Reinforcement Learning in a POMDP Based Intelligent Tutoring System for Optimizing Teaching Strategies

Fangju Wang

Abstract—**The abilities to improve teaching strategies online is important for an intelligent tutoring system (ITS) to perform adaptive teaching. Reinforcement learning (RL) may help an ITS obtain the abilities. Conventionally, RL works in a Markov decision process (MDP) framework. However, to handle uncertainties in teaching/studying processes, we need to apply the partially observable Markov decision process (POMDP) model in building an ITS. In a POMDP framework, it is difficult to use the improvement algorithms of the conventional RL because the required state information is unavailable. In our research, we have developed a reinforcement learning technique, which enables a POMDP-based ITS to learn from its teaching experience and improve teaching strategies online.**

Index Terms—**Computer supported education, intelligent tutoring system, reinforcement learning, partially observable Markov decision process.**

I. Introduction

Intelligent tutoring systems (ITSs) have become helpful tools in education [1]. In many fields, including mathematics, physics, medical science, astronaut training, and web-based adult education, teachers and students have benefited from ITSs [1]-[3]. It can be anticipated that ITSs would play more important roles in the future's education [1].

An ITS is characterized by adaptive teaching: It should be able to choose optimal teaching actions to satisfy individual students' studying needs. Reinforcement learning (RL) provides useful techniques for ITSs. RL enables an ITS to improves its teaching abilities online to fit its students. *Online* means that the system improves its teaching abilities continuously when it works, without stopping its teaching.

Currently, RL techniques in ITSs are mainly based on Markov decision processes (MDPs) [4]. An MDP has limitations in dealing with uncertainties in observing states. Such uncertainties commonly exist in teaching/studying processes. Quite often, the teacher does not know exactly what the student's states are, and what the most beneficial tutoring actions should be [1]. A partially observable Markov decision process (POMDP) can be more suitable for modeling a teaching/studying process. A POMDP allows an ITS to choose optimal tutoring actions even when the uncertainties exist.

Reinforcement learning on a POMDP is a challenging task because the state information required for policy improvement is unavailable. Currently few ITSs built on POMDP are able to perform online learning to improve adaptive teaching. With POMDPs, the systems obtained better abilities for handling uncertainties, but became weaker in improving teaching online. In our research, we address the challenge. We have developed a learning technique for a POMDP-based ITS to conduct online improvement of teaching strategies.

II. Related Work

Research for applying RL to intelligent tutoring started in 1990s. RL has been used in developing systems. In [5], Iglesias and co-workers used an RL algorithm to enable a system to adaptively teach individual students, based on its teaching experience. In [6], Litman and co-workers applied RL in a spoken tutoring system. The system engaged the student in a spoken dialogue to provide feedback and correct misconceptions. In the work reported in [7], Sarma and Ravindran proposed to use RL for building a tutoring system to teach autistic students. In the system, a policy is continuously updated for choosing appropriate teaching actions.

In addition to developing systems, researchers also worked on system evaluation and analysis. Chi and co-workers performed empirical evaluation on the application of RL to adaptive pedagogical strategies [8]. In [9], the researchers evaluated the learning performance of the educational system through three issues: The learning convergence, exploration/exploitation strategies, and reduction of training phase.

The work of applying POMDP to adaptive/intelligent tutoring also started in1990s. The more recent work included [10]-[14]. The work had the common feature of using POMDP to handle uncertainties in teaching processes.

In the work reported in [12], the researchers built a system for tutoring concepts. They developed a technique of teaching by POMDP planning. A core component in the technique was a set of approximate POMDP policies, for dealing with uncertainties. The work described in [14] was aimed at making POMDP solvers feasible for real-world problems. In the work, a POMDP was used to cope with uncertainties in learners' mental processes and plans, which were difficult to discern. A data structure was created to describe student status. The status was made up of knowledge states and cognitive states. The knowledge states were defined in terms of so called gaps. The POMDP enabled the intelligent tutor to take actions to discover and remove gaps, even when uncertainties existed.

