Improving Student's Modeling Framework in a Tutorial-Like System based on Pursuit Learning Automata and Reinforcement Learning

Seyedeh Leila Javadi[1], Behrooz Masoumi[2], Mohammad Reza Meybodi[3]
[1]Sama Technical And Vocational Training College
Islamic Azad University Ayatolah amoly Branch
Amol, Iran
[2]Department of Computer and Information Technology Engineering
Islamic Azad University, Qazvin Branch
Qazvin, Iran
[3]Department of Computer and Information Technology Engineering
Amirkabir University of Technology
Tehran, Iran
Leila_javadi87@yahoo.com, Bmasoumi@Qazviniau.ir, mmeybodi@aut.ac.ir

ABSTRACT: *Intelligent Tutorial Systems are educational software packages that occupy Artificial Intelligence (AI) techniques and methods to represent the knowledge, as well as to conduct the learning interaction. Tutorial-like systems simulates a Socratic model of learning for teaching uncertain course material by simulating the learning process for both Teacher and a School of Students. The Student is the center of attention in any Tutorial system. The proposed method in this paper improves the student's behavior model in a tutorial-Like system. In the proposed method, student model is determined by high level learning automata called Level Determinant Agent (LDA-LAQ), which attempts to characterize and improve the learning model of the students. LDA-LAQ actually use learning automata as a learning mechanism to show how the student is slow,normal or fast in the term of learning. This paper shows how learning model increases speed accuracy using Pursuit learning automata and Reinforcement Learning.*

Keywords: Learning Automata (LAs), Q Learning, Student Modeling, Tutorial-Like Systems

Received: 12 August 2012, Revised 3 October 2012, Accepted 8 October 2012

1. Introduction

Intelligent Tutoring System (ITS) is one of the best ways of one-to-one teaching. ITS instructs the subject to a student, who is using it. The student has to learn the subject from an ITS by solving problems. ITSs are special educational software packages that involve artificial intelligence techniques and methods to represent the knowledge, as well as conducting the learning interaction [1-3]. ITSs are characterized by their responsiveness to the learner's need. They get adapted according to the knowledge/skill of the users. They also incorporate experts' domain-specific knowledge.

According to various discussions on intelligibility concept and various practical fields, there are various suggestions for architectures and components on intelligent tutoring system. In general, these systems basically include three main components and sometimes a communication component is added [4, 5]. Three components which form intelligent tutoring system are Domain model, Student model and Pedagogical [3].

Student modeling is the focal point and one of the components of training system. This model is a representation of student behavior and status, and also a paragon which models student status. The model can recognize Student Model fast and more accurately. In the proposed model it is supposed that student can be modeled in three status of slow, normal and fast student [3, 7]. The system determines Student Learning Model with scrutiny of student continuous actions and using LDA-LAQ.

In some researches, using machine learning on didactic systems improvement was conducted. Frasson in 1996 have designed main components of intelligent tutoring system (student model, domain model and Pedagogical model) as intelligent elements [8]. Lelouche in 2000 used a set of interactive elements for student main modeling on intelligent tutoring system [9]. Legaspi and Sison in 2000 have modeled instruction on intelligent tutoring system such as central learning procedure using reinforcement learning [3]. Mooney and Baffes in 1996 utilize ASSERT which use reinforcement learning in Student model in order to calculate new student's error only by means of using correct domain knowledge[7]. Hashem in 2007 utilized learner automata in intelligent tutoring system [3].

The aim of the paper is to improve student modeling on Learning System and it is shown that how the modeling can be successful in tutorial-like system frame work by the help of Learning Automata and reinforcement Learning. In order to improve how to determine Student Modeling, three students type were modeled according to Q Learning Algorithms.

This paper structs as follows: section 2 gives a brief description of Learning Automata and Q Learning. In section 3 Tutorial–like system is explained. In section 4 the model of student is represented. In Section 5 experimental results will be discussed and finally the conclusion is in section 6.

2. Learning Automata and Q-learning

In this section, concepts of Learning Automata and Q Learning as two Learning models have been described.

2.1 Learning Automata

Learning Automata are adaptive decision-making devices operating on unknown random environments [12]. The Learning Automaton has a finite set of actions and each action has a certain probability (unknown for the automaton) of getting rewarded by the environment around the automaton. The aim is learning to choose the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction on the system. If the learning algorithm be chosen properly, then the iterative process of interacting with the environment leads to result in selecting the optimal action.

