Integrated Introspective Case-Based Reasoning for Intelligent Tutoring Systems

Leen-Kiat Soh

Computer Science and Engineering, University of Nebraska
256 Avery Hall
Lincoln, NE 68588-0115 USA
lksoh@cse.unl.edu

Abstract

Many intelligent tutoring systems (ITSs) have been developed, deployed, assessed, and proven to facilitate learning. However, most of these systems do not generally adapt to new circumstances, do not self-evaluate and self-configure their own strategies, and do not monitor the usage history of the learning content being delivered or presented to the students. These shortcomings force ITS developers to often spend much development time in manual revision and fine-tuning of the learning and instructional contents of an ITS. In this paper, we describe an intelligent agent that delivers learning material adaptively to different students, factoring in the usage history of the learning materials and student profiles as observed by the agent. Student-tutor interaction includes the activities of going through learning material, such as a topical tutorial, a set of examples, and a set of problems. Our assumption is that our agent will be able to capture and utilize these student activities as the primer to select the appropriate examples or problems to administer to the student. Using an *integrated introspective case-based reasoning* approach, our agent further learns from its experience and refines its reasoning process—including the instructional strategies—to adapt to student needs. Moreover, our agent monitors the usage history of the learning materials to improve its performance. We have built an end-to-end ITS using an agent powered by this integrated introspective case-based reasoning engine. We have deployed the ITS in a CS course. Results indicate that the ITS was able to learn to deliver more appropriate examples and problems to the students.

Introduction

Intelligent tutoring systems (ITSs) adapt instruction, examples, problems, and feedback to the needs or state of knowledge of individual students given the tutoring situation that the students are in. As summarized in (Graesser *et al.* 2001), ITSs are clearly one of the successful outcomes of Artificial Intelligence (AI) research. Many intelligent tutoring systems have been developed, deployed, assessed, and proven to facilitate learning (Graesser *et al.* 2001). However, most of these systems do not generally adapt to new circumstances, do not self-evaluate and self-configure their own strategies, and do not monitor the usage history of the learning content being delivered or presented to the students. Further, these systems do not tag the *quality* of the learning content based on how successfully a module has been used and how it has matched the expected objectives. These shortcomings force ITS developers to often spend much development time in manual revision and fine-tuning of the learning and instructional contents. (Here we define learning content as tutorial material such as an example, a problem, or an explanation for a particular topic. We define instructional content as the instructional strategies—the tutoring heuristics or expertise—that determine which examples or problems should be delivered to the students, how to scaffold the explanations appropriately, and so on.) As a result, in the vast majority of ITS applications, examples, problems, and instructional strategies must be authored, evaluated, and modified manually. To automate the revision process, the selection, sequencing, and adaptation of examples and problems are thus a key research problem in ITSs. Since examples and questions can vary according to many dimensions, e.g., difficulty, cognitive level, types of verbiage, context and length, and students can vary in aptitude, motivation, and self-efficacy, it is challenging for an ITS to be able to learn to improve its own performance over time.

In this paper, we describe an intelligent tutoring system called the Intelligent Learning Material Delivery Agent (ILMDA). This system integrates intelligent techniques from the research areas of agents, case-based reasoning, and machine learning. It delivers learning material based on the usage history of the learning material, the student relatively static background profile (such as GPA, majors, interests, and courses taken), and the student dynamic activity profile (based on their interactions with the ITS). The agent uses the profiles to decide, through case-based reasoning (CBR) (Kolodner 1993), which learning modules (examples and problems) to present and how to present them to the students. Our CBR treats the usage history and the profiles as the input *situation* of a case, and the output *solution* of a case is basically the specification of an appro-

priate example or problem. Our agent also uses the usage history of each learning material to adjust the appropriateness of the examples and problems in a particular situation. Each case is thus an instructional strategy.