In the existing work of applying RL to ITSs, learning was performed in the MDP framework. Such ITSs could improve their teaching through system-student interactions, but had difficulties to handle uncertainties. In the existing work for developing POMDP-based ITSs, progress has been achieved

Manuscript received May 31, 2017; revised March 13, 2018. This work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

F. Wang is with the School of Computer Science, University of Guelph, Ontario, Canada (e-mail: fjwang@uoguelph.ca).

in dealing with uncertainties. However, little has been done in online improvement of teaching abilities. In those systems, teaching strategies were fixed and would not be updated during teaching processes.

III. Technical Background

A. Reinforcement Learning

Reinforcement learning (RL) is an interactive machine learning technique [4]. It learns knowledge through interactions with the environment, and may update the knowledge it has learned while applying the knowledge to solve problems. RL is especially suitable for developing systems in which online learning is required or desirable.

Conventionally, RL is based on a Markov decision process (MDP). The components of an MDP include S, A, T, and ρ, where S is a set of states, A is a set of actions, T is a state transition probability function, and ρ is a reward function.

At any point in the decision process, the agent is in a state, where it applies policy $\pi(s)$ to choose the optimal action that maximizes the long term benefit. Policy π can be defined as

$$\pi(s) = \hat{a} = arg\max_{a} Q^{\pi}(s,a) \tag{1}$$

where $Q^{\pi}(s,a)$ is the *action-value function* evaluating the expected return after $a \in A$ is taken in $s \in S$ given π, defined as

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)V^{\pi}(s') \tag{2}$$

where $P(s'|s,a)$ is the transition probability from s to $s' \in S$ after a is taken, $V^{\pi}(s)$ is the *state-value function* evaluating the expected return of s given policy π, defined as:

$$V^{\pi}(s) = \sum\nolimits_{a} \pi(s,a) \sum\nolimits_{s'} P(s'|s,a)[\mathcal{R}(s,a,s') + \gamma V^{\pi}(s')] \tag{3}$$

where $\pi(s,a)$ returns the probability that a may maximize the long term benefit when it is taken in s, γ is a discounting factor, and $\mathcal{R}(s,a,s')$ is the expected reward after the agent takes a in s and enters s' .

When π is optimal, $\pi(s)$ returns the optimal action. To ensure optimality of the policy, we need to periodically improve the policy based on the information obtained from interactions.

Let π be the current policy, and $\pi(s)$ be the action maximizing $V^{\pi}(s)$. In state s if there is an action $a \neq \pi(s)$, we may expect that there exists a policy π' such that $a = \pi'(s)$. If $\forall s \in S$,

$$Q^{\pi}(s,\pi'(s)) \geq V^{\pi}(s) \tag{4}$$

then based on the policy improvement theorem [4], π' must be as good as, or better than, π. That is, $\forall s \in S$

$$V^{\pi'}(s) \geq V^{\pi}(s) \tag{5}$$

The task of policy improvement is to find such a π' if it exists. Learning from interactions to improve the policy continuously is a central part of RL.

B. Partially Observable Markov Decision Process

In an MDP, choosing an optimal action requires that the agent knows exactly which state it is in (see Eqn (1)). However, in many applications, states are not completely observable. To enable the agent to make decisions with uncertainties in observing states, we can model the decision process as a partially observable Markov decision process (POMDP).

A POMDP is an extension of an MDP. The components in a POMDP include S, A, T, ρ, O, and Z. Of them, S, A, T, and ρ are the same as in the underlying MDP, O is a set of observations, and Z is the observation probability function.

When states are not completely observable, the agent infers the information of the current state from its immediate action and observation, and represents the information as a *belief*:

$$b = [b(s_1), b(s_2), \ldots, b(s_Y)] \tag{6}$$

where $s_i \in S$ $(1 \leq i \leq Y)$is the ith state in S, Y is the number of states in S, $b(s_i)$ is the probability that the agent is in s_i, and $\sum_{i=1}^{Y} b(s_i) = 1$.