Figure1 illustrates how a stochastic automaton works in feedback connection with a random environment. Learning Automata can be classified into two main families: fixed structured learning automata and variable structured learning automata (VSLA) [12]. In the following, the variable structured learning automaton is described. In a stochastic variable structured automaton, the probabilities of the various actions are updated on the basis of the information the environment provides. Action probabilities are updated at every stage using a reinforcement scheme. It is defined by a quadruple $\{\alpha, \beta, P, T\}$ in which α is the action or output set of the automaton, β is a random variable in the interval [0,1], P is the action probability vector of the automaton or agent, and T denotes an update scheme.

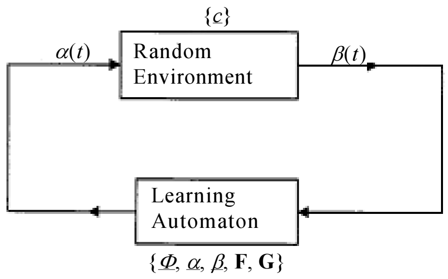


Figure 1. Learning automata environment [12]

The output of the automaton is actually the input to the environment. The input β of the automaton is the output of the environment, which is modeled through penalty probabilities *ci*. Important examples of linear update schemes are linear reward–penalty, linear reward–inaction, and linear reward–ε penalty. The philosophy of those schemes is essentially to increase the probability of an action which results in a success and to decrease it when the response is failure [12].

2.2 Q-Learning

Q-learning is a popular RL algorithm that does not need a model of its environment and can used on-line. In Q-learning algorithm, the values of state-action pairs are estimated. After these Q-values are obtained, action having the highest Q-value of any state would be the optimal action for that particular state, this action is called *greedy* action. If the greedy action is selected then we are *exploiting* the action values. Instead, if we select an action other than greedy action then we are *exploring* the action values [13].

Generally, exploration is done to increase the total reward in long-run. Q-values are estimated on the basis of experience as in Equation (1).

$$Q(s, a) \leftarrow Q(s, a) + \alpha\,[r + \gamma \max Q(s', a') - Q(s, a)] \qquad (1)$$

This algorithm converges to the correct Q-values with probability one if the environment is stationary and if the state is Markov. In order to increase the total reward, actions are selected from these Q-values e-greedily, which means we are exploring the action values with e probability, and for the remaining time we are exploiting [13].

3. Tutorial-*Like* Systems

Since our research involves tutorial-*like* systems, which are intended to mimic tutorial systems, a brief overview is represented as follows.

In these systems, it's not necessary to be *real-life* students, but rather, each student could be replaced by a student simulator that mimics a *real-life* student. Alternatively, it could also be a software entity that attempts to learn. The teacher, in these systems, attempts to present the teaching material to a *school* of student simulators. The students are also permitted to share information between each other to gain knowledge [6].

Tutorial-*like* systems share some similarities with the well developed field of tutorial systems. Thus, they model the teacher, the student, and the domain knowledge. However, they are different from "*traditional* " tutorial systems in the characteristics of their models.

The tutorial-*like* system derives a model for the student by assessing and determining the way he learns. To achieve this, Hashem [7] assumed that the system has a finite set of possible learning models for each student and the domain knowledge will be presented via multiple-choice Socratic-type questions. Thus, for each question, every choice has a corresponding probability of being correct, implying that the choice with the highest reward probability is the answer to the question [21].

The finite set of learning models represents the different families that characterize the way the student learns. As mentioned, if the tutorial-*like* system can understand how the student perceives knowledge, it will be able to customize the way in which it communicates the knowledge to the student toward an optimal teaching strategy.

Tutorial-*like* system in [3] incorporates multiple LAs that are indirectly interconnected with each other. The student modeler will itself utilize an LA in the *meta-LA* level to represent the different potential learning models for the student. Furthermore, as the student interaction with the tutorial-*like* system increases, *this* LA would hopefully converge to the model that most accurately represents the way the student learns [3].

4. Proposed model

In this section, a new model based on learning automata and Q-learning is proposed. In the proposed model, Meta LA mentioned in [3] is restructured based on LA and Q-learning. The structure of the proposed model which we call it LDA-LAQ is given with more details in Figure 2.

The LAD-LA\Q in Tutorial-like system LA and Q learning which are, indirectly, interconnected with each other. We observe that our system represents a synchronous model.