We have also devised an *introspective* CBR engine that incorporates elements of machine learning such that the agent examines its similarity-based retrieval, adaptation, and case learning processes: heuristics or weights associated with an instructional strategy (a case) are reinforced according to the outcome of applying that case. To integrate the impacts of the three learning mechanisms on the agent's casebase and reasoning, we have also used a principled approach to minimize the "canceling out" effects on the learned knowledge. This allows the ITS itself to gradually adjusts its instructional strategies and also to tag the quality of the learning material. We have deployed ILMDA in a CS1 (first CS core course). Results show that ILMDA is able to refine its instructional strategies to improve outcomes and pedagogy.

Related Work

There have been successful ITSs such as as PACT (Koedinger et al. 1997), ANDES (Gertner and VanLehn 2000), AutoTutor (Graesser et al. 2001), and SAM (Cassell et al. 2000). Some have the capabilities to classify learner engagement (Beal et al. 2006), detect student motivation and proficiency (Johns and Woolf 2006), and recognize goals (Mott et al. 2006) by tracking and modeling real-time interactive data through "machine learning"-like steps. However, these systems do not have "introspective" capabilities to modify, for example, possibly *weights* on the multiple data sources during integration in (Beal et al. 2006), or the different mixture models in (Johns and Woolf 2006). That is, these systems do not self-evaluate and self-configure their own strategies to improve their own performance over time. ILMDA is different; it learns how to deliver appropriate learning content to different types of students and to monitor and evaluate how the learning materials are received by the students.

The notion of instrospective CBR has also been proposed with the emphasis on feature weighting for similarity-based case retrieval. For example, Wettschereck and Aha (1995) proposed weighting features automatically for case retrieval using a hill-climbing algorithm; Cardie (1999) used cognitive biases to modify feature set selection (changing, deleting, and weighting features appropriately); Bonzano et al. (1997) used a decay policy together with a push-pull perspective to adjust the term weights; Avesani et al. (1998) used reinforcement learning to reward nearest neighbors that can be used correctly to solve input problems to adapt the local weights to the input space; Jarmulak et al. (2000) used genetic algorithms to determine the relevance or importance of case features and to find optimal retrieval parameters; Zhang and Yang (2001) used quantitative introspective learning resembling back-propagation neural networks to learn feature weights of cases; Park and Han (2002) used an analogical reasoning structure for feature weighting using a new framework called the analytic hierarchy process; and Patterson et al. (2002) proposed a hybrid approach based on the k-nearest neighbor algorithm and regression analysis. While case learning has also been a staple of CBR (e.g., Watson and Marir 1994), learning about case adaptation has not received as much attention. Leake et al. (1995) formulated the task of acquiring case adaptation knowledge as learning the transformation and memory search knowledge, by learning successful adaptation cases for future use. In Stahl (2005)'s formal view of a generalized CBR model, the author considered learning similarity measures while assuming that the adaptation and output function remain static during the lifetime of the CBR system. However, none of the above approaches proposed an integrated, introspective machine learning framework for CBR that learns cases, revises similarity weights, and revises adaptation heuristics at the same time.

An important and relevant work in introspective learning is the Meta-AQUA system (Cox and Ram 1999) that learns to improve its story-understanding performance through analysis of its own reasoning. The system uses cases to tie explanation-patterns. The learning is separated into three phases: (1) failure explanation, (2) learning decision (what to learn), and (3) strategy construction (how to learn). In our work, there is no failure explanation as the domain knowledge is not rich or complete enough to be able to pinpoint the causes for failure in, say, a tutoring session, especially when there are external factors that ITSs of today still cannot track or observe and consider.

Integrated Introspective CBR Framework

The integrated introspective CBR framework learns new cases, learns about how to determine case similarity (feature weights), and about how to adapt previous solutions to new situations. This machine learning is driven by feedback provided by the environment about how well the solution of a case has worked.

In this framework for our intelligent tutoring system ILMDA, a case consists of a *situation* and a *solution*. The situation describes the learning material that a student has just viewed, the student's relatively static background profile, and the student's dynamic profile. The set of parameters used to describe the learning material includes the average amount of time spent on the learning material, the average number of examples viewed for this particular topic, and the number of going back-and-forth between an example and the tutorial, and between a question and an example, and between a question and a tutorial.

