevant naval scenarios. SGD collects participant actions and game events in order to analyze and refine the educational experience of the users either post hoc or in real time. The data-based collection capability of SGD has opened the way for the development of machine learning approaches that can analyze and improve the user educational experience.

In SGD, users must learn and employ the techniques and tactics relevant to the defense of naval assets against anti-ship missiles (hereafter referred to as ASMD). The game focuses on the proper use of naval electronic warfare – the use of signals instead of missiles for ship defense, otherwise known as *soft-kill* weapons (i.e., decoys) – but also includes *hard-kill* weapons (i.e., interceptor missiles) and information, surveillance, and reconnaissance (ISR) elements.

SGD is comprised of two level types: *training* and *test*. There is one training level for each enemy missile type to afford players an simple scenario to develop techniques to combat each of those missiles. There is also an introductory tutorial level and an tutorial exam. The tutorial levels are as follows: "Basics Tutorial", "Hungry Missile Tutorial",

Figure 2: SGD enables development of automated teaching tools for ASMD.

"Moth Missile Tutorial", "Catfish Missile Tutorial", "Longshot Missile Tutorial", "Weasel Missile Tutorial", "Muffled Missile Tutorial", "Headhunter Missile Tutorial", "Cerberus Missile Tutorial", and the "Tutorial Exam".

There are also three test levels: "Daily Performance Evaluation", "Operation Neptune", and "Final Countdown". Daily Performance Evaluation is a level where threat types are randomized, and threat bearings are spread across a range of angles such that it appears one's own ship is surrounded. The Daily Performance Evaluation provides a randomized threat scenario each time the user plays that level. Operation Neptune and Final Countdown are difficult, but deterministic, levels.

In each level, players assign and deploy soft-kill weapons (e.g., flare, chaff, etc.) to deceive or distract enemy missiles away from valuable ships. The proper coordination of soft-kill decoys with hard-kill interceptor missiles and ISR limitations ensures the long-term survivability of the ships in the strike group against a formidable raid of heterogeneous anti-ship cruise missiles.

Data Set

To train Claire's apprenticeship scheduling algorithm, we needed to collect a data set of human domain experts solving representative ASMD problems. To collect this data, we conducted a "March Madness" Tournament in which domain experts would compete in a bracket-based competition. Games were scored as follows: 10,000 points were received each time a threat was neutralized and 2 points for each second each threat spent homing in on a decoy. 5,000 points were subtracted for each threat impact and 1 points for each second each threat spent homing in on one's own ship. Lastly, 25-1,000 points were subtracted for each deploy of a decoy, depending on the type.

We focused on the Daily Performance Evaluation because it is randomized and is more well-suited to test the ability of a player to generalize learning as opposed to repeatedly playing a deterministic level or playing drastically different, deterministic levels. The collected data set consisted of 311 games played from 35 human players across 45 threat configurations or "scenarios" in this level. We sub-selected sixteen threat configurations such that each configuration had at least one human demonstration that mitigated all enemy missiles. For these sixteen threat configurations, there were 162 total games played by 27 unique human demonstrators. We then used the best human demonstration from each of the sixteen threat configurations to train the apprenticeship scheduling algorithm, which we will now describe.

Apprenticeship Scheduling

In this section, we review the apprenticeship scheduling algorithm developed by (Gombolay et al. 2016). This approach works by learning from demonstrations of human domain experts how to solve scheduling problems, such as ASMD. The approach has been shown suitable for learning high-quality policies from ASMD experts (Gombolay et al. 2016).

Consider an ASMD problem containing a set of enemy missiles $\mu \in M$, decoys $a \in A$, locations to deploy those decoys $x \in X$, as well as a set of actions taken at each moment in time $\tau_i = \langle \mu, a, x, t \rangle$. A trajectory given by a human domain expert demonstrator then provides a time-sequence of ordered actions. For each action the expert takes, we can compute the set of alternative actions τ_j the expert could have taken.

