Artículo de Investigación. Revista Ciencia e Ingeniería. Vol. 46, No. 3 pp. 343-352, agosto-noviembre, 2025.
ISSN 1316-7081. ISSN Elect. 2244-8780 Universidad de los Andes (ULA)

Q-learning based algorithm for learning children's difficulties in multiplication tables

Algoritmo basado en Q-learning para el aprendizaje de las dificultades de los niños en las tablas de multiplicar

Pérez, Jesús[1*]; Aguilar, Jose [2,3]; Dapena, Eladio[4]
[1]LaSDAI, Universidad de Los Andes, Mérida, Venezuela.
[2]CEMISID, Universidad de Los Andes, Mérida, Venezuela.
[3]IMDEA Networks Institute, Leganés, Madrid, España.
[4]Universidad Intercontinental de la Empresa UIE, La Coruña, España.
* jesuspangulo@ula.ve

Abstract

In order to help children with their difficulties in multiplication tables, this paper presents a learning system based on the Q-learning algorithm, which allows software agents to learn what multiplication tables are difficult for a child. The results indicate that our learning system is effective to learn the children's levels of difficulties (low, medium, high, and very high) in the multiplication tables, when an agent randomly asks all operations of the multiplication tables (64 operations, since 2 until 9). In addition, our learning system allows knowing what multiplication tables have more difficulty with respect to the others, after asking at least once an operation for each multiplication table.

Keywords: Agent, social robot, Q-learning, reinforcement learning.

Resumen

Para ayudar a los niños con sus dificultades con las tablas de multiplicar, este artículo presenta un sistema de aprendizaje basado en el algoritmo Q-learning, que permite a los agentes de software aprender qué tablas de multiplicar son difíciles para un niño. Los resultados indican que nuestro sistema de aprendizaje es eficaz para aprender los niveles de dificultad de los niños (bajo, medio, alto y muy alto) en las tablas de multiplicar, cuando un agente pregunta aleatoriamente todas las operaciones de las tablas de multiplicar (64 operaciones, del 2 al 9). Además, nuestro sistema de aprendizaje permite saber qué tablas de multiplicar presentan mayor dificultad con respecto a las demás, tras preguntar al menos una operación por cada tabla.

Palabra clave: Agente, robot social, Q-learning, aprendizaje por refuerzo.

1 Introduction

The software agents are being used to help children in mathematical learning, such as: an agent asks deep questions on the learning material during a computer game (Pareto, 2014); an agent guides algebra lessons with prescriptive instructional guidance and anxiety treating messages (Kim *et al.*, 2016); children guide to the agent to play correctly a game (Axelsson *et al.*, 2013); and an agent integrated into a learning environment acts as a tutor and motivator (Mohammadhasani *et al.*, 2018).

Recently, the typical agents used are social robots. For example, children complete mathematical problems with robots (Ramachandran *et al.*, 2017); children solve tests of multiple-choice math questions while robots give verbal feedback (Brown & Howard, 2014); children participate in a game-based learning activity to learn arithmetic with the guidance of robots (Vrochidou *et al.*, 2018); children practice multiplication with problems that include whole numbers, while robots guide the practice (Liles, 2021).

In general, the agents as social robots have been shown to be effective for increasing cognitive and affective outcomes of children, achieving outcomes similar to those of human tutoring on restricted tasks

(Belpaeme *et al.*, 2018). In addition, the social robots can provide a personalized learning and social support (Michaelis & Mutlu, 2019). Therefore, the social robots are promising to continue helping children.

Particularly, in the context of mathematical learning, multiplication is a fundamental area in which many students manifest learning difficulties (Zhang *et al.*, 2016). According to Zhang *et al.* (2013), helping students with mathematics difficulties to develop sufficient multiplication problem-solving skills, is a priority for educators and researchers. It means that helping children with difficulties in multiplication tables, also known as times tables, is relevant.

According to Park *et al.* (2019), the current personalized education technologies are capable of delivering adaptive interventions that play an important role in addressing the needs of diverse young learners. In this context, children's difficulties in multiplication tables are different for every child, therefore, social robots need to learn the particular difficulties in multiplication tables for every child.

According to Castro-González *et al.* (2011), learning is an active area in robotic, and reinforcement learning is one of the learning methods that has been most successfully implemented in robots. Among the different solution methods, Q-learning is performing extremely well in the field of robotic (Jang *et al.*, 2019). For example, a robot uses Q-learning to select stories that are optimized for each child's linguistic skill progression (Park *et al.*, 2019). In our case, a Q-learning based algorithm will allow learning the particular difficulties in multiplication tables for every child.