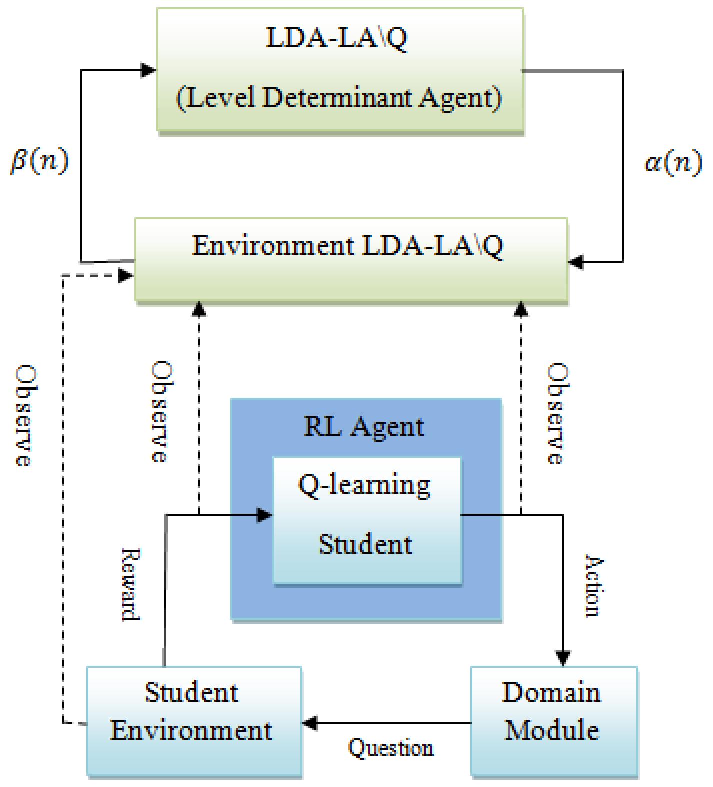


Figure 2. Proposed Model Using a Network of LA and Q learning

However, our system has a rather unique LA and Q learning interaction model. While the Student Q is affecting the LDA-LAQ, there is no direct connection between them. The LDA-LAQ Environment monitors the performance of the Student Q over a period of time, and depending on that, the Environment would penalize or reward the action of the LDA-LAQ.

This model represents a new structure of interconnection which can be viewed as a composition of two levels: a higher-level automaton, i.e., the LDA-LAQ, and a lower-level agent Q, which is the Student Q. The convergence of the higher-level automaton is dependent on the behavior of lower-level agent Q. observing the uniqueness of such interactive modules, organizes as the following consequence:

1. The LDA-LAQ has access to the environment of the lower level, including the set of the penalty probabilities that are unknown to the lower-level Q agent.

2. The LDA-LAQ has access to the actions chosen by the lower-level Q agent.

3. Finally, the LDA-LAQ has access to the responses made by the lower-level environment.

In conclusion, items 1-3 above collectively constitute the environment for the LDA-LAQ. Such a modeling is unknown in the field of Q agent.

In this paper, to speed up student learning, Q Learning Algorithm is considered without learners' simulation and DP_{RI} is used to increase student model determination accuracy in LDA-LAQ. So by means of this method we are able to increase two mentioned factors simultaneously on Student Model determination.

Student Model will use Q-Learning without status with three different learning rates for slow, normal and fast instead of LA in [3]. Each of them is used as an action for high level Automata.

Stateless Q-learning yields a temperature parameter that allows balancing exploration and exploitation. It proceeds in two steps: Action selection based on the temperature and applying a Q-value update to the played action. The Q-value update rule Equation (2) features a learning rate and is applied to the selected action i played in joint action $\mathbf{s_t}$, after receiving the payoff $r_t = u_i(s_t)$.

$$Q_{(t+1)}(a) \leftarrow Q_t(a) + \lambda(r - Q_t(a)) \tag{2}$$

Given the Q-values, an action is selected based on the Boltzmann distribution, with probability P_i for action i:

$$P_i(\text{action}) = \frac{e^{\frac{EV(\text{action})}{\tau}}}{\sum_{(\text{action} \in A_i)?} e^{\frac{EV(\text{action})}{\tau}}} \tag{3}$$

In normal status EV(action) = Q(action) is chosen [26].

In the proposed model, the *LDA-LAQ* is used to learn the best possible model for the student simulator as a 4-tuple: $\{\alpha, \beta, P, T\}$, where the variables are described as follows.