The student static and dynamic profiles form the basis for the ILMDA's *learner* model. A learner model tells us the metacognitive level of a student by looking at the student's behavior as he or she interacts with the instructional content set. The background of a student stays relatively static and consists of the student's major, GPA, goals, affiliations, aptitudes, and competencies. It also includes

self-reported self-efficacy and motivation, based on a survey taken before a student processes a content set. The dynamic student profile captures the student's real-time behavior and patterns. It consists of the student's online interactions with ILMDA including the number of attempts on the same item, number of different modules taken so far, average number of mouse clicks during the tutorial, average number of mouse clicks viewing the examples, average length of time spent during the tutorial, number of quits after tutorial, number of successes, and so on.

Given the situation, a solution specifies the characteristics of the most appropriate examples or problems to be delivered next to the student. The characteristics include the level of difficulty, the cognitive level in terms of Bloom's taxonomy (Bloom et al. 1964) (i.e., recall, comprehension, application, analysis, evaluation, and synthesis), the level of scaffolding (whether to provide highlights, hints, references or additional explanations), length (in terms of text length and average amount of time taken by students to view a content set), and the amount of interactions (i.e., the average number of clicks by students when viewing a content set).

Thus, in ILMDA, a case is an instructional strategy mapping a tutoring situation that takes into account the learning material and the student profile to a decision on what the characteristics of the next appropriate example or problem should be.

Case Learning Module

The case learning module is typical. It allows the system to learn new, different cases in order to build a more robust casebase. It compares the situation description (i.e., the input space of a case) with the currently stored cases. If the new case' situation is distinct enough from the other cases', the new case is stored in the casebase, diversifying the casebase in order to cover more situations. The system learns new cases in order to allow the system to more easily handle a wider range of situations that it will inevitably encounter when faced with different types of students.

Similarity Weights Learning Module

This learning module uses the outcome to analyze the true similarity of two given problems in order to re-weight the case features (i.e., situation parameters in our design). It is built on several base assumptions: (1) similar cases will have similar solutions; and (2) if a solution has been generally successful, and the system believes that if a new situation is similar to the situation to which this solution has been applied successfully, then the solution should have been successful for this problem. If the situations are deemed to be truly similar, then the system will reward the feature (parameter) weights that most strongly influenced the similarity. If they are not deemed similar, then those strongly influencing weights will be penalized. The system analyzes several iterations worth of adaptations at once, in order to have an accurate depiction of how each case has performed and an accurate assessment of the true similarity. Adjusting these weights allows the system to value the similarity of certain situation parameters more heavily than other parameters. For instance, if the system deemed that a situation parameter *a* was the most influential in determining that case *x* and case *y* were very similar, but the cases had very different performances, then the learning module will adjust the similarity weights to make *a* less influential, thereby making cases *x* and *y* less similar. However, if our similarity measurement did not determine that the cases were similar, then we do not adjust the weights as the cases may or may not perform similarly.

Adaptation Heuristics Learning Module

This module revises how the system adapts old solutions to new situations. The system analyzes past adaptations, including (1) the new situation specification it was given, (2) the case from the casebase that was used, (3) the adapted solution that was reached, and (4) the outcome that resulted from this solution. It determines which adaptation heuristic was the cause of a good or bad adaptation and adjusts the heuristics accordingly. The adaptation heuristics in our system modify the retrieved solution to account for differences in the parameters described in the situation space. Similar to the previous module, if an adaptation is deemed successful—i.e., the adapted solution leads to a successful outcome, then the heuristics that were influential in making that adaptation will be reinforced, and vice versa.

