2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 1-4, 2023, Oahu, Hawaii, USA

Reinforcement Learning with Experience Sharing for Intelligent Educational Systems

Ryan Hare
Dept. of Electrical and Computer Engineering
Rowan University
Glassboro, NJ, USA
harer6@rowan.edu

Ying Tang
Dept. of Electrical and Computer Engineering
Rowan University
Glassboro, NJ, USA
tang@rowan.edu

2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 979-8-3503-3702-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/SMC53992.2023.10394095

***Abstract*—With higher education pushing toward larger class sizes, a large portion of current methodology focuses on one-size-fits-all approaches that can effectively educate a large class. However, when these approaches fail, students can be left behind and fail classes due to simple misunderstandings. Inspired by these issues, this paper proposes a modular reinforcement learning system that can be used in intelligent educational systems to inform personalized student support. Based on a similar method detailed in prior work, this paper proposes experience sharing with tutor agents as a computationally light approach to improve reinforcement learning training speed on the task of student support. We also provide preliminary results obtained from student simulations to demonstrate the effectiveness of the proposed method on reinforcement learning agent performance.**

***Index Terms*—intelligent systems, educational technology, reinforcement learning**

I. Introduction

Recent advancements in artificial intelligence (AI) and machine learning (ML) have opened up new potential in the field of personalized education and automated student support [1]–[3]. A large portion of current educational methodology focuses on one-size-fits-all content that can lead to poor learning outcomes when students fall significantly behind, or have preferred learning methods that differ from the general-purpose approaches. Compared to general-purpose approaches, personalized education has been demonstrated as an effective way to improve student education and engagement with subject matter [4], [5], especially when coupled with serious games and intelligent tutoring systems [6], [7]. Designing effective personalized lessons, however, is a time-consuming process for educators. Furthermore, determining what assistance a student requires is also difficult, especially with large class sizes and limited instructor resources.

In our research, reinforcement learning (RL) has been of particular interest for providing personalized recommendations to students within educational software due to its ability to automatically learn new and optimized behaviors based on prior interactions with students. Inspired by other recent successes in using reinforcement learning for educational recommendations [8], improving student engagement [9], tutorial-like systems [10], and intelligent tutoring [11], we apply RL to the task of personalized student assistance. By using RL to add personalized support and recommendations into these types of educational systems, the proposed approach aims to augment existing technologies and lead to more effective education. Additionally, the proposed system is designed to be modular and easily extensible, allowing for any number of topics or any system to interface and receive personalized recommendations. Through reinforcement learning [12], the system automatically learns new behavior on added topics.

Our developments in this paper are driven by both the aforementioned need for more personalized educational software through serious games and intelligent tutoring systems and our findings from our prior work [13]. In our prior exploration, we detailed a multi-agent reinforcement learning approach to personalized student assistance that applied experience sharing to boost RL agent performance [13]. While our prior results were promising, the system showed issues when adding a large number of agents due to the computational intensity of the experience sharing algorithm. Simply put, with too many agents, it became impossible to conduct RL updates in real-time. In this paper, we present a more modular system that can work with a larger number of agents on a large variety of domains, leveraging experience sharing to boost overall system performance.

The proposed approach focuses on training a set of RL agents each of which deals with a specific narrow domain or topic. For example, in a system focused on mathematics, specific agents would be designated to deal with addition, subtraction, and multiplication. Despite their disparate domains however, each agent operates on the same task of observing student performance and providing appropriate assistance. As stated, we use experience sharing as a form of transfer learning [14] to improve data efficiency and increase the learning speed of all agents by allowing them to share past observations. To address the concerns raised in our prior work, this paper uses a method we refer to as experience sharing with tutor agents, allowing each RL agent to store a

This paper is supported in part by the National Science Foundation (NSF) under grant #1913809, and by the U.S. Department of Education Graduate Assistance in Areas of National Need (GAANN) under grant #P2000A180055.

set of similar "tutors" that it can sample experience from. In doing so, the number of required similarity calculations is greatly reduced, decreasing the computational intensity of the method while maintaining comparable performance.