In this paper, we present how a Q-learning based algorithm can be used for learning the children's difficulties in the multiplication tables. The paper is organized as follows: in section 2, we present the main concepts related to our proposition; in section 3, we describe our proposed learning system; in section 4, we show the experimental tests; and in section 5, we present the conclusions and future works.

2 Preliminary concepts

Reinforcement learning allows learning what to do through trial and error interactions with an environment. The task of reinforcement learning is to use observed rewards to learn an optimal policy for the environment (Aguilar, 2014; Russell & Norvig, 2016). The distinction between problems and solution methods is very important in reinforcement learning (Sutton & Barto, 2018): the problems are formalized using ideas from the dynamical system theory, for example, through the Markov Decision Processes (MDP); and the solution methods allow solving such problems.

In general, a learning agent interacts over time with its environment to achieve a goal, therefore, the learning agent must be able to perceive the state of its environment; take actions that affect the state; and receive rewards relating to the state. In this context, a MDP is composed of states, actions, transitions between states, and a reward function. Formally, a MDP is a tuple $<S,A,T,R>$ (Van Otterlo & Wiering, 2012), in which S is a finite set of states, A is a finite set of actions, T is a transition function defined as $T:S\times A\times S\rightarrow[0,1]$, and R is a reward function defined as $R:S\times A\times S\rightarrow R$. According to the problem, a MPD can have or not the transition function.

There are three types of systems that can be modelled by a MDP (Van Otterlo & Wiering, 2012): fixed horizon (tasks in which each episode consists of a fixed number of steps), indefinite horizon (tasks in which each episode can have an arbitrary length, but ends), and infinite horizon (tasks where the system does not end at all). An episode happens when the agent reaches a goal state and the process restarts in a new initial state; and a step is related to each action that the agent must take to reach a goal state. It means that a MDP can have one or more starting and goal states.

Given a MDP $<S,A,T,R>$, a policy is a computable function that outputs for each state $s\in S$ (except goal states) an action $a\in A$ (or $a\in A(s)$). Formally, a deterministic policy π is a function defined as $\pi:S\rightarrow A$ (Van Otterlo & Wiering, 2012). It is also possible to define a stochastic policy as $\pi:S\times A\rightarrow[0,1]$, such that for each state $s\in S$ (except goal states), it holds that $\pi(s,a)\geq 0$ and $\sum_{(a\in A)} \pi(s,a)=1$. Then, solving a given MDP means computing an optimal policy $\pi*$.

Solution methods for solving a MDP can be divided in several ways. On one side, there are two categories (Russell & Norvig, 2016): passive learning, where the agent's policy is fixed and the task is to learn the utilities of the states; and active learning, where the agent must learn the policy. In other side, there are three fundamental classes of methods (Sutton & Barto, 2018): dynamic programming, Monte Carlo methods, and temporal-difference learning. Also, the algorithms can be divided in two classes (Van Otterlo & Wiering, 2012): model-based, where a model of the MDP is known beforehand, and can be used to compute value functions and policies; and model-free, where the agent interacts with the environment, generating samples of state transitions and rewards, in order to estimate state-action value functions.

Q-learning method belongs to active learning, temporal-difference learning, and model-free algorithms. Jang *et al.* (2019) covered all variants of Q-learning algorithms, and they conclude that improved Q-learning algorithms might perform poorly while solving simple problems in a simple environment, but they outperform basic Q-learning algorithms when the problem at hand is complex and under a sophisticated environment. It means that a basic Q-learning shows an excellent learning ability in simple environments. In our case study, we will use a basic Q-learning because our problem is relatively simple (4 states and 8 actions).

Q-learning uses an off-policy method to separate the acting policy from the learning policy. The basic idea in Q-learning is to incrementally estimate Q-values for actions, based on rewards and the agent's Q-value function. Specifically, the agent makes a step in the environment from state s to s' using action a, and receives a reward R(s'); then, the update takes place on the Q-value of action a in the state s from which this action was executed. According to Watkins & Dayan (1992), equation for the Q-value is as follows (see equation 1):

$$Q(s,a) = (1- \alpha)\ Q(s,a) + \alpha\ [R(s') + \gamma\ \max_{a'} Q(s',a')] \qquad (1)$$

Equation 1 uses two hiperparameters: the learning rate α that determines the update rate, and the discount factor γ that determines the present value of future rewards. Formally, $\alpha \in [0, 1]$ and $\gamma \in [0, 1]$.