Which the actions set is $\{\alpha_1, \alpha_2, \alpha_3\}$. α_1 shows action is respondent with slow learner, α_2 shows action is respondent with normal learner and α_3 shows action is respondent with fast learner.

$\beta=\{0,1\}$, $\beta=0$ implies the selected reward for this action and $\beta=1$ implies the selected fine for the action. $P = [p_1, p_2, p_3]^T$, $P_{i(n)}$ is student simulation current probability which is shown with α_i and T is probability updating $T: (P, \alpha, \beta) \rightarrow P$. Each of these will now be clarified.

The student simulator was implemented to mimic three typical types of students as follows:

• Slow learner (α_1): For this type of students, the student simulator uses a stateless Q learning with $\lambda = 0.3$ to mimic the student's behavior.

• Normal learner (α_2): For this category of students, the student simulator uses a stateless Q learning with $\lambda = 0.5$ to simulate the student's behavior.

• Fast learner (α_3): To simulate students of this type, the student simulator utilizes a stateless Q learning with $\lambda = 0.7$ to represent the student's behavior.

β is the input that the *LDA-LAQ* receives. *LDA-LAQ* must observe a sequence of decisions made by the student simulator, and based on this sequence, it must deduce whether its current model is accurate or not. We propose to achieve it as follows:

For a fixed number of queries, which we shall refer to as the "*window*", the *LDA-LAQ* assumes that the student simulator's model is αi. Let $\theta(t)$ the rate of learning associated with the lower RL agent at time "*t*," measured in terms of the number of penalties received for any action, weighted by the current action probability vector.

By inspecting the way by which the student simulator learns within this window, it infers whether this current model should be rewarded or penalized. We propose to achieve this by using two thresholds, $\{\theta j \mid 1 \leq j \leq 2\}$ (as shown in Table 1).

If the current observed $\theta(t)$ of the student simulator is less than $\theta 1$, we assert that the student simulator demonstrates the phenomenon of a fast learner. Now, if the *LDA-LAQ* has chosen $\alpha 3$, this choice is rewarded. Otherwise, this choice is penalized.

If the current observed $\theta(t)$ of the student simulator be equal or greater than $\theta 1$ and less than $\theta 2$, we assert that the student simulator demonstrates the phenomenon of a *normal* learner. Thus, if the *LDA-LAQ* has chosen $\alpha 2$, this choice is rewarded. Otherwise, this choice is penalized.

Finally, if the current observed $\theta(t)$ of the student simulator is equal or greater than $\theta 2$, we assert that the student simulator demonstrates the learning properties of a *slow* learner. Thus, if the *LDA-LAQ* has chosen $\alpha 1$, this choice is rewarded. Otherwise, the *LDA-LAQ* is penalized.

Type Of Student	Boundary Threshold
Slow Learner	$\theta(t) < \theta_1$
Normal Learner	$\theta_1 \leq \theta(t) < \theta_2$
Fast Learner	$\theta_2 \leq \theta(t)$

Table 1. Threshold used LDA-LAQ [3]

P is the action probability vector that contains the probabilities that the *LDA-LAQ* assigns to each of *its* actions. At each instant *t*, the *LDA-LAQ* randomly selects an action $\alpha(t) = \alpha i$. The probability that the automaton selects action αi at time *t* is the action probability.

$$p_i(t) = pr[\alpha(t) = \alpha_i],$$
$$where, \sum_{i=1}^{r} p_i(t) = 1 \;\forall t, i = 1, 2, 3 \tag{4}$$

Initially, the *LDA-LAQ* will have an equal probability for each of the three learning models as Equation (5):

$$p(0) = [0.333, 0.333, 0.333]^T \tag{5}$$

T is the updating algorithm or the reinforcement scheme used to update the *LDA-LAQ*'s action probability vector and is represented by Equation (6):

$$p(t+1) = T[p(t), \alpha(t), \beta(t)] \tag{6}$$

Learning Automata model is an continuous schema in which actions' probability can be a value between [0,1]. Thathachar and Oommen posed discretized learning Automata concept by choosing probability restrict form a finite set of values between [0,1]. The most important advantage of using discrete algorithm is to speed up [23]. Oommen have proved that discrete Automata in all environments have faster convergence in comparison with their continuous counterparts [24]. Pursuit algorithms as DPRI are used in the proposed models which are sort of estimator algorithm and follow optimized action.