Each action, scheduled and unscheduled, has an associated real-valued feature vector, γ_{τ_i}. Features of this vector may include the time the decoy will evaporate, its bearing, etc. Further, there is a common feature vector, ξ_t, which captures features that are not well described by pairwise comparisons, such as the total number of decoys remaining.

$$
\begin{aligned}
{}^{priority}\theta^m_{\langle \tau_i, \tau_j \rangle} &:= \left[\xi_{\boldsymbol{\tau}}, \gamma_{\tau_i} - \gamma_{\tau_j}\right], y^m_{\langle \tau_i, \tau_j \rangle} = 1, \\
&\forall \tau_x \in \boldsymbol{\tau} \backslash \tau_i, \forall O_m \in \boldsymbol{O} | \tau_i \text{ scheduled in } O_m \qquad (1) \\
{}^{priority}\theta^m_{\langle \tau_j, \tau_i \rangle} &:= \left[\xi_{\boldsymbol{\tau}}, \gamma_{\tau_x} - \gamma_{\tau_i}\right], y^m_{\langle \tau_j, \tau_i \rangle} = 0, \\
&\forall \tau_x \in \boldsymbol{\tau} \backslash \tau_i, \forall O_m \in \boldsymbol{O} | \tau_i \text{ scheduled in } O_m \qquad (2)
\end{aligned}
$$

These vectors serve to create the training data, as shown in Equations 1-2. For each observation (i.e., a specific time point within a schedule), the apprentice scheduler creates a set of positive and negative examples. To create these examples, the apprentice scheduler subtracts the feature vector of the action not taken γ_{τ_j} from the feature vector describing the action taken, γ_{τ_i}. To this difference, the algorithm concatenates the common feature vector vector, $\xi_{\boldsymbol{\tau}}$. This concatenated vector then serves as a positive example. To create the corresponding negative example, the subtraction operation is reversed: The apprentice scheduler subtracts the feature vector of the action taken γ_{τ_i} from the feature vector describing the action not taken, γ_{τ_j}

With these examples, the apprentice scheduler trains a classifier $f_{priority}(\tau_i, \tau_j)$ to predict whether action τ_i or τ_j is better. The function returns a probability in $[0, 1]$. To predict which is the best overall action at a given moment in time, the apprentice scheduler evaluates Equation 3. The equation requires computing $|\boldsymbol{\tau}|^2$ comparisons. We apply a

decision tree classifier to learn $f_{priority}(\tau_i, \tau_j)$. In turn, the computational complexity of Equation 3 is $O(|\boldsymbol{\tau}|^2 d)$, where d is the depth of the tree.

$$\tau_i^* = \underset{\tau_i \in \boldsymbol{\tau}}{\operatorname{argmax}} \sum_{\tau_x \in \boldsymbol{\tau}} f_{priority}(\tau_i, \tau_x) \tag{3}$$

Claire: An Autonomous Tutor

We pose the challenge of developing an autonomous tutor that has the ability to receive questions from human students and autonomously generate answers to those questions in the context of serious games. In our approach, we develop an AI tutor by first, employing apprenticeship scheduling to learn from human domain expert example how to perform the intended task and, second, using the trained apprenticeship scheduler to tutor human students. We call our tutor "Clair,"[1].

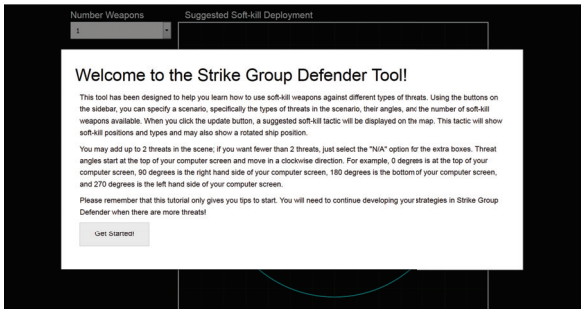
Students interact with Claire via a human-computer interface we developed to facilitate the tutoring session (Figure 3). A player might want assistance from Claire if, for example, he or she was unable to defeat a certain combination of enemy missiles. The player would construct a representative scenario specifying the number of enemy missiles, their types, and their bearings. In turn, Clair would use its apprenticeship scheduler to predict how a human expert would respond to those enemy missiles. Specifically, Claire would place a set of soft kill weapons of the type and bearing predicted by the apprenticeship scheduler.