In active learning methods as Q-learning, the principal issue is the exploration (Russell & Norvig, 2016), because the agent must interact with the environment to learn by trial and error a correct policy. However, also the exploitation is used to get the right action that maximizes the expected reward on the one step (Sutton & Barto, 2018). First, the agent has to explore the environment by performing actions and perceiving their consequences through rewards, and then it can exploit the knowledge. In addition, if the environment is not stationary, the agent must explore to keep its policy up-to-date (Van Otterlo & Wiering, 2012). This naturally induces an exploration-exploitation trade-off, which has to be balanced to obtain an optimal policy.

3 Our learning system

3.1 Problem formulation

Social robots communicate through human-like interactions and must learn from these interactions. In this paper, social robots must learn children's difficulties in multiplication tables. In general, mathematical difficulties are determined by a low performance on tests of mathematics (Zhang *et al.*, 2013). Therefore, social robots must emulate tests through the interactions. A simple mathematic test consists in a social robot asking all multiplication tables to a child. It means that the problem will be formulated for one child, and social robots must have one model for every child.

In a mathematic test, a social robot must randomly ask all operations of the multiplication tables of one digit (from the multiplication table of 2 to the multiplication table of 9), and a child must answer it. In the problem formulation as a MDP, each multiplication table represents an action, and every answer represents a state. The idea is to reward every answer, in order to benefit negative answers, because in this way the policy π of the MDP will return actions (multiplication tables) with more difficulty to the child.

The states are qualifications related to difficulties: conscious difficulty (child knows that he doesn't know the correct answer), unconscious difficulty (child doesn't know that he doesn't know the correct answer), moderate difficulty (child knows the correct answer, but he is not sure), and without difficulty (child knows the correct answer). The rewards must mostly benefit conscious and unconscious difficulties; and it must less benefit the moderate difficulty.

The states can be determined indirectly combining two dichotomic parameters: speed of answer and validity. The speed of answer can be slow or fast, and the validity can be incorrect or correct. Specifically, a conscious difficulty is an answer slow and incorrect; an unconscious difficulty is an answer fast and incorrect; a moderate difficulty is an answer slow and correct; and without difficulty is an answer fast and correct.

Formally, the problem is formulated as a MDP, denoted as a tuple <A,S,R>, where A is the set of actions or multiplication tables {multiplication table of 2, multiplication table of 3, multiplication table of 4, multiplication table of 5, multiplication table of 6, multiplication table of 7, multiplication table of 8, multiplication table of 9}; S is the set of states or answer's qualifications {conscious difficulty, unconscious difficulty, moderate difficulty, without difficulty}; and R is the reward function as follows (see equation 2):

$$\boldsymbol{R(s)} = \begin{cases} \mathbf{0.99} & \boldsymbol{if\ s = \ conscious\ difficulty} \\ \mathbf{0.66} & \boldsymbol{if\ s = unconscious\ difficulty} \\ \mathbf{0.33} & \boldsymbol{if\ s = \ moderate\ difficulty} \\ \mathbf{0} & \boldsymbol{if\ s = \ without\ difficulty} \end{cases} \qquad (2)$$

The problem is modelled as an indefinite horizon, because each episode must end, but episodes can have arbitrary length. Each episode starts in without difficulty state (initial state) and the goal is to reach a conscious difficulty state or unconscious difficulty state (goal states). In addition, there is a fixed number of operations (steps) to end the test. The total operations are 64 (8 operations for each multiplication table) and the idea is to learn the policy for the formulated problem taking these operations, such that a social robot can know the multiplication tables that are difficult for a child.

3.2 Solution method

The formulated problem in the previous section has not a transition function because the problem is a model-free. Also, the problem does not have a policy, which means that it is an off-policy problem. The solution method used for this kind of problems is Q-learning, which goal is estimate Q-values. In our problem, the Q-values will allow knowing the difficulties in multiplication tables for every child.

Table 1. Learning algorithm

Step	Instruction
1.	Initialize Q(s,a) = 0
2.	s = initial state
3.	For each a in O:
3.1.	Take action a
3.2.	Observe s', R(s')
3.3.	Update Q(s, a) using equation 1
3.4.	s = s'
3.5.	If (s == goal state) then s = initial state

In the process of estimation of Q-values, our algorithm requires only exploration, because it must determine the end Q-values in the least amount of iterations. According to the amount of operations required for the test, we use a set of operations called O, which contains 64 operations (8 operations for each multiplication table). Because our problem was modelled like an indefinite horizon, we use an initial state, but several goal states in the algorithm. The algorithm used is shown in Table 1.