II. System Design & Proposed Methods

This section discusses our proposed methods and RL agent configurations. Rather than use a single reinforcement learning agent, we use a set of N agents that each operate on an individual task. We separate out the overall learning space to limit the size of both the state and action spaces, and to create agents that learn specialized behavior for their individual tasks. However, overall performance is still limited in this format, so we use weighted experience sharing, allowing agents to share their experience with each other to boost training while weighing the experience to account for differences in system dynamics between agents. Then, in this paper, we propose experience sharing with tutor agents to address the computational complexity of the experience sharing weight calculations to allow for real-time computations.

A. Reinforcement Learning

Reinforcement learning (RL) is a machine learning method where automated systems can learn optimal behavior through interactions with the environment [12]. RL is typically a cycle of behavior wherein the automated agent observes a current environment state. From that state observation, the agent selects some action and ends up in a new state. At this point, the environment provides the agent with a numerical reward. The end goal of reinforcement learning is to learn a mapping function, often called a policy, that maps any environment state to the best possible action that maximizes the agent's immediate and future rewards [12].

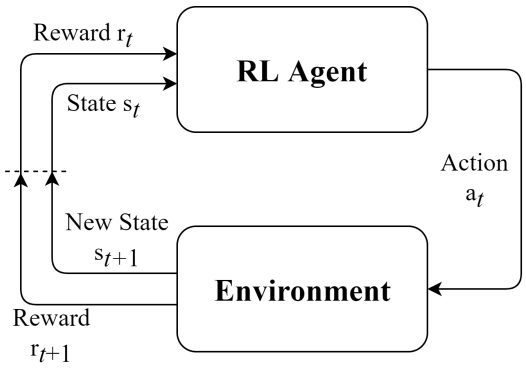

Fig. 1. The standard cycle of reinforcement learning.

RL agents follow a typical cycle of state, action, next state, reward, as shown in Figure 1. In this paper, a tuple containing a state, action, resulting state, and reward is referred to as an experience tuple since it represents an agent's past observation. RL environments also assume the Markov property, where future state probabilities only depend on the current state. Formally, RL problems are defined as Markov decision processes (MDP) in the form $< S, A, R, P, \pi >$:

1) S is the state space such that any state $s \in S$ is a valid state that the agent could observe. Often, the observed state at a given timestep is denoted s_t, and the resulting state upon taking an action is denoted s_{t+1}.
2) A is the action space defined such that any action $a \in A$ is a valid action that the agent can select. The action selected upon observing state s_t is denoted a_t.
3) R is the reward function defined as $R(s, a, s')\forall s, s' \in S, a \in A$. Upon observing a state s_t, selecting an action a_t, and arriving at state s_{t+1}, the agent receives reward $r = R(s_t, a_t, s_{t+1})$.
4) P is the transition probability function defined as $P(s, a, s')\forall s, s' \in S, a \in A$ that determines the probability of arriving in state s' given that action a was taken from state s. P is difficult to know or predict as it is often determined by unknown environmental factors.
5) π is the agent policy defined as $\pi(s|a)$ which determines the probability of selecting action $a \in A$ given that state $s \in S$ was observed. Learning π^*, the optimal policy, is the goal of the RL agent.

B. Proposed Agent Structure

In the proposed work, we assume a modular system that contains N RL agents that each operative on a specific subject. As mentioned previously, these subjects are meant as discrete subsets of knowledge that combine to comprise an overall learning space. For example, a system that focuses on teaching basic physics might have a single agent that operates on the property of friction and another agent that deals with force. Compared to a single-agent approach, these N agents are designed to share their knowledge to improve overall learning speed, and the specifics of the MDP for the problem of educational support is discussed below. Furthermore, a single-agent approach would require a larger state/action space to represent the various subjects, while each agent in a multi-agent approach has a smaller overall learning space, leading to quicker learning and better performance.