In the proposed model, DP_{RI} schema is used instead of DL_{RI} in [3]. the action probability vector is updated in a discrete manner when the *LDA-LAQ* is rewarded and unchanged when the *LDA-LAQ* is penalized. When the chosen action is rewarded, the algorithm decreases the probability for al1 the actions that do not correspond to the highest estimate, by a step A, where A= l / rN. In order to keep the sum of the components of the vector P(t) equal to unity, the LA can be described entirely by the following action probability updating equations [25].

As presented in Section IV, the *LDA-LAQ* was implemented as a higher level LA, which had access to the learning environment and the actions and penalties of the lower level agent Q associated with the student simulators, while the latter learned the teaching material. The *LDA-LAQ* was implemented as a discrete LA, the DP_{RI} scheme.

5. Evaluation

This section presents the experimental results obtained by testing the prototype implementation of the *LDA-LAQ*. To obtain these results, we performed numerous simulations to accurately simulate how the student modeler is able to recognize the learning model of the student.

The simulations were performed for many different types of environments. These environments represent the teaching "*problem*", which contains the uncertain course material of the domain model associated with the tutorial-*like* system. In all the tests performed, a LA algorithm was considered to have converged if the probability of choosing an action was greater than or equal to a threshold $T (0 < T \leq 1)$ and Q learning was considered to have converged if difference between $Q_{t+1}(a)$ and $Q_t(a)$ was greater than or equal to a threshold *T*, If the Q learning converged to the best action (i.e., the one with the highest probability of being rewarded), it was considered to have correctly converged.

As mentioned, the simulation has been performed for the different existing benchmark environments, for which the threshold T was set to 0.99 for LA and 0.00001 for Q learning , and the number of experiments $NE = 75$, in which the results were averaged. The results of these simulations are described below.

5.1 Environment With Four Actions

The four-action environment represents a multiple-choice question with four options. The student needs to learn the content of this question and conclude that the answer to this question is the action that possesses the minimum penalty probability. Based on the performance of the student, the *LDA-LAQ* is supposed to "*learn*" the learning model of the student. The results of this simulation are presented in Tables 2, 3. The two different settings for the two environments are specified by their reward probabilities given in Tables 2, 3. The results obtained were quite remarkable. Consider the case of EA, where the best action was $\alpha 1$.

In this paper we achieved upper accuracy from main model and with less iteration, determine student model, for instance in suggested model whereas fast learner with 34 iteration and 97% accuracy recognize correctly, but in main model number of iteration is 607. In normal learner with 28 iteration and 96% accuracy and in slow learner with 35 iteration and 100% accuracy reach a correct convergence.

The analogous results for EB were 100% (at its best accuracy) and 97% for the normal learner.

Env. E $_{(A,\ 4)}$	θ_1	θ_2	Fast learner		Normal learner		Slow learner	
			# of Iter.	*% correct convergence*	*# of Iter.*	*% correct Convergence*	*# of Iter.*	*% correct Convergence*
Meta LA [3]	0.50-0.31	0.60-0.35	607	100%	1037	87%	1273	91%
LDA LA/Q purposed	0.46-35	0.57-0.4	34	97%	28	96%	35	100%
Reward Probabilities	E$_A$= {0.7 0.5 0.3 0.2 }							

Table 2. Compare Convergence of Meta LA [3] And LDA-LAQ in E $_{(A,\ 4)}$

Env. E $_{(B,\ 4)}$	θ_1	θ_2	Fast learner		Normal learner		Slow learner	
			# of Iter.	*% correct convergence*	*# of Iter.*	*% correct Convergence*	*# of Iter.*	*% correct Convergence*
Meta LA[3]	0.28-0.17	0.37-0.24	790	97%	1306	89%	1489	91%
LDA LA/Q purposed	0.28-0.17	0.37-0.24	39	99%	44	92%	39	100%
Reward Probabilities	E$_A$= {0.1 0.45 0.84 0.76 }							

Table 3. Compare Convergence of Meta LA [3] And LDA-LAQ in E $_{(A,\ 4)}$

Env.E (*A*,10)	θ_1	θ_2	*fast learner*		*Normal learner*		*Slow learner*	
			# of Iter.	*% correct convergence*	*# of Iter.*	*% correct convergence*	*# of Iter.*	*% correct Convergence*
***Meta LA* [3]**	0.55-0.31	0.65-0.40	668	95%	748	93%	1369	96%
LDA LA/Q purposed	0.50-0.25	0.60-0.35	53	93%	36	97%	12	100%
Reward Probabilities	E$_A$= {0.7 0.5 0.3 0.2 0.4 0.5 0.4 0.3 0.5 0.2 }							