In lines 1 and 2, the algorithm begins with all Q-values in zero and in a specific initial state (without difficulty). Then, each action belonging to the set of 64 operations O (line 3) is performed (line 3.1) and based on child answers (line 3.2), the Q-value is updated (line 3.3), a transition is made to a new state (line 3.4), and it is compared in order to restart the initial state when it reach a goal state (line 3.5). The 64 operations are randomly taken, and the algorithm ends when all operations are processed.

In our formulated problem, future rewards are not important because there is not interdependence among the states to reach a goal state. Therefore, according to Vlachogiannis & Hatziargyriou (2004), the value of the discount factor   is set to 0.005. On the other hand, the selection of the learning rate α for our algorithm will be fixed with helping of a sensitivity analysis, according to the best performance to our problem (see section 4). Finally, when the algorithm ends, the social robot knows the multiplication tables that are difficult for the child, through the Q-values, where the greatest values represent the multiplication tables with the greatest difficulties.

4 Testing

4.1 Experimental protocol

The experimental goal is to verify that our learning system allows learning children's difficulties in multiplication tables. The experiments will be successful when the Q-values indicate the multiplication tables that are difficult for the children. The experimental protocol is divided in four experiments: 1) convergence, in order to know the convergence Q-values; 2) sensitivity analysis of the learning rate α, in order to select the best value for our model; 3) learning fixed profile of two children to know how the model learns in two instances; and 4) learning of a variable profile of one child, in order to analyze how the model can adapt to variable performance of a child.

In each experiment, it is necessary to instance our model at least one time, and to simulate the children answers. The children answer is simulated with profiles, where each profile is related to the child's knowledge about each multiplication table. For that, we specified two attributes: difficulty level (low, medium, high, and very high) of each multiplication table, and answer's probabilities of each difficulty level.

The answer's probabilities for difficulty levels try to simulate that in each level of difficulty there is an answer with higher probability, but it can be another. In Table 2, we present the probability distribution, where the low difficulty level means that the child can answer without difficulty 80% of the time, with moderate difficulty 10% of the time; and the rest is distributed between unconscious difficulty (5% of the time) and conscious difficulty (5% of the time). The others levels in Table 2 can be interpreted in the same way.

Table 2. Answer's probabilities of each difficulty level

Difficulty	P(cd)	P(ud)	P(md)	P(wd)
Low	0.05	0.05	0.1	0.8
Medium	0.05	0.05	0.8	0.1
High	0.1	0.8	0.05	0.05
Very High	0.8	0.1	0.05	0.05

cd = "conscious difficulty" state, ud = "unconscious difficulty" state, md =

"moderate difficulty" state, and wd = "without difficulty" state.

In order to carry out the experiments, we define three profiles for the children: bad (it simulates a child with bad performance in the tests of the multiplication tables), regular (it simulates a child with medium performance), and good (it simulates a child with good performance). In Table 3, we present the profiles mapped to each multiplication table, where bad profile means that high difficulty is mapped to multiplication tables of 2, 3, 4 and 5; and very high difficulty level with multiplication tables of 6, 7, 8 and 9. The others profiles in Table 3 can be interpreted in the same way.

The profiles are used in the four experiments as follows: in the convergence and the sensitivity analysis of the learning rate, we use the regular profile because it allows analysing how our model learns each level of difficulty; in the learning fixed profile of two children, we use the bad and good profiles to train one instance of our model for each child; and in the learning of a variable profile of one child, we use the bad and good profiles in only one instance of our model, in order to simulate a child with an initial bad performance and later it changes to a good performance.

Table 3. Difficulty level of each multiplication table in the three children profiles

Multipl. Table	**Profile**		
	Bad	**Regular**	**God**
2	High	Low	Low
3	High	Low	Low
4	High	Medium	Medium
5	High	Medium	Medium
6	Very High	High	Low
7	Very High	High	Low
8	Very High	Very High	Medium
9	Very High	Very High	Medium

4.2 Results

4.2.1 Convergence

Q-learning allows convergence in Q-values after several iterations (Watkins & Dayan, 1992). The goal of this experiment is to know the Q-values of convergence for each level of difficulty. For that, the model is instanced using the regular profile (see Table 3), and according to Castro-González *et al.* (2011), the rate learning (α) is set to 0.3. Then, we run our learning algorithm for 256 operations (32 random operations for each multiplication table, using a uniform distribution). In the Figure 1, we show the average Q-values for each step (operation) in 100 runs, and we appreciate that the four levels of difficulty converge after the step 100 in Q-values close to: 0.13 for low level of difficulty; 0.35 for medium level of difficulty; 0.63 for high level of difficulty; and 0.86 for very high level of difficulty. These results follow our goal, the Q-values of the states that represent the greatest difficulty for children are the biggest.