To support the experience sharing discussed in section II-D, the agents are structured as follows: The agent state space can be represented by a vector of student data, $\mathbf{v}$ with $|\mathbf{v}|$ elements. Ideally, this student data should be representative of a student's knowledge in relevant subject matter. For example, variables in $\mathbf{v}$ could be score on a test, time taken to answer given questions, emotion estimates from external sensors, or other relevant educational data. We do assume that $|\mathbf{v}_m| = |\mathbf{v}_n|\forall m, n \in [1, N]$ in this work to enable experience sharing as discussed in Section II-D.

The agents actions, then, represent student support. However, to support experience sharing, the agents' actions need to be standardized between agents to maintain identical dimensionality. We could assume that $|A_m| = |A_n|\forall m, n \in [1, N]$, but that would put a restriction on the amount of help actions that could be created, and would leave some agents with "empty" actions where other agents have support to provide. To address this, the proposed system assumes that all agents' action spaces are identical, and that they represent a set of m_a properties that can inform an expert

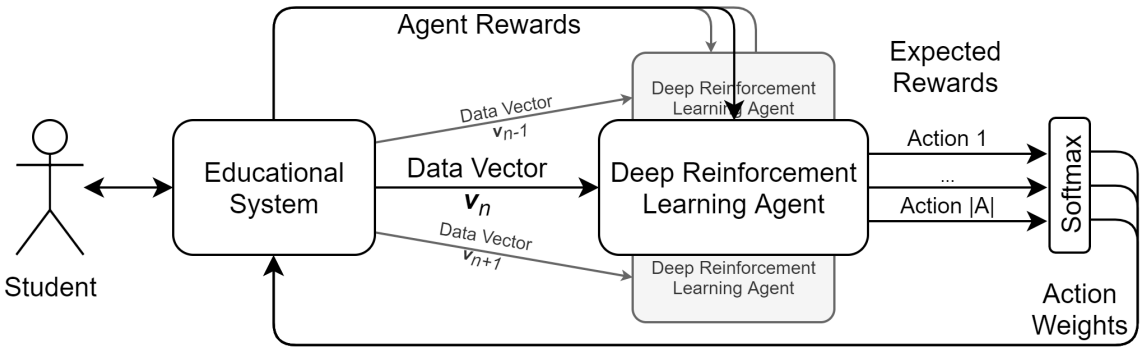

Fig. 2. The student support environment for multi-agent reinforcement learning.

or system on how best to provide assistance. For example, such properties could include weights for basic and advanced content, equations, images, videos, or even properties of how the content is written, such as factual versus emotional. Then, rather than taking the best action, the deep Q-network can output a weight in $[0.0, 1.0]$ for each action. This way, $|A_m| = |A_n|$ without requiring a strictly identical amount of possible help actions in each subject. The entire environment is visualized in Figure 2. Translating these weights into proper student support is beyond the scope of this paper, but recent technologies like ChatGPT [15] have made great strides in the field of dynamic content generation.

C. Deep Q-learning

To solve a Markov decision process, reinforcement learning focuses on determining an optimal policy $\pi^*(s, a)$ that the agent can follow to take the best possible action in every state. The optimal policy can typically be determined by selecting the action with the highest expected reward. Expected reward, in turn, is estimated based on recordings of past rewards. To estimate past rewards, agents focus on predicting the Q-function, $Q(s, a)$, which predicts the expected future reward in state s when selecting action a. In deep Q-learning, the Q-function is approximated by a neural network with weights θ_t [16]. The weights are then updated every time step when the agent obtains new experience. This method also makes use of a second target network to stabilize learning with weights ϑ_t, where $\vartheta_t \leftarrow \theta_t$ on a set interval of more than one time step [17]. The second network stabilizes predictions of future reward, which are used to compute the cost function that network training aims to minimize, as shown in Equation 1.

$$Cost \leftarrow Q(s_t, a_t|\theta_t) - [r_{t+1} + \gamma \max_a Q(s_{t+1}, a|\vartheta_t)] \quad (1)$$

A main part of Deep Q-learning that is utilized by the proposed method is experience replay [16]. Experience replay improves data efficiency in deep Q-learning by storing a replay memory of past experience, which we refer to as M_n for agent $n \in [1, N]$. Each transition stored in memory $\mathcal{T} = (s, a, s', r)$ stores a state $s \in S$, chosen action $a \in A$, resulting state $s' \in S$, and obtained reward $r = R(s, a, s')$. This memory can then be sampled at any time by the agent for training, with the agent sampling a batch of b transitions $B = \{\mathcal{T}_i\}_{i=0}^{b} = \{(s_i, a_i, s'_i, r_i)\}_{i=0}^{b}$. With the replay memory, past data are reused for training, helping to prevent the agent from "forgetting" past experiences. For our purposes, the experience replay serves as a pool of data for each agent that other agents can sample from, as discussed in the following section.