To give Claire the ability to respond to the students' raid scenarios, we trained Clair on data from our March Madness tournament, which contains examples of human domain experts responding to representative ASMD raid scenarios. Specifically, we trained our model on a set of the best demonstrations of users playing the "Daily Performance Evaluation" level, which best captures the skills required for a robust policy. Robustness is important because we want Claire's recommendations to be applicable to a wide range of students' queries.

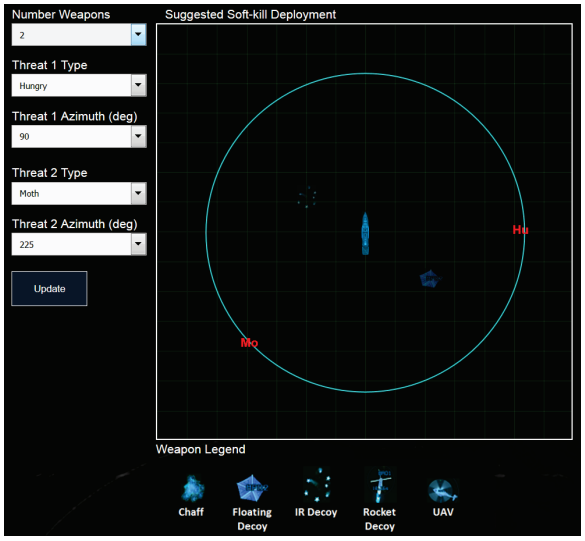
Human Benchmark

To validate the efficacy of our approach, we developed a second tutor, the "Human Tutor", based upon hard-coded responses by a human domain expert (i.e., one of the game's designers). Codifying rules for a raid of k missiles with n_t missile types from n_b bearings requires considering $O(\binom{n_t n_b}{k})$ scenarios, which grows to be intractable for a human demonstrator with even one or two missiles. Instead, we asked the human domain expert to develop a set of one-to-one rules (e.g., "deploy a flare ten degrees starboard if you see a moth missile"). These rules provide effective strategies for mitigating one threat with one decoy, but they do not scale well to handling multiple missiles with multiple decoys. The interaction effects between the missiles and decoys can cause these myopic rules to become quite suboptimal as more missiles and decoys are added to the environment.

[1]Claire's name is inspired by the word "clairvoyant."



(a) This figure depicts the welcome screen with instructions for participants.



(b) This figure depicts an example in which the participant asked Clair how to respond to two threats: a hungry missile and moth missile, at bearings 90° and 225°, respectively. Clair responded by deploying an IR flare at bearing -45°, which mitigates the moth missile, and chaff at 135°, which mitigates the hungry missile. Clair's response is a generalization of knowledge it learned from training the apprenticeship scheduling algorithm on data of human experts' demonstrations.

Figure 3: Figures 3a-3b depict the tutoring interface used for players in both the human tutor and computer tutor (i.e., Claire) condition. We note that participants in the human tutor condition would only be able to specify one missile threat because of the lack of scalability of the manually codified knowledge.

We embedded these rules in the same interface used by Claire (Figure 3), where users could similarly input an enemy raid and receive a recommendation based on the human experts' rules. In turn, the human tutor would respond with the relevant soft kill deployment and ship-turn movement using the codified rules. The Human Tutor interface was graphically identical to Clair with one exception: users could prescribe raids with up to only one missile. As we mentioned previously, enumerating a rule-set for n missiles was too time-consuming; even codifying rules for 2 missiles

with 5 missile types and 8 possible bearings would require considering 780 scenarios. While this difference introduces a possible experimental confound, we argue that the difference is inherent to comparing the efficacy of human and AI tutoring systems. Thus, this pilot study is a helpful first step in understanding the benefits of AI tutors for serious gaming.

Empirical Validation

We conducted a human-subject experiment, where users would have access to our computer and human tutors to augment their gameplay experience. Players were divided into two groups: one group would have access to the computer tutor (i.e., Claire), and one would have access to the human tutor. We asked players to explore their respective tutors for at least five minutes. Further, players were required to play SGD for at least 30 minutes. We hypothesize that our apprenticeship-scheduler-based computer tutor would provide players with more helpful instruction.