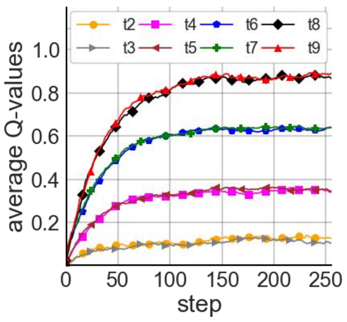

Figure 1. Average Q-values of initial state for a regular profile, using α = 0.3 and 256 steps

In order to get a better idea of the convergence of Q-values, in Figure 2 we show the standard deviations of the average Q-values presented on the Figure 1. It allows checking that the four levels of difficulty converge after the step 100, being the standard deviations between 0.06 and 0.13. These results are the references to use our learning system, however, we need to work with a smaller number of steps (ideally 64 steps because this cover the multiplication test).

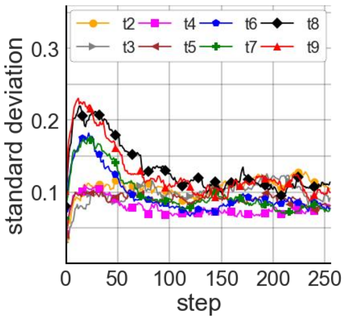

Figure 2. Standard deviations of average Q-values of initial state for a regular profile, using α = 0.3 and 256 steps

4.2.2 Sensitivity analysis

The goal of this experiment is to understand the relationships between an input and the outputs of our model, in order to select the appropriate value of the input, according to the expected behavior of our model. The input of interest is the learning rate (α), the outputs of interest are the end Q-values, and the expected behavior is that end Q-values of the initial state (without difficulty) in the last operation (operation 64) indicates the multiplication tables that are difficult for a child (specified in a profile).

For that, our model is instanced using the regular profile, we test our learning algorithm on 100 runs for different values of α, and the results are shown in the Table 4. The end Q-values shown are average of 100 runs with its standard deviations. Because the regular profile has associated all levels of difficulty (low, medium, high, and very high), in Table 4 we show the results of only one multiplication table (which is representative) for each difficulty level (for example, multiplication table of 3 for low difficulty).

In Table 4, we appreciate that the learning rate (α) has an incidence in the end Q-values. In general, α represents how much the Q-values are updated; it means that higher values of α allow faster convergence of the Q-values. That behavior can be appreciated in our results. For example, the results for the smallest α tested (0.1) have the smallest standard deviations (between 0.05 and 0.08), but they have far Q-values with respect to the convergence values; and the results for the highest α tested (1) are close Q-values with respect to the convergence values, but they have high standard deviations (between 0.13 and 0.29).

Since the learning rate 0.4 allows getting Q-values close to the convergence values with standard deviation between 0.10 and 0.14, we will select this learning rate for our algorithm. In order to understand better the behavior of our model while is running our algorithm, we show in the Figure 3 the average Q-values for each step (operation) of the multiplication test (64 operations). In Fig 3, we appreciate that multiplication tables with low level of difficulty have end Q-values close to 0.11 (multiplication tables off 2 and 3, see Table 3); multiplication tables with medium level of difficulty have end Q-values close to 0.32; multiplication tables with high level of difficulty have end Q-values close to 0.60; and multiplication tables with very high level have end Q-values close to 0.82.

Table 4. Sensitivity analysis of learning rate

α	Low (T3)		Medium (T5)		High (T7)		Very High (T9)	
	$\bar{Q}$	σ	$\bar{Q}$	σ	$\bar{Q}$	σ	$\bar{Q}$	σ
0.1	0.06	0.05	0.17	0.05	0.31	0.05	0.42	0.08
0.2	0.09	0.09	0.25	0.06	0.48	0.07	0.63	0.11
0.3	0.11	0.11	0.30	0.07	0.57	0.09	0.75	0.13
0.4	0.11	0.12	0.32	0.10	0.60	0.10	0.82	0.14
0.5	0.12	0.16	0.33	0.11	0.64	0.11	0.83	0.16
0.6	0.13	0.18	0.34	0.12	0.64	0.14	0.88	0.16
0.7	0.13	0.22	0.36	0.13	0.62	0.17	0.87	0.18
0.8	0.11	0.18	0.35	0.14	0.66	0.17	0.89	0.21
0.9	0.15	0.27	0.34	0.16	0.63	0.16	0.85	0.23
1	0.14	0.29	0.31	0.13	0.65	0.19	0.88	0.26