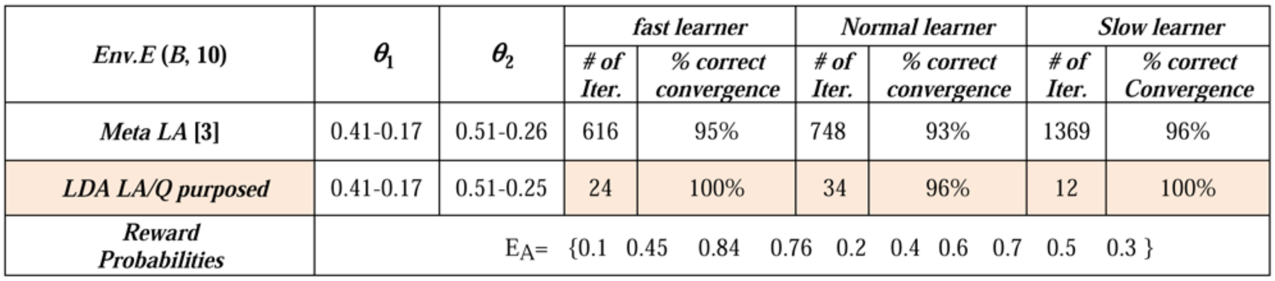
Table 4. Compare Convergence of Meta LA [3] And LDA- LA/Q in **E$_{(A,\ 10)}$**

5.2 Environment With Ten Actions

The ten-action environment represents one with multiple choice questions with ten options. The respective environments' reward probabilities are given in Tables 4, 5. Similar to the environments for example, consider the case of EA, where the best action was $\alpha 1$. While a slow learner converged in 12 iterations, normal learner converged in 36

Iterations and the fast learner converged in 53 iterations. By using the values of $\theta 1$ and $\theta 2$ as specified in Tables 4, 5 the *LDA-LAQ* was able to determine the true learning capabilities of the student 100% of the time, when slow learner.

For EB, the best accuracy was 100% for the fast leaner and slow learner, and the worst accuracy was 96% for the normal learner. Indeed, in a typical learning environment, the teacher may incorrectly assume the wrong learning model for the student.



Env.E (*B*, **10**)	θ_1	θ_2	*fast learner*		*Normal learner*		*Slow learner*	
			# of Iter.	*% correct convergence*	*# of Iter.*	*% correct convergence*	*# of Iter.*	*% correct Convergence*
***Meta LA* [3]**	0.41-0.17	0.51-0.26	616	95%	748	93%	1369	96%
LDA LA/Q purposed	0.41-0.17	0.51-0.25	24	100%	34	96%	12	100%
Reward Probabilities	E_A= {0.1 0.45 0.84 0.76 0.2 0.4 0.6 0.7 0.5 0.3 }							

Table 5. Compare Convergence of Meta LA[3] And LDA- LA/Q in $\mathbf{E}_{(\mathbf{B}, 10)}$

6. Conclusion

This paper presents a new model for tutorial like system which imply with student modeling. Student modeler use automata learner as an internal mechanism to determine student learning model, as it can be used in tutorial like system for learning exercise determination for each student. To reach this, student modeler uses higher level Automata which is called LDA-LAQ that can follow student simulation automata and teaching environment and try to determine student learning.

By means of Q Learning Algorithm for modeling various students, student this paper proves that learning model determination is highly speeded up and helps tutorial-like system for better student model determination. Simulations' results showed this method is a validated mechanism for student learning process execution and also showed student model is successful on student learning model determination and using Q Learning on students' modeling help for student learning model determination with lower iterations.

References

[1] Shute, V. J., Psotka, J. (1995). *Intelligent Tutoring Systems: Past, Present, and Future*, Handbook of Research on Educational Communications and Technology, Scholastic Publications.

[2] Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*, Los Altos, CA: Morgan Kaufmann Publishers, Inc.

[3] Hashem, M. K., Oommen, B. J. (2009). *Modeling a Student's Behavior in a Tutorial-Like System Using Learning Automata*, IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, IEEE.

[4] Fischetti, E., Gisolfi, A. (1990). *From computer-aided instruction to intelligent tutoring systems*, Educ. Technol.