A policy in POMDP is a function of belief: $\pi(b)$. In a decision step in a POMDP, the agent is in s_t (which is not completely observable), chooses and takes a_t based on its belief b_t , observes o_t , enters state s_{t+1} (which is not completely observable either), and infers $b_{t+1}$from b_t, a_t, and o_t.

In a POMDP, policies can be represented as *policy trees*. In a policy tree, a node is labeled with an action, and an edge is labeled with an observation. At a node, there is an edge for each of the possible observations, connecting to a node at the next level. Based on a policy tree, after an action is taken, the next action is determined by the agent's observation.

With the policy tree technique, a policy can be defined as:

$$\pi(b) = \hat{\tau} = arg\max_{\tau \in \mathcal{T}} V^{\tau}(b) \tag{7}$$

where τ is a policy tree, $\hat{\tau}$ is the optimal policy tree at b, $\mathcal{T}$ is the set of policy trees to evaluate in making the decision, and $V^{\tau}(b)$ is the value function at b given policy tree τ:

$$V^{\tau}(b) = \sum_{s \in S} b(s)V^{\tau}(s) \tag{8}$$

where $V^{\tau}(s)$ is the value function of $s \in S$ given tree τ:

$$V^{\tau}(s) = \mathcal{R}(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \sum_{o \in O} P(o|a,s')V^{\tau^{o}}(s') \tag{9}$$

where a is the root action of τ, o is the observation after the agent takes a, τ^{o} is the subtree in τ which is connected to the node of a by the edge of o , $\mathcal{R}(s,a)$ is the expected immediate reward after a is taken in s, and $P(o|a,s')$ is the observation probability for the agent to observe o after it takes a and enters s'.

IV. An ITS Built on POMDP

A. An Overview

We built an ITS for experimenting our techniques for developing POMDP-based ITSs, including the technique for online learning to improve tutoring strategies reported in this paper. The system is for teaching concepts. It tutors a student at a time, in a way of question - and - answer. Fig. 1 illustrates some sample questions and answers in using the ITS.

m1: What would you like to learn?
h1: What is a database?
m2: To understand "database", you may need to learn *file* first. May I teach *file*?
h2: Yes, please.
m3: A file is a set of data on some durable storage.
h3: OK.
m4: You may also need to know *query language.*
May I teach *query language*?
h4: Yes.
m5: A query language is a high-level language for querying.
h5: But what is a high-level language?
m6: A high-level language is a programming language independent of any particular type of computer, and is closer to human languages than assembly languages.
May I return to *database*?
h6: Yes.
m7: A database is a collection of interrelated files manipulated by application programs written in a query language.
h7: I see.
m8: Other questions?
...

Fig. 1. System and student actions in tutoring. "m" – machine (system), "h" – human (student).

The ITS consists of a student model, a tutoring model, and a domain model. The student model tracks information of the student's knowledge states. The tutoring model represents tutoring strategies. The domain model stores knowledge of the instructional subject. In each tutoring step, the agent accesses the student model, and takes a tutoring action based on information of the student's current knowledge state.

In many science and mathematics subjects, concepts have prerequisite relationships. To well understand a concept, a student should first study all the prerequisites of the concept. For example, in calculus, *function* and *limit* are prerequisites of *derivative*. A student should study *function* and *limit* before *derivative*. When answering a question about a concept, a teacher must determine whether to teach a prerequisite for the student to make up some knowledge, and if so, which one.

A goal of our ITS is to choose an optimal action in each tutoring step. An *optimal action* is to teach a concept that the student needs to study, and can study without making up other concepts. The concept to teach may be a prerequisite the student needs to study to understand the concept originally asked. For example, in Fig. 1, m3 is an optimal action if the student did not know *file* before.

B. States, Actions, and Observations

We cast the ITS structure onto a POMDP: The POMDP states represents the student model; the policy represents the tutoring model. The agent takes actions to teach, and treats student actions as observations.