D. Weighted Mutual Experience Transfer

A major issue with reinforcement learning approaches to student assistance is the high data requirement to learn correct behavior. Due to this, the agents will have highly suboptimal performance in the early stages of training which may negatively impact student education. Furthermore, data are not readily available since gathering new interactions requires student participation. Since all agents essentially operate on the same task (provide assistance based on a student's performance), we apply experience sharing to augment the amount of training data. With experience sharing, agents can sample past observations from other agents to increase the amount of training data they have access to.

In our prior work, we detailed a method we refer to as mutual weighed experience sharing [13] This method was inspired by transfer learning methods in reinforcement learning [14], and in particular, learning from demonstrations [18]. With learning from demonstrations, RL agents can be pre-trained [19] or guided [20] by behavior created by a human expert on tasks that would otherwise be challenging to learn.

Unlike learning from demonstrations, which leverage expert knowledge to train agents, we instead use other agents' experiences to train. Thus, when training, agent $n \in [1, N]$ samples not just a batch $B_n = \{\mathcal{T}_i\}_{i=0}^{b}$ but also a batch $B_m \forall m \in \{1, 2, ..., n-1, n+1, ..., N\}$ from the other agents. This gives a final combined training batch B^* as shown in Equation 2. Furthermore, this sampling process is visualized in Figure 3.

$$B^* = B_n \cup \{B_m\} \forall \{m \in [1, N] : m \neq n\} \quad (2)$$

But since we assume that the transition probabilities are similar but not truly equal, we apply a training weight to de-

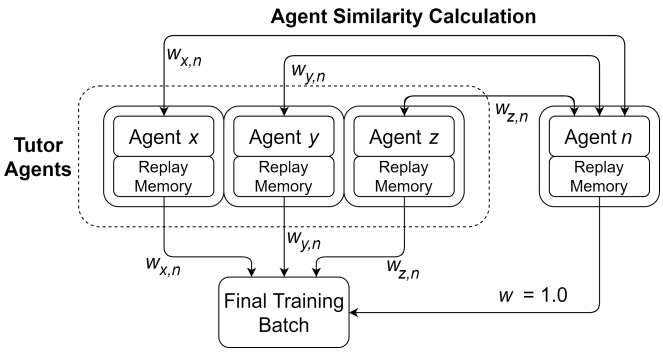

Fig. 3. Training batch creation in weighted experience transfer.

prioritize experience from other agents. The proposed method computes the centered kernel alignment (CKA) similarity [21] between the neural network representations that estimate the value functions in the deep reinforcement learning in each of the N agents. CKA is computed as shown in Equation 3, where K and L are kernel matrices derived from a randomly sampled batch of training data, and $HSIC$ is the Hilbert-Schmidt independence criterion that computes statistical dependence between the two matrices [22]. Then, the similarity between agent m and n can be defined as $w_{m,n}$, and the weighted training batch B^* shown in Equation 4. Once the weights are calculated, they can be passed in to the neural network to weigh those individual samples accordingly.

$$CKA(K, L) = \frac{HSIC(K, L)}{\sqrt{HSIC(K, K)HSIC(L, L)}} \quad (3)$$

$$B^* = (B_n, 1.0) \cup \{(B_m, w_{m,n})\} \forall \{m \in [1, N] : m \neq n\} \quad (4)$$

E. Experience Sharing with Tutor Agents

One issue with the aforementioned approach to experience sharing is the computational intensity of mutual sharing. As the number of agents N increases, there is a proportional increase in the computation time needed to compute the similarity values. Further, since similarity must be recomputed every time an agent is retrained, the reinforcement learning updates become impossible to conduct in real-time, as is required when integrating the system with educational software, even with as few as 10 agents. To ensure that the system is scalable and usable as an online learning approach, this paper proposes instead to maintain a list of i "tutor" agents per agent that each agent can sample training data from. Then, when retraining each agent, they only have to compute i similarity values while still gaining the benefits of mutual experience transfer.