Integrated Learning

This framework presents several advantages and disadvantages over a static CBR system. One key advantage is that no one module is entirely responsible for improving the performance of the system. However, this also exposes a potential weakness of our approach. The system may never converge on a steady state of knowledge as one module improving may undo the improvements made by another module. For example, adjusting the similarity weights affects how the system decides which new cases to learn and which best case to retrieve and adapt. Learning new cases reduces the need for the adaptation module to make radical, far reaching heuristics, and allows for more choices when the similarity module attempts to find a case to use. Adjusting the adaptation heuristics will reduce the need for the casebase to cover *every* situation. To minimize the "canceling each other out" effects, we have devised a set of principles to guide the design of the integrated approach:

Principle 1. When the framework learns a new case, its objective is to expand the situation space but not the solution space of the casebase. With this arrangement, while it is still possible for the solution coverage of the casebase to expand, it is the adaptation heuristics that have to shoulder the task of expanding the solution space.

Principle 2. The framework may also learn new cases with failed solutions as the CBR system also performs failure-driven adaptations. This means if the same situation arises in the future, it is possible for the system to retrieve a newly-learned case with a failed solution, and adapt *away*

from the failed solution, to incrementally and eventually obtain a successful solution without straining the adaptation heuristics.

Principle 3. Each case is tagged with a utility or competence vector that records how successful the case has been used, how often the case has been retrieved, and how many new cases has been spawned as a result of the retrieval of this case (Soh and Luo 2004). Cases that have been more successful will carry more weight, for example, in changing the similarity weights. Cases that have spawned more new cases will carry more weight in determining the adaptation heuristics.

Principle 4. The framework has tuning parameters for each learning module that specify the amount of change in weights used in reinforcement. This allows the developer to inject his or her confidence in the instructional content. For example, if the similarity weights—i.e., the relative importance of the features—are highly regarded to be correct, then the amount of changes allowed on these weights can be set to be small. This in turn facilitates the overall coherence or convergence of the various learning modules.

Principle 5. The framework learns conservatively—changing only the most influential similarity weights and adaptation heuristics for each learning episode. That is, after the blame or credit assignment, the similarity weights or adaptation heuristics are ranked in terms of the blame or credit. Only the top contributor to a failure or a success is penalized or rewarded accordingly. This has the effect of reducing the impact of a single outcome on the weights or heuristics, allowing the system to gradually adjust the impact of each weight or heuristic.

Principle 6. The framework staggers its learning activities with different activation frequencies. For example, case learning can be activated every time a new case is evaluated while weights or heuristics learning can be activated after every *n* cases, or only when the system has seen too many failures for some period of time.

We have implemented our integrated introspective CBR system following all the above principles. We have adopted Principles 1 and 2 in the case learning module, Principles 3, 4, and 5 in the similarity weights and adaptation heuristics learning modules, and Principle 6 in the overall learning management.

Implementation and Deployment

We have implemented the integrated introspective CBR framework in ILMDA. As alluded to earlier, ILMDA is designed to deliver the appropriate tutorial, examples, and exercises to the students based on student aptitudes and backgrounds, their interactions with the system, and the learning content. For each session, the CBR module has to decide what examples or problems to deliver to the student and how to present these examples or problems (such as the amount of scaffolding) for ILMDA. Figure 1 shows the flowchart of ILMDA.

When a student starts the ILMDA application, he or she is first asked to login. This associates the user with his or her profile information. After a student is logged in, he or she selects a topic and then views the tutorial on that topic. Following the tutorial, the agent composes a *new situation* from (1) the student's static profile, (2) the dynamically tracked activities the student performed when viewing the tutorial, and (3) the parameters of the tutorial viewed. The agent subsequently uses this new situation to retrieve the *best case*—i.e., the case with the most similar situation to the new situation—from the casebase. The agent then adapts the solution of that similar case depending on how the cases differ, and uses the adapted output to search for a suitable example (with the appropriate level of difficulty, scaffolding, Bloom's level, length, and so on) to give to the student. After the student is done looking at the examples, the same process is used to select an appropriate problem. Again, the agent takes into account how the student behaved during the example, as well as his or her background profile. After the student completes an example or problem, he or she may choose to be given another and the process repeats. We deem an instructional strategy (or a case) as successful if a student behaves within the normal (or Gaussian) range of the expected behavior associated with each example or problem. Also, if a student elects to view another example after viewing one, that also indicates that the first example might not have been appropriate. If a student *quits*—exiting out of the session—before finishing an example or answering a question, that also indicates that the example or question might not have been appropriate.