We define POMDP states in terms of the concepts in the instructional subject. When there are D concepts in the subject, we create D Boolean variables, with variable $\mathbb{C}_i$ representing concept C_i $(1 \le i \le D)$. Variable $\mathbb{C}_i$ may take two values: $\surd C_i$ and $\neg C_i$. $\surd C_i$ represents that the student already understands C_i, while $\neg C_i$ represents that the student does not understand C_i. We associate each $s \in S$ with a conjunctive formula of the D variables to represent a knowledge state of the student. For example, formula $(\surd C_1 \land \surd C_2 \land \neg C_3 \land \ldots)$ represents a state in which the student understands C_1 and C_2, does not understand C_3, ... We call it a state formula. The states defined in this way have the Markov property. To deal with the exponential state space, we have developed techniques to make the computing efficient [15].

As mentioned, the system teaches in a question-and-answer manner. The student's actions are mainly asking questions about concepts, and the system's actions are mainly answering questions. In a tutoring step, the agent takes action a, and then observes o, which is the next question from the student.

V. Online Policy Improvement

A. Policy Initialization

In this section, we will first describe how we initialize a policy in the POMDP, and then how we improve the policy online while the system teaches.

As discussed, policy $\pi(b)$ is defined in term of belief value function $V^{\pi}(b)$ (see Eqn (7)), $V^{\pi}(b)$ is defined in terms of state value function $V^{\pi}(s)$ (see Eqn (8)), and $V^{\pi}(s)$ is defined as a function of $\mathcal{R}(s,a)$, $P(s'|s,a)$, and $P(o|a,s')$ (see Eqn (9)). In policy initialization, we initialize the parameters $\mathcal{R}(s,a)$, $P(s'|s,a)$, and $P(o|a,s')$. As will be discussed later, in policy improvement, we update them.

We create a set of training data for initializing the parameters. The training data set is a sequence of tuples

$$
\begin{array}{l}
(s_1, a_1, o_1, r_1, s_2) \\
\cdots \\
(s_i, a_i, o_i, r_i, s_{i+1}) \\
(s_{i+1}, a_{i+1}, o_{i+1}, r_{i+1}, s_{i+2}) \\
\cdots
\end{array}
$$

where s denotes a state, a an action, o an observation, and r a reward. The sequence simulates teacher-student interactions, with a tuple being a tutoring step: The agent is in a state, takes an action, perceives a student action (observation), receives a reward, and then enters a new state.

After the agent takes an action (teaching a concept), if the student rejects the action, the agent receives a *low reward*; otherwise it receives a *high reward*. In a tutoring step, after the agent teaches C, if the student says something like "I already know C", we consider that the student rejects the agent's action. If the student asks about a prerequisite of C, we also consider the student rejects the agent's action (answer) because the student is not satisfied with the answer. For example, in Fig. 1, we consider the student rejects the system action for teaching *query language* when asking "But what is a high-level language?" because the student still does not understand what a query language is.

Formally, the *reward function* $\rho(o,a)$ is defined as

$$\rho(o,a) = \{ \begin{matrix} r' & if\ o\ rejects\ a \\ r" & otherwise \end{matrix} \quad (10)$$

where $r"$ is the high reward, r' is the low reward, and $r" > r' > 0$. The rewards in the training data are assigned by using the reward function.

A tuple in the training data contains instances of the relationships $S \times A \longrightarrow S$ and $A \times S \longrightarrow O$. Let "*" be any value and "| |" be the operator of counting tuples. $P(s'|s,a)$ and $P(o|a,s')$ can be initialized as

$$P(s'|s,a) = \frac{|(s,a,*,*,s')|}{|(s,a,*,*,*)|}, \quad (11)$$

$$P(o|a,s') = \frac{|(*,a,o,*,s')|}{|(*,a,*,*,s')|}. \quad (12)$$

To initialize $\mathcal{R}(s,a)$, we calculate $\mathcal{R}(s,a,s')$, which is the expected immediate reward after the agent takes a in s and enters s' :

$$\mathcal{R}(s,a,s') = \frac{\sum(r \in (s,a,*,r,s'))}{|(s,a,*,r,s')|} \quad (13)$$