We report the result of statistical testing of our hypothesis. For the Daily Performance Evaluation (Figure 4), the level used as the basis for training our computer tutor, players using the computer tutor performed better (n = 14, M = $56,204$, SD = $\pm$ $29,408$) than players in human-tutor condition (n = 17, M = $41,639$, SD = $\pm$ $22,870$), $p = 0.132$. We perform the same analysis on the players overall tournament score (Figure 5), and found that players with the computer tutor performed better (n = 10, M = $231,280$, SD = $\pm$ $99,729$) than players in human-tutor condition (n = 11, M = $228,811$, SD = $\pm$ $81,938$), $p = 0.95$.

While the p-value is not statistically significant at the $\alpha = 0.05$ level, there is a strong indication for the Daily Performance Evaluation ($p = 0.132$) that our computer tutor offers players more benefit than the human tutor. As such, we propose conducting a second data collection phase to ascertain definitive results. Further, we hypothesize that training on a further variety of levels could yield improved benefit for results other than the Daily Performance Evaluation. Conducting a power analysis for two independent samples where we estimate the common standard deviation to be $26,609$ by pooling the variance, the mean performance for the computer tutor is 56,204, and the mean performance for the human tutor is $41,639$, we find we need 42 people per condition to have an 80% power for finding a statistically significant difference at the $\alpha = 0.05$ confidence level for the Daily Performance Evaluation. In future work, we propose collecting more data to establish statistical significance.

Conclusion and Future Work

We propose a computer-based tutoring system that learns a model for teaching pupils based on data of expert demonstrators. Our system, Clair, employs a state-of-the-art technique in apprenticeship scheduling that learns how to solve resource allocation problems from expert demonstration. Clair then uses this knowledge to serve as an autonomous tutor who can help solve a student's problem when that student becomes stuck. We conduct an initial pilot study examining the effectiveness of our technique. We found that

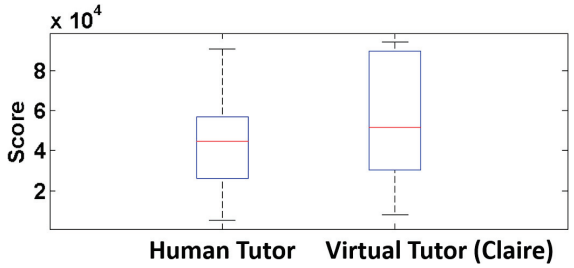


Figure 4: This figure depicts a histogram of scores players assigned to the human tutor and computer tutor conditions achieved the first time they played the Daily Performance Evaluation level.

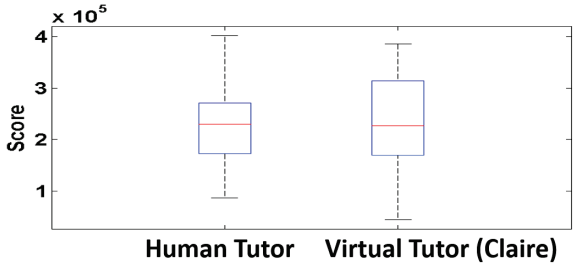


Figure 5: This figure depicts a histogram of tournament score for players assigned to the human tutor and computer tutor conditions. The tournament score is the sum of the average Daily Performance Evaluation score, the maximum Operation Neptune score, and the Final Countdown score.

while students prefer a traditional human tutor, their performance improved more when using Clair. In future work, we will conduct a more robust study to isolate the causal link in performance change and better understand the apparent contradiction between students' perception and reality of Clair's benefits.

References

Albert, S., and Thomas, C. 2000. A new approach to computer-aided distance learning: the'automated tutor'. *Open Learning* 15(2):141–150.

Bauckhage, C.; Kersting, K.; Sifa, R.; Thurau, C.; Drachen, A.; and Canossa, A. 2012. How players lose interest in playing a game: An empirical study based on distributions of total playing times. In *CIG*, 139–146.