However, this approach creates a new problem; how to update an agent's tutors to reflect changes in the environment. To address this issue, the agent must compute similarity values of non-tutor agents to ensure that its chosen tutors are still the most similar agents. To that end, the proposed method adds an additional step of sampling j additional agents from the total pool of agents. Similarity is computed between the i tutor agents and the j other agents, allowing the agent to select the i best agents from the pool to become the new tutors. The full steps for updating an agent's tutors are shown in Figure 4. To further improve performance, tutor updates can also be staggered from every training step to a set interval of steps.

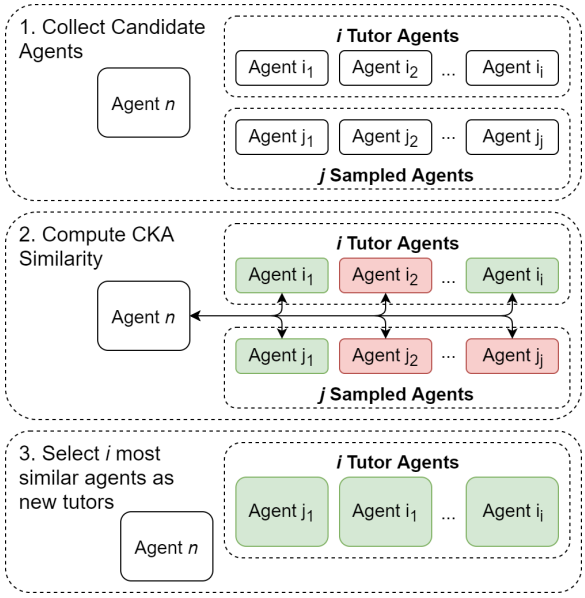

Fig. 4. Process of updating tutor agents for agent n

III. Preliminary Evaluation

A. Student Simulation

As mentioned in prior sections, student data are difficult to collect since each instance of data requires a new student to interact with an educational system. To verify the approaches presented in this work, we instead focus on simulating "student-like" behavior to provide a proof-of-concept of our proposed methods. In this simulated student environment, a virtual student interacts with our educational system similar to how a real student would; first, the system records a set of initial student data. In this case, a series of state variables. In comparison to a real environment, these variables would represent measurements such as score on a test, time taken to answer questions or solve problems, index of questions given, or emotion values estimated from a webcam. In practice, these variables are all float values in the range $[0.0, 1.0]$.

The actions in the simulation environment are set up as discussed in Section II-A. The agent's actions must then be able to influence the simulated student's performance in a predictable but varied way, as we would expect from a real student. To generate human-like responses, the simulated environment takes inspiration from behavior trees [23] through a process shown in Figure 5. At the beginning of testing, behavior trees are randomly generated to determine the simulated student's response in each subject. The behavior trees take in the student's state vector as an input, with the output

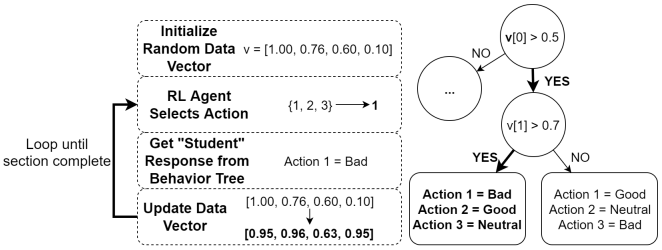

Fig. 5. Step-by-step process used to simulate student responses (left) and an example behavior tree (right). The example data vectors and behavior tree shown are significantly smaller than the real simulation for ease of viewing.

determining how the student's data changes in response to the agent's chosen action. In the leaf nodes of the tree, actions are rated on a scale from positive to negative, with positive actions leading to a student response that moves them closer to completing the section, while negative actions move them further away from completion. With this approach, the simulated responses are consistent between different tests. Additionally, to help mimic human responses, some random noise is also applied to ensure responses are similar but not entirely consistent, creating a more challenging environment for the RL agent to learn.