Based on $\mathcal{R}(s,a,s')$ and $P(s'|s,a)$, we can initialize $\mathcal{R}(s,a)$:

$$\mathcal{R}(s,a) = \sum_{s' \in S} P(s'|s,a)\mathcal{R}(s,a,s'). \quad (14)$$

In (11), (12), and (13), the counting and summation are conducted on the training data. We use the Lidstone estimate [16] to deal with the data sparsity problem. After calculating $P(s'|s,a)$, $P(o|a,s')$, and $\mathcal{R}(s,a)$from the training data, we have initialized the policy. The agent can use it to choose actions, and will improve it when teaching students.

B. Parameter Update

In policy improvement, the agent learns from the system-student interactions, and continuously improves the policy for better teaching performance.

We use a *delayed updating* method for the improvement. In this method, the agent applies the current policy for a number of sessions without changing it. During the sessions, the agent records system and student actions, as well as its beliefs, and keep them in a logfile. After a given number of sessions, it updates the policy using the recorded data. Then it applies the updated policy, and then improves the policy again, and so on.

The agent improves a policy in two steps: It uses the recorded data to update the policy parameters, and then it improves the policy based on the updated parameters.

As mentioned, in a POMDP, states are not completely observable, and the agent infers information of states and represents the information as beliefs. For this reason, the agent cannot record states and state transitions, and updates the probabilities and rewards directly from state information. In our technique, the agent updates the parameters using information in beliefs.

The data recorded during tutoring sessions are tuples of:

$$\begin{array}{l} (b_1, a_1, o_1, r_1, b_2) \\ \dots \\ (b_i, a_i, o_i, r_i, b_{i+1}) \\ (b_{i+1}, a_{i+1}, o_{i+1}, r_{i+1}, b_{i+2}) \\ \dots \end{array}$$

In a tuple, b denotes a belief. When recording the data, the agent determines rewards by using the reward function in (10).

A state transition probability $P(s'|s,a)$ is updated as:

$$P(s'|s,a) = N_1/(N_2 + N_4) + N_3/(N_2 + N_4) \quad (15)$$

where

$$N_1 = \sum_{b,b'}[|(b,a,*,*,b')| \times b(s) \times b'(s')] \quad (16)$$

$$N_2 = \sum_{b,b'}[|(b,a,*,*,b')| \times b(s)], \quad (17)$$

N_3 is the sum of the numerator in (11) and the cumulated N_1 values in previous updates, and N_4 is the sum of the denominator in (11) and the cumulated N_2 values in previous updates.

An observation probability $P(o|a,s')$ is updated as:

$$P(o|a,s') = M_1/(M_2 + M_4) + M_3/(M_2 + M_4) \quad (18)$$

where

$$M_1 = \sum_{b,b'}[|(b,a,o,*,b')| \times b'(s')] \quad (19)$$

$$M_2 = \sum_{b,b'}[|(b,a,*,*,b')| \times b'(s')] \quad (20)$$

M_3 is the sum of the numerator in (12) and the cumulated M_1 values in previous updates, and M_4 is the sum of the denominator in (12) and the cumulated M_2 M2 values in previous updates.

To update $\mathcal{R}(s,a)$, we first update $\mathcal{R}(s,a,s')$:

$$\mathcal{R}(s,a,s') = R_1/(L_1 + L_2) + R_2/(L_1 + L_2) \quad (21)$$

where

$$R_1 = \sum_{b'b}[(r \in (b,a,*,r,b')) \times b(s) \times b'(s')] \quad (22)$$

$$L_1 = |(b,a,*,r,b')| \times b(s) \times b'(s'), \quad (23)$$

R_2 is the sum of the numerator in (13) and the cumulated R_1 values in previous updates, and L_2 is the sum of the denominator in (13) and the cumulated L_1 values in previous updates. Then we can update $\mathcal{R}(s,a)$ by using (14).