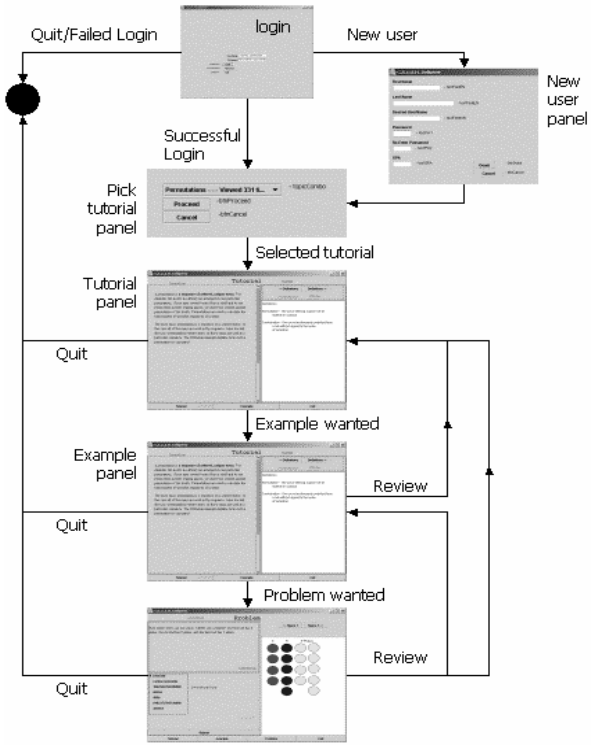

Figure 1. GUI and interactions between ILMDA and students.

We deployed ILMDA to the first core course for the computer science and computer engineering majors (CS1) at the University of Nebraska. Typically, this course has

about 100 students per year, with a diverse group of students from majors such as CS, Math, Engineering, and so on. Further, the programming backgrounds of these students are very diverse. Some incoming freshmen have had some programming in their high schools; some have had none. Thus, it is important for the course to be able to adapt to the different student aptitude levels and motivations. For details on the deployment, a simulator built to analyze the ITS, and a case study on a Recursion topic, please refer to (Blank 2005), (Soh and Blank 2005), (Soh et al. 2005), respectively. In this paper, we focus on the comparison of two versions of ILMDA: learning, and non-learning. The non-learning agent performs CBR without any learning. The learning ILMDA enables all its CBR and machine learning modules. Here is a report on some key results briefly: how the introspective CBR impacted the outcomes of the tutoring system and how it revised the instructional content: cases, similarity weights, and the adaptation heuristics.

Impact on Outcome

One metric to measure the impact of integrated introspective CBR is the average number of problems delivered between a wrong answer and solving a problem correctly. This metric indicates how effective and efficient the system is in adjusting to an unsuccessful session—giving a problem that a student answered incorrectly—to eventually locate a problem that the student is able to answer correctly. Though given the same casebase to begin with, the learning agent outperformed the non-learning agent on the average number of problems delivered between an incorrectly answered problem and a correctly answered one (0.93 vs. 1.72). This indicates that the net learning gain was positive for the integrated introspective CBR framework.

Further, the learning agent was able to increase the utility of the instructional content of ILMDA, as shown in Figure 3. The utility of the instructional content is the sum of (1) the utility of each similarity weight in retrieving a successfully applied case; (2) the utility of each adaptation heuristic in contributing significantly to an adaptation of the old solution to the new situation that led to the successful application of the adapted solution; and (3) the utility of each case as the ratio of the number of successful applications of the case over the number times the case has been retrieved. As shown in Figure 3, there is a significant improvement from the first time period to the second time period, and then there is a slow decline. However, even with the slight decline in outcome, the final content set still outperformed the first one significantly. This graph shows the ability of the learning agent to improve the utility of the instructional content set.

Impact on Instructional Strategies

Here we briefly summarize how integrated introspective CBR has allowed us to evaluate, refine, and discover instructional strategies on effective delivery of learning materials.