[5] Winkels, R., Breuker, J. (1990). *What's in an ITS? a functional decomposition*, *In*: New Directions for Intelligent Tutoring ystems, Costa, E., Ed.Berlin, Germany: Springer-Verlag.

[6] Self, J. (1999). *The defining characteristics of intelligent tutoring systems research: ITSs care, precisel*, Int. J. Artif. Intell. Educ.

[7] Hashem, M. K. (2007). *Learning automata based intelligent tutorial-like systems*, Ph.D. dissertation, School Comput. Sci., Carleton Univ., Ottawa, Canada.

[8] Frasson, C., Mengelle, T., Aimeur, E., Gouarderes, G. (1996). *An actor-based architecture for intelligent tutoring system*, *In*: intelligent Tutoring system: 3rd international Conference, ITS '96: Lecture Notes in computer Science, p. 57-65. Springer-verlag.

[9] Lelouche, R. (2000). A collection of pedagogical agents for intelligent educational systems, *In*: Intelligent tutoring systems:6th international Conference, ITS 2000: Lecture Notes in computer Science.

[10] Legaspi, R. S., Sison, R. C. (2000). Modeling the tutor using reinforcement learning, *In*: Proc.

[11] Baffers, P., Mooney, R. (1996). Refinement-based student modeling and automated bug library construction, *J. Artif. Intell. Educ,* p. 75–116.

[12] Thathachar, M. A. L., Sastry, P. S. (2002). Varieties of learning automata: An overview, *IEEE Trans. Syst., Man, Cybern,* Dec.

[13] Sutton, R. S., Barto, A.C. (1998A). Reinforcement Learning: An introduvtion, MIT Press, Cambridge, MA, Bradford Book.

[14] Lakshmivarahan, S. (1981). Learning Algorithms Theory and Applications, New York: Springer-Verlag.

[15] Narendra, K. S., Thathachar, M. A. L. (1989). Learning Automata: An Introduction**.** Englewood Cliffs, NJ: Prentice-Hall.

[16] Meybodi, M. R., Lakshmivarhan, S. (1983). A Learning Approach to Priority Assignment in a Two Class M/M/1 Queuing System with Unknown Parameters, *In*: Proceedings of Third Yale Workshop on Applications of Adaptive System Theory, Yale University, p. 106-109.

[17] Meybodi, M. R., Beigy, H. (1998). *New Class of Learning Automata Based Scheme for Adaptation of Backpropagation Algorithm Parameters*, *In*: Proceedings of EUFIT-98, Achen, Germany.

[18] Johnson, D. S., McGeoch, L. A. (2002). *Experimental Analysis of Heuristics for the STSP*, in the Traveling Salesman Problem and its Variations, Gutin, G., Punnen, A., Editors, Kluwer Academic Publishers, Boston, p. 369-443.

[19] Hashem, M. K., Oommen, B. J. (2007). *On using learning automata to model a student's behavior in a tutorial-like system*, *In*: Okuno, H.G, Ali, M. (eds.) IEA/AIE 2007. LNCS (LNAI).

[20] Hashem, M. K., Oommen, B. J. (2009). Learning Automata Based Intelligent Tutorial-like System,Velásquez, J. D. et al. (Eds.): KES.

[21] Hashem, M. K., Oommen, B. J. (2007). Using learning automata to model a domain in a tutorial-like system, *In*: Proceedings of ICMLC 2007, the 2007 International Conference of Machine Learning and Cybernetics, Hong Kong, August.

[22] Thathachar, M. A. L., Oommen, B. J. (1979). Discretized reward-inaction learning automata, Spring.

[23] Oommen, B. J., Lanctôt, J. K. (1990). Discretized pursuit learning automata*, IEEE Trans. Syst., Man, Cybern.*, Jul./Aug.

[24] Thathachar, M. A. L., Oommen, B. J. (1979). Discretized reward-inaction learning automata, J. Cybern. Inf. Sci., Spring.

[25] Agache, M., Oommen, B. J . (2002). Generalized pursuit learning schemes: New families of continuous and discretized learning automata, *IEEE Transactions on Systems*.

[26] Kaisers, M., Tuyls, K., Parsons, S., Thuijsman,F. (2009). *An Evolutionary Model of Multi-agent Learning with a Varying Exploration Rate*, of 8th Int.Conf. on Autonomous Agents and Multiagent Systems, Decker, sichman, Sierra and Castelfranchi (eds.), , Budapest, Hungary. May, p. 10–15.