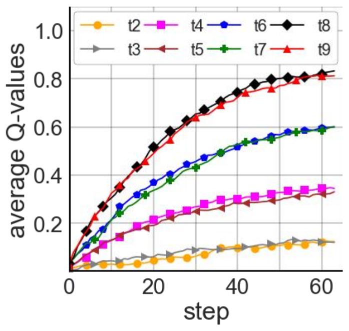

Figure 3. Average Q-values of initial state for a regular profile, using α = 0.4 and 64 steps

4.2.3 Learning fixed profile of two children

The goal of this experiment is to analysis the behavior of our learning system with the basic test of multiplication tables. For that, we create two instances of our model in order to test with two profiles (each profile simulates one child). The first profile is bad, and the second one is good (see Table 3). In Figure 4, we show the results of the bad profile, where can be appreciated that the end Q-values (step 64) are close to the convergence Q-values of our model: the multiplication tables of 2, 3, 4 and 5 have end Q-values close to 0.60; and the multiplication tables of 6, 7, 8 and 9 have end Q-values close to 0.82. It means that a social robot could know the specific multiplication tables that are difficult for a child of bad performance, and in addition, it could know the levels of difficulty of each one, making comparisons with the convergence Q-values.

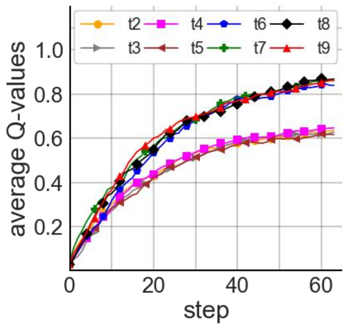

Figure 4. Average Q-values of initial state for a bad profile, using α = 0.4 and 64 steps

In Figure 5, we show the results of the good profile, where can be appreciated that the end Q-values are close to the convergence Q-values of our model: the multiplication tables of 2, 3, 6 and 7 have end Q-values close to 0.11; and the multiplication tables of 4, 5, 8 and 9 have end Q-values close to 0.32. In this case, the end Q-values are close to the convergence Q-values. In this way, a social robot could know the specific multiplication tables that are not difficult for a child with good performance, and knows the levels of difficulty of each one, making comparisons with the convergence Q-values.

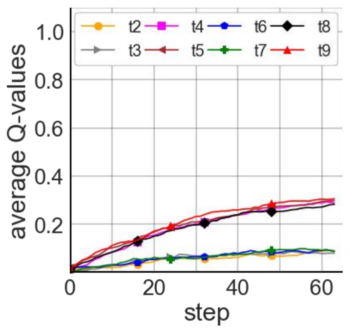

Figure 5. Average Q-values of initial state for a good profile, using α = 0.4 and 64 steps

4.2.4 Learning of a variable profile of one child

The goal of this experiment is to analysis the behavior of our learning system when the basic test of multiplication tables is applied to a child who has changed his profile. For that, we use the bad and good profiles in the same instance of our model. When we run our learning system with the bad profile, we get the results as is shown in Figure 4, and later, when we run it with the good profile, we get the results that are shown in Figure 6, where can be appreciated that the initial Q-values are the same end Q-values of the Fig 4. These Q-values are modified until the last step to get the end Q-values, which are close to 0.4 for multiplication tables with medium level of difficulty and are close to 0.2 for multiplication tables with low level of difficulty. It means that the Q-values in this case are far to the convergence Q-values, in comparison with the other cases, however, a social robot could know which multiplication tables are more difficult for a child in comparison with the other multiplication tables, through the Q-values. In general, our learning system follows the change of profile of the child.

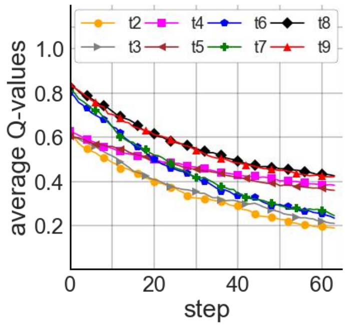

Figure 6. Average Q-values of initial state from bad to good profile, using α = 0.4 and 64 steps

5 Conclusions

In our learning system, the Q-values converge after of the step 100 as follows (see Figure 1): 0.13 for multiplication tables with low level of difficulty; 0.35 for multiplication tables with medium level of difficulty; 0.63 for multiplication tables with high level of difficulty; and 0.86 for multiplication tables with very high level of difficulty. These Q-values have a standard deviation between 0.06 and 0.13, and can be used to get the level of difficulty of each multiplication table in a given context. However, in our case of study, 100 steps (it means that social robot asks 100 operations to the child) is so much. One option is to use the Q-values before the step 100, but implies a higher standard deviation (see Figure 2). Because a basic multiplication test consists in asks one time all operations of the multiplication tables, 64 operations are acceptable for our case study.