C. Policy Improvement

After updating the parameters, for each policy tree τ, the agent executes the following procedure, which is modified from the evaluation- improvement algorithm proposed in [4]. In the procedure, b_s denotes a belief in which $b(s) = 1$ and all the other elements are zeros, τ^o is the subtree connected by o, and θ is a small positive number.

```
1. Initialization
      Initialize $V^\tau$
2. Policy Evaluation
      Repeat
         $\Delta \longleftarrow 0$
         For each $s \in S$
            $v \longleftarrow V^\tau(s)$, $a \longleftarrow$ root action of $\tau$
            $V^\tau(s) \longleftarrow \mathcal{R}(s,a) +$
                  $\gamma \sum_{s'} P(s'|s,a) \sum_o P(o|a,s') V^{\tau^o}(s')$
            $\Delta \longleftarrow \max(\Delta, |v - V^\tau(s)|)$
      Until $\Delta < \theta$
3. Policy Improvement
      policy_stable $\longleftarrow$ true
      For each $s \in S$
         $\tau' \longleftarrow \pi(b_s)$, $\pi(b_s) \longleftarrow arg \max_\tau V^\tau(b_s)$
         If $\tau' \neq \pi(b_s)$ then policy_stable $\longleftarrow$ false
      If policy stable, then stop; else go to 2
```

VI. Experiments

We implemented our learning technique in the experimental ITS. We tested it through examining the teaching performance of the policy obtained by the machine learning technique.

The system could work in two modes: with the POMDP and RL on or off. When the POMDP and RL were on, the system taught adaptively. Each time after the student asked about a concept that has prerequisites, the system applied the policy to choose an optimal action to take. Meanwhile it continuously improved the policy. When the POMDP and RL were off, the system either directly answered the question, or randomly taught a prerequisite.

In testing the performance of adaptive teaching, we used a domain knowledge data set of software basics. 30 students participated in the experiment. The students knew how to use operating systems of Microsoft Windows and Apple OS X, and application programs like Web browsers, Microsoft Office, etc. None of them had formal training in software development, nor took a course on software. We randomly divided the participants into two groups of the same size. Group 1 studied with the ITS when the POMDP and RL were turned on, while Group 2 studied with the system when the POMDP and RL were off. The student and system interactions were recorded for performance analysis.

We used *rejection rate* to evaluate the system's performance in adaptive teaching: A rejected system action is not optimal. For a student, the rejection rate was the ratio of the number of system actions rejected by the student to the total number of system actions for teaching him/her. A student's rejection rate could be used to evaluate how well the system chose optimal actions in teaching this student.

TABLE I: Number of Participants, Mean and Estimated Variance of Each Group

	Group 1	Group 2
Number of participants	$n_1 = 15$	$n_2 = 15$
Mean	$\overline{X_1} = 0.5966$	$\overline{X_2} = 0.2284$
Estimated variance	$s_1^2 = 0.0158$	$s_2^2 = 0.0113$

We calculated the average rejection rates and variances for the two groups (see Table I): The adaptive teaching reduced the rejection rate from about 60% to 23%. The method for analyzing the experimental results was the independent samples t-test. In the analysis, the alternative hypothesis H_a was that the POMDP based RL improved the teaching performance of the ITS, and the null hypothesis H_0 was that the learning did not improve the performance. The analysis suggested that we could reject H_0 and accept H_a, which indicated that the difference between the two means was significant: The online policy improvement works. For more details about the analysis, see [17].

VII. Concluding Remarks

Reinforcement learning (RL) and the partially observable Markov decision process (POMDP) model both offer powerful techniques essential for building ITSs. However, their marriage has not generated good results. In our research, we have developed a learning technique, which was aimed at enabling a POMDP-based ITS to improve its teaching abilities online. Experimental results suggested that the technique might serve as a good starting point for building practical ITSs.

References

[1] B. P. Woolf, *Building Intelligent Interactive Tutors*, Burlington, MA, USA: Morgan Kaufmann Publishers, 2009.