For example, the initial instructional content assumed that GPA was an important indicator in deciding the next appropriate example or problem, and that the "frequent going back-and-forth" behavior of a student was a key indicator that a student is trying to look for answers to problems without actually reading through the learning content carefully in the first place, as encoded in the cases. However, ILMDA learned that (1) GPA was not an important situation feature (its importance dropped from 1.0 to 0.46) and (2) the number of times a student went back-and-forth between an example and the tutorial was quite important (0.80) but not as important as the back-and-forth between a problem and the tutorial (0.99).

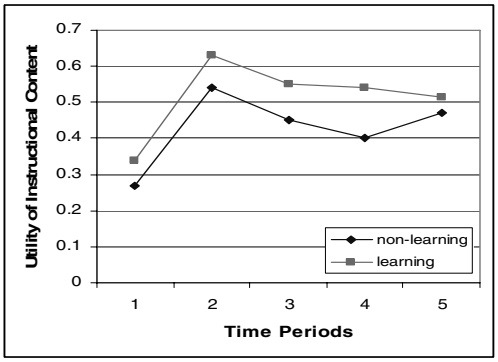

Figure 3. Utility of instructional content set over the course of five time periods ILMDA was used.

Another example involves the adjustment of an adaptation heuristic. Originally, there was a heuristic that increased the degree of scaffolding when presenting an example to the students if the new situation showed that the student failed to answer problems correctly more often than the student profile in the situation of the best case. However, the learning ILMDA learned to reduce the degree of scaffolding to better guide a student to attempt to answer a problem. ILMDA observed that when students were found to spend more time on examples, or were found to refer back to the examples, they were more likely to quit ILMDA before answering questions correctly. Thus, ILMDA cut down the amount of scaffolding to reduce the length of the examples delivered.

These examples show that, using the introspective CBR engine, an ITS can refine its own instructional strategies to better adapt to the students that it deals with and the learning content that it has.

Conclusions

We have built an intelligent tutoring system called ILMDA with the underlying reasoning being the proposed integrated introspective case-based reasoning. This approach integrates case learning, revision of similarity weights and revision of adaptation heuristics such that the ITS can reward and penalize the cases and heuristics according to the outcomes of their usages. To minimize the "canceling out"

effects, we have also used a set of principles to facilitate concurrent learning modes. Together, this approach allows the agent to refine its instructional content. Our experiments, based on an actual deployment in an introductory CS course, indicated that the ILMDA learning agent was able to learn to better deliver more appropriate examples and problems and refine its instructional strategies (cases, similarity weights, and adaptation heuristics), making it more adaptive to different learner models topics.

For our future work, we plan to study how the three learning modules interact and how each principle impacts the overall learning behavior. We also plan to refine ILMDA to make it a testbed to quality tag the learning content. Currently, it is able to reduce the weights on the learning content (as situation parameters in a case), giving an overall picture of how consistent or accurate the collective set of examples or the collective set of problems has been labeled. We plan to translate that into a quality tag for each individual example or problem. .

Acknowledgments

This research project was supported by the Great Plains Software Technology Initiative and the Computer Science and Engineering Department at the University of Nebraska. The author thanks Todd Blank, L.D. Miller, and Akira Endo for their programming work and experiments, and the anonymous reviewers for their comments and feedback.

References

Avesani, P., Perini, A., and Ricci, F. 1998. The Twofold Integration of CBR in Decision Support Systems. In *Technical Report AAAI WS-98-02*. Menlo Park, CA: AAAI Press.

Beal, C. R., Qu, L., and Lee, H. 2006. Classifying Learner Engagement through Integration of Multiple Data Sources, in *Proc.* A*AAI'2006*, Boston, MA, 151-156.

Blank, T. 2005. ILMDA: An Intelligent Tutoring System with Integrated Learning, MS Thesis, Computer Science & Engineering, University of Nebraska, Lincoln, NE.