The sensitivity analysis for different values of the learning rate, allows appreciating that for the highest values of the learning rate, the Q-values are close faster to convergence Q-values, but the standard deviation is higher: a learning rate of 0.1 has a standard deviation between 0.05 and 0.08, and a learning rate of 1 has a standard deviation between 0.13 and 0.29. A learning rate of 0.4 is an equilibrium between the fast convergence and the standard deviation values, and allows getting Q-values close to the convergence Q-values in the step 64, which is acceptable for our case study.

Our learning system allows knowing the multiplication tables that are difficult or not for children with several profiles: regular (see Figure 3), bad (see Figure 4) and good (see Figure 5). This is possible comparing the end Q-values with the convergence Q-values as follows: the end Q-value of a child has the same level of difficulty in that multiplication table as the nearest convergence Q-value. In this sense, an end Q-value nearest to 0.13 means a low level of difficulty; nearest to 0.35 means a medium level of difficulty; nearest to 0.63 means a high level of difficulty; and nearest to 0.86 means a high level of difficulty.

Our learning system allows knowing the multiplication tables that are difficult for a child that change his profile. In general, our learning system is able to know which multiplication tables are more difficult for a child after asks him at least one operation for each multiplication table in all the case studies.

There are several future works. A first future work must extend this study with information of the context, like more actions guided by the educational paradigms and based on the learning profiles of the children. Also, another future work must analyze the sorting of the Q-values to get the order of difficulty, which can be useful for a social robot in order to help children through specific learning strategies adequate to them. For example, once the system knows which multiplication tables are more difficult for the child, which policy of actions to follow in order to reduce that difficulty. Some of the possible results would be practice sessions of all the multiplication tables, making emphasis on the ones that the child has more difficulty without knowing, or fun strategies to reduce the difficulty without asking the multiplication tables.

Also, in the field of social robotics, it will be necessary to include other actions linked to the body of the robot, such as its mobility and expressions (non-verbal language like emotional expressions through the face (Pérez *et al.*, 2020)), to make the learning process more harmonious. Finally, the article has used two parameters to determine the status (response time of the child, and validity of the response). A future work should interpret human characteristics (for example, affective states (Pérez *et al.*, 2024)) when asked, to enrich the model because it gives evidence of knowledge.

References

Aguilar, J. (2014). Introducción a los Sistemas Emergentes. Talleres Gráficos, Universidad de Los Andes: Venezuela.

Axelsson, A., Anderberg, E., & Haake, M. (2013). Can Preschoolers Profit from a Teachable Agent Based Play-and-Learn Game in Mathematics? In Lecture notes in computer science (pp. 289–298). https://doi.org/10.1007/978-3-642-39112-5_30

Belpaeme, T., Kennedy, J., Ramachandran, A., Scassellati, B., & Tanaka, F. (2018). Social robots for education: A review. Science Robotics, 3(21). https://doi.org/10.1126/scirobotics.aat5954

Brown, L., & Howard, A. (2014). The positive effects of verbal encouragement in mathematics education using a social robot. In 2014 IEEE integrated STEM education conference (pp. 1-5). http://doi.org/10.1109/ISECon.2014.6891009

Castro-González, Á., Malfaz, M., & Salichs, M. A. (2011). Learning the selection of actions for an autonomous social robot by reinforcement learning based on motivations. International Journal of Social Robotics, 3(4), 427–441. https://doi.org/10.1007/s12369-011-0113-z

Jang, B., Kim, M., Harerimana, G., & Kim, J. W. (2019). Q-Learning Algorithms: A Comprehensive Classification and applications. IEEE Access, 7, 133653–133667. https://doi.org/10.1109/access.2019.2941229

Kim, Y., Thayne, J., & Wei, Q. (2016). An embodied agent helps anxious students in mathematics learning. Educational Technology Research and Development, 65(1), 219–235. https://doi.org/10.1007/s11423-016-9476-z

Liles, K. R. (2019). Ms. An (Meeting Students' Academic Needs): Engaging Students in Math Education. Lecture Notes in Computer Science, 645–661. https://doi.org/10.1007/978-3-030-22341-0_50

Michaelis, J. & Mutlu, B. (2019). Supporting interest in science learning with a social robot. In Proceedings of the 18th ACM international conference on interaction design and children (pp. 71-82). https://doi.org/10.1145/3311927.3323154