[2] B. Cheung, L. Hui, J. Zhang, and S. M. Yiu, "SmartTutor: An intelligent tutoring system in web-based adult education," *Elsevier the Journal of Systems and Software*, vol. 68, pp. 11-25, 2003.

[3] K. Lehn, B. Sande, R. Shelby, and S. Gershman, "The Andes physics tutoring system: An experiment in Freedom," *Advances in Intelligent Tutoring Systems*, Berlin Heidelberg: Springer-Verlag, pp. 421-443, 2010.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Cambridge, Massachusetts: The MIT Press, 2005.

[5] A. Iglesias, P. Martinez, and F. Fernandez, "An experience applying reinforcement learning in a web-based adaptive and intelligent educational system," *Informatics in Education*, vol. 2, no. 2, pp. 223-240, 2003.

[6] A. J. Litman and S. Silliman, "Itspoke: An intelligent tutoringspoken dialogue system," in *Proc. Human Language Technology Conference 2004*, 2004.

[7] B. Sarma and B. Ravindran, "Intelligent tutoring systems using reinforcement learning to teach autistic students," *Home Informatics and Telematics: ICT for The Next Billion*, Springer, vol. 241, pp. 65-78, 2007.

[8] M. Chi, K. Lehn, D. Litman, and P. Jordan, "Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies," *User Model User-Adap*, Kluwer Academic, pp. 137-180, 2011.

[9] A. Iglesias, P. Martnez, R. Aler, and F. Fernndez, "Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning," *Applied Intelligence*, vol. 31, no. 1, pp. 89-106, 2009.

[10] J. D. Williams and S. Young, "Partially observable Markov decision processes for spoken dialog systems," *Elsevier Computer Speech and Language*, vol. 21, pp. 393-422, 2007.

[11] G. Theocharous, R. Beckwith, N. Butko, and M. Philipose, "Tractable POMDP planning algorithms for optimal teaching inSPAIS," in *Proc. IJCAI PAIR Workshop 2009*, 2009.

[12] A. N. Rafferty *et al*., "Faster teaching by POMDP planning," in *Proc. Artificial Intelligence in Education (AIED) 2011*, pp. 280-287, 2011.

[13] H. R. Chinaei, B. Chaib-draa, and L. Lamontagne, "Learning observation models for dialogue POMDPs," *Canadian AI'12 Proceedings of the 25th Canadian Conference on Advances in Artificial Intelligence*, Springer-Verlag Berlin, Heidelberg, pp. 280-286, 2012.

[14] J. T. Folsom-Kovarik, G. Sukthankar, and S. Schatz, "Tractable POMDP representations for intelligent tutoring systems," *ACM Transactions on Intelligent Systems and Technology (TIST) -Special Section on Agent Communication, Trust in Multiagent Systems, Intelligent Tutoring and Coaching Systems Archive*, vol. 4, no. 2, p. 29, 2013.

[15] F. Wang, “Handling exponential state space in a POMDP-based intelligent tutoring system,” in *Proc. 6th International Conference on E-Service and Knowledge Management (IIAI ESKM 2015)*, pp. 67-72, 2015.
[16] G. J. Lidstone, “Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities,” *Transactions of the Faculty of Actuaries*, vol. 8, pp. 182-192, 1920.
[17] P. Zhang, “Using POMDP-based reinforcement learning for online optimization of teaching strategies in an intelligent tutoring system,” MSc thesis, University of Guelph, Canada, 2013.

Fangju Wang obtained his BEng from the Central-South University, Changsha, China, the MSc from Peking University, Beijing, China, the PhD from the University of Waterloo, Waterloo, Canada. He is presently a professor in the School of Computer Science, University of Guelph, Canada. Fangju Wang’s research fields include artificial intelligence, machine learning, intelligent systems, and computer supported education. He has had more than one hundred papers published in major journals and conference proceedings. Currently he is working in the development of intelligent tutoring systems (ITSs), focusing on issues in uncertainty handling, system performance, and online improvement.