From these student responses, the agent is positively or negatively rewarded based on the data vector's proximity to a "passing" grade. When positive actions are chosen, the student's data moves toward completion, creating a positive reward for the agent. Negative actions, meanwhile, move the student further away, creating a negative reward. Finally, some actions fall in between positive and negative, meaning they could create a positive or negative reward with a smaller magnitude compared to positive and negative actions.

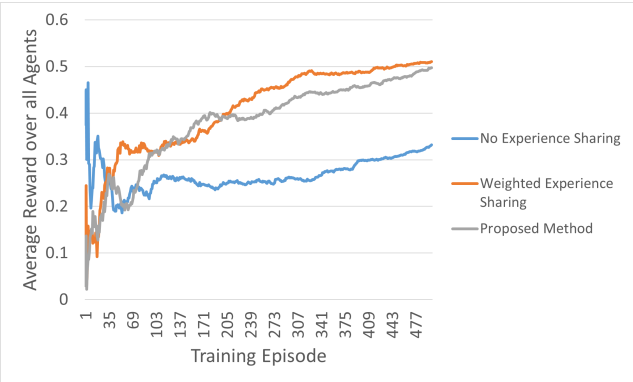

Fig. 6. Comparison of average reward over 10 agents between multi-agent, conditional experience sharing, and experience sharing with tutor agents, with 3 tutor agents.

B. Results

Evaluation of the proposed system mainly serves as a proof-of-concept before the system is implemented into an intelligent tutoring system or serious game. To that end, our evaluation focuses on verifying two aspects of the proposed method: 1) The proposed experience sharing method allows agents to perform more optimally in the early stages of training compared to a multi-agent method with no experience sharing; and 2) The proposed tutor sampling achieves comparable performance to mutual experience sharing despite its massively decreased computational expense.

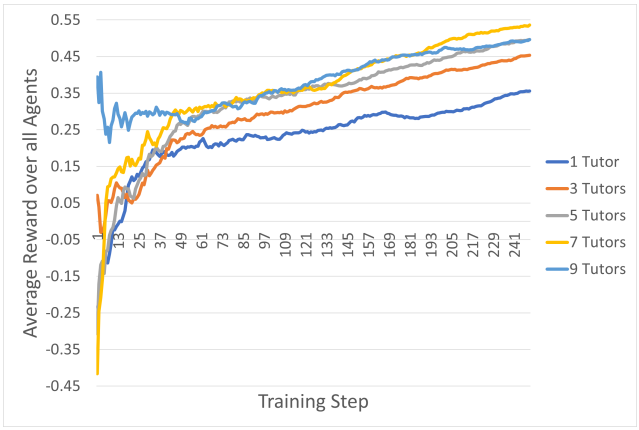

Fig. 7. Average cumulative reward for 10 agents comparing different numbers of tutor agents.

TABLE I
STATISTICAL COMPARISON OF TUTOR AGENT EXPERIMENT FROM THE MANN-WHITNEY U TEST.

Experiment	Effect Size	Interpretation
1 Tutor vs. 3 Tutors	0.373	Moderate improvement
3 Tutors vs. 5 Tutors	0.194	Small improvement
5 Tutors vs. 7 Tutors	0.109	Small improvement
7 Tutors vs. 9 Tutors	0.010	No change

Initial experiments were conducted with 10 agents to verify early system performance. All parameters were held constant between methods. As shown in Figure 6, both conditional experience sharing and experience sharing with tutors had improved performance when compared to a standard multi-agent approach. This improvement can be attributed to the similarity between the environments; while not identical, the environment dynamics are similar. Thus, agents that share experience are able to benefit from a much larger training pool even though the extra data may not be fully reliable.