Mohammadhasani, N., Fardanesh, H., Hatami, J., Mozayani, N., & Fabio, R. A. (2018). The pedagogical agent enhances mathematics learning in ADHD students. Education and Information Technologies, 23(6), 2299–2308. https://doi.org/10.1007/s10639-018-9710-x

Park, H. W., Grover, I., Spaulding, S., Gomez, L., & Breazeal, C. (2019). A Model-Free affective reinforcement learning approach to personalization of an autonomous social robot companion for early literacy education. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01), 687–694. https://doi.org/10.1609/aaai.v33i01.3301687

Pareto, L. (2014). A teachable agent game engaging primary school children to learn arithmetic concepts and reasoning. International Journal of Artificial Intelligence in Education, 24(3), 251–283.https://doi.org/10.1007/s40593-014-0018-8

Pérez, J., Aguilar, J., & Dapena, E. (2020). MIHR: a Human-Robot Interaction Model. IEEE Latin America Transactions, 18(09), 1521–1529. https://doi.org/10.1109/tla.2020.9381793

Pérez, J., Dapena, E., & Aguilar, J. (2024). Emotions as implicit feedback for adapting difficulty in tutoring systems based on reinforcement learning. Education and Information Technologies, 29(16), 21015–21043. https://doi.org/10.1007/s10639-024-12699-8

Ramachandran, A., Huang, C. M., & Scassellati, B. (2017). Give me a break! Personalized timing strategies to promote learning in robot-child tutoring. In Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction (pp. 146-155). https://ieeexplore.ieee.org/document/8534798

Russell, S. J., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall Press: Upper Saddle River, USA.

Sutton, R. & Barto, A. (2018). Reinforcement Learning: An Introduction. A Bradford Book: USA.

Van Otterlo, M., & Wiering, M. (2012). Reinforcement learning and Markov decision processes. In Adaptation, learning, and optimization (pp. 3–42). https://doi.org/10.1007/978-3-642-27645-3_1

Vlachogiannis, J., & Hatziargyriou, N. (2004). Reinforcement learning for reactive power control. IEEE Transactions on Power Systems, 19(3), 1317–1325. https://doi.org/10.1109/tpwrs.2004.831259

Vrochidou, E., Najoua, A., Lytridis, C., Salonidis, M., Ferelis, V., & Papakostas, G. A. (2018). Social robot NAO as a self-regulating didactic mediator: A case study of teaching/learning numeracy. In 2018 26th international conference on software, telecommunications and computer networks (SoftCOM) (pp. 1-5). https://doi.org/10.23919/softcom.2018.8555764

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. Machine Learning, 8(3), 279–292. https://doi.org/10.1007/bf00992698

Zhang, D., Ding, Y., Lee, S., & Chen, J. (2016). Strategic development of multiplication problem solving: Patterns of students' strategy choices. The Journal of Educational Research, 110(2), 159–170. https://doi.org/10.1080/00220671.2015.1060928

Zhang, D., Xin, Y. P., Harris, K., & Ding, Y. (2013). Improving multiplication strategic development in children with math difficulties. Learning Disability Quarterly, 37(1), 15–30. https://doi.org/10.1177/0731948713500146

Received: *December 28th, 2024*

Accepted: *March 22th, 2025*

Pérez, Jesús: *Postdoctorado en Investigación Educativa (2022), Dr. en Ciencias de la Educación (2019), MSc. en Educación Superior (2014), Ing. de Sistemas (2014), Ing. en Electrónica (2012). Es Profesor Agregado en la Universidad de Los Andes (ULA), e Investigador del Laboratorio de Sistemas Discretos, Automatización, e Integración (LaSDAI). https://orcid.org/0000-0002-6585-2648*

Aguilar, Jose: *Ingeniero de Sistemas en 1987 (ULA), MSc. en Ciencias de la Computación en 1991 (Universite Paul Sabatier-Francia), Doctorado en Ciencias de la Computación en 1995 (Universite Rene Descartes-Francia) y Postdoctorado en Ciencias de Computación en 2000 (Universidad de Houston). Actualmente es investigador Senior en IMDEA Networks Institute. Email: aguilar@ula.ve https://orcid.org/0000-0003-4194-6882*

Dapena, Eladio: *Doctor Ingeniero Industrial, UC3M, España (2002). Automatización Industrial (Esp.), UFSC, Brasil (1998). Ingeniería de Sistemas, ULA, Venezuela (1990). Profesor jubilado Universidad de Los Andes (ULA). Profesor Titular Universidad Intercontinental de la Empresa UIE. Email: eladio.dapena@uie.edu https://orcid.org/0000-0002-9135-0967*