Bloom, B. S., Mesia, B. B., and Krathwohl, D. R. 1964. Taxonomy of Educational Objectives. New York, NY: David McKay.

Bonzano, A., Cunningham, P., and Smyth, B. 1997. Using Introspective Learning to Improve Retrieval in CBR: A Case Study in Air Traffic Control. In *Proc. ICCBR'97*, 291-302.

Cardie, C. 1999. Integrating Case-Based Learning and Cognitive Biases for Machine Learning of Natural Language, *J. Experimental & Theoretical AI*, 11(3):297-337.

Cassell, J., Annany, M., Basur, N., Bickmore, T., Chong, P., Mellis, D., Ryokai, K., Smith, J., Vilhjálmsson, H., and Yan, H. 2000. Shared Reality: Physical Collaboration with a Virtual Peer, *ACM SIGCHI Con. on Human Factors in Comp. Sys.*, April 1-6, The Hague, The Netherlands

Cox, M. T. and Ram, A. 1999. Introspective Multistrategy Learning: On the Construction of Learning Strategies, *AI*, 112(1-2):1-55.

Gertner, A. S. and VanLehn, K. 2000. ANDES: A Coached Problem-Solving Environment for Physics, in *Proc. ITS'2000*, 133-142.

Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W., and Harter, D. 2001. Intelligent Tutoring Systems with Conversational Dialogue, *AI Magazine*, 22(4):39-51.

Jarmulak, J., Craw, S., and Rowe, R. 2000. Self-Optimising CBR Retrieval. In *Proc. Int. Conf. Tools with AI*, 376-383.

Johns, J. and Woolf, B. 2006. A Dynamic Mixture Model to Detect Student Motivation and Proficiency, in *Proc.* A*AAI'2006*, Boston, MA, 163-168.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., and Mark, M. A. 1997. Intelligent Tutoring Goes to School in the Big City, *Int. J. AI in Ed.* 8(1):30-43.

Kolodner, J. 1993. *Case-Based Reasoning*. Morgan Kaufmann.

Leake, D. B., Kinley, A., and Wilson, D. 1995. Learning to Improve Case Adaptation by Introspective Reasoning and CBR. In *Proc. ICCBR'95*, 229-240.

Mott, B., Lee, S., and Lester, J. 2006. Probabilistic Goal Recognition in Interactive Narrative Environments, in *Proc.* A*AAI'2006*, Boston, MA, 187-192.

Park, C.-S. and Han, I. (2002). A Case-Based Reasoning with the Feature Weights Derived by Analytic Hierarchy Process for Bankruptcy Prediction, *Expert Systems with Applications*, 23(3):255-264.

Patterson, D., Rooney, N., and Galushka, M. 2002. A Regression Based Adaptation Strategy for Case-Based Reasoning. In *Proc. AAAI'2002*, 87-92.

Soh, L.-K., Blank, T. and Miller, L. D. 2005. Intelligent Agents that Learn to Deliver Online Materials to Students Better: Agent Design, Simulation, and Assumptions, in C. Chaoui, M. Jain, V. Bannore, and L. C. Jain (eds.) *Studies in Fuzziness and Soft Computing: Knowledge-Based Virtual Education*, Chapter 3, 49-80.

Soh, L.-K. and Blank, T. 2005. An Intelligent Agent that Learns How to Tutor Students: Design and Results. In *Proc. Int. Conf. on Computers in Education*, 420-427.

Soh, L.-K. and Luo, J. 2004. Cautious Cooperative Learning for Automated Reasoning in a Multiagent System, *Frontiers in Artificial Intelligence and Applications*, pp. 183-199, IOS Press.

Watson, I. And Marir, F. 1994. Case-Based Reasoning: A Review, *The Knowledge Engr. Review*, 9(4):327-354.

Wettschereck, D. and Aha, D. W. 1995. Weighting features. In *Proc. ICCBR'1995*, 347-358.

Zhang, Z. and Yang, Q. 2001. Feature Weight Maintenance in Case Bases Using Introspective Learning, *J. Intelligent Information Systems*, 16(2): 95-116.