Clifton, R. A.; Mandzuk, D.; and Roberts, L. W. 1994. The alienation of undergraduate education students: a case study of a canadian university. *Journal of Education for Teaching* 20(2):179–192.

Deslandes, R.; Royer, .; Potvin, P.; and Leclerc, D. 1999. Patterns of home and school partnership for general and special education students at the secondary level. *Exceptional Children* 65(4):496–506.

Drachen, A.; Sifa, R.; Bauckhage, C.; and Thurau, C. 2012.

Guns, swords and data: Clustering of player behavior in computer games in the wild. In *CIG*, 163–170.

Gombolay, M.; Jensen, R.; Stigile, J.; Son, S.-H.; and Shah, J. 2016. Decision-making authority, team efficiency and human worker satisfaction in mixed human-robot teams. In *Proc. IJCAI*.

Graunke, S. S., and Woosley, S. A. 2005. An exploration of the factors that affect the academic success of college sophomores. *College Student Journal* 39(2):367 – 376.

Korsah, G. A.; Stentz, A.; and Dias, M. B. 2013. A comprehensive taxonomy for multi-robot task allocation. *IJRR* 32(12):1495–1512.

Kumar, A. N. 2005. Generation of problems, answers, grade, and feedback—case study of a fully automated tutor. *Journal on Educational Resources in Computing (JERIC)* 5(3):3.

Lee, Z.-J.; Su, S.-F.; and Lee, C.-Y. 2003. Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics. *IEEE Trans. SMC Part B* 33(1):113–121.

Mahlmann, T.; Drachen, A.; Togelius, J.; Canossa, A.; and Yannakakis, G. 2010. Predicting player behavior in tomb raider: Underworld. In *CIG*, 178–185.

Mcinnis, C. 2002. Signs of disengagement? In Enders, J., and Fulton, O., eds., *Higher Education in a Globalising World*, volume 1 of *Higher Education Dynamics*. Springer Netherlands. 175–189.

McLaren, B. M.; Koedinger, K. R.; Schneider, M.; Harrer, A.; and Bollen, L. 2004. Bootstrapping novice data: Semi-automated tutor authoring using student log files.

Mostow, J.; Burkhead, P.; Corbett, A.; Cuneo, A.; Eitelman, S.; Huang, C.; Junker, B.; Sklar, M. B.; Tobin, B.; et al. 2003. Evaluation of an automated reading tutor that listens: Comparison to human tutoring and classroom instruction. *Journal of Educational Computing Research* 29(1):61–117.

Nygren, N.; Denzinger, J.; Stephenson, B.; and Aycock, J. 2011. User-preference-based automated level generation for platform games. In *Computational Intelligence and Games*, 55–62.

Pirovano, M.; Mainetti, R.; Baud-Bovy, G.; Lanzi, P.; and Borghese, N. A. 2012. Self-adaptive games for rehabilitation at home. In *CIG*, 179–186.

Rahman, M. K.; Sanghvi, S.; and El-Moughny, N. 2009. Enhancing an automated braille writing tutor. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2327–2333. IEEE.

Remolina, E.; Ramachandran, S.; Stottler, R.; and Davis, A. 2009. Rehearsing naval tactical situations using simulated teammates and an automated tutor. *IEEE Transactions on Learning Technologies* 2(2):148–156.

Shaker, N.; Yannakakis, G.; and Togelius, J. 2011. Feature analysis for modeling game content quality. In *Proc. CIG*, 126–133.

that Listens, R. T. 1997. The sounds of silence: towards automated evaluation of student learning in a reading tutor that listens. In *AAAI-97: Proceedings of the Fourteenth National Conference on Artificial Intelligence and The Ninth Annual Conference on Innovative Applications of Artificial Intelligence*, volume 14, 355. AAAI Press.

Thurau, C., and Bauckhage, C. 2010. Analyzing the evolution of social groups in world of warcraft. In *CIG*, 170–177.

van Lankveld, G.; Spronck, P.; Van den Herik, J.; and Arntz, A. 2011. Games as personality profiling tools. In *CIG*, 197–202.