Further experiments were conducted to show the effect of number of tutor agents on overall agent performance, as shown in Figure 7. In this experiment, 10 agents were used to test 1, 3, 5, 7, and 9 tutor agents. As shown, there are diminishing returns on number of tutor agents, and our results showed a significant improvement between 1 tutor and 3 tutors, with 3, 5, 7, and 9 tutors having no significant difference between them. Table I shows the effect sizes between experiments. This shows that a low number of tutors relative to the total number of agents achieves comparable performance to a high number with a significantly lower computational requirement.

Finally, to further show system scalability, Figure 8 shows agent performance as measured by average reward on 10, 20, and 30 agents. As shown, agent performance is consistent regardless of the number of agents used; in fact, system

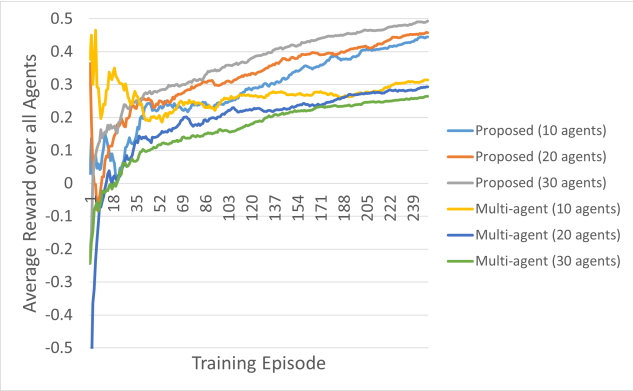

Fig. 8. Average agent reward over 250 training steps for 10, 20, and 30 agents and 3, 5, and 7 tutor agents, respectively, comparing the proposed method with a multi-agent approach without experience sharing.

performance tends to increase slightly with more agents due to each agent having a larger number of potential tutors.

IV. Conclusions

This paper presents a modular system for using reinforcement learning to create intelligent, automated educational systems. We've demonstrated that the proposed experience sharing method achieves comparable performance to our prior method with significantly decreased computational complexity. Furthermore, we've also demonstrated that the system is scalable and offers improved performance over a standard multi-agent approach. Overall, we hope that the proposed system provides a solid starting point for other researchers in this area to develop modular automated educational systems that can help students learn more effectively and efficiently with a lower resource cost to instructors.

As mentioned in Section II-A, there are a few constraints placed on the system that could be relaxed in future work. Mainly the constraint of assuming that each agent has an identical state space. This assumption helps to enable experience sharing, but it may not be viable for larger systems that collect a wider variety of student data. Furthermore, it may be beneficial to other researchers if the system could share experience between comparable systems. As such, more exploration is needed into additional transfer learning methods that can help to bridge this gap. One additional area of future work is the system to translate action weights into student assistance, as mentioned in Section II-A. Through the use of methods like natural language processing and fuzzy logic, or technologies like ChatGPT, such a system should be entirely feasible and would greatly benefit methods like the one proposed in this paper.

References

[1] J. D. Fletcher, "Adaptive instructional systems and digital tutoring," in *Adaptive Instructional Systems*, R. A. Sottilare and J. Schwarz, Eds. Cham: Springer International Publishing, 2019, pp. 615–633.

[2] N. Peirce, O. Conlan, and V. Wade, "Adaptive educational games: Providing non-invasive personalised learning experiences," 12 2008, pp. 28 – 35.

[3] J. Liang, R. Hare, T. Chang, F. Xu, Y. Tang, F.-Y. Wang, S. Peng, and M. Lei, "Student modeling and analysis in adaptive instructional systems," *IEEE Access*, vol. 10, pp. 59 359–59 372, 2022.

[4] NAE, "Nae grand challenges: Advance personalized learning," 2009. [Online]. Available: http://www.engineeringchallenges.org/challenges/learning.aspx

[5] Y. Tang, J. Liang, R. Hare, and F.-Y. Wang, "A personalized learning system for parallel intelligent education," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 2, pp. 352–361, 2020.

[6] M. Riopel, L. Nenciovici, P. Potvin, P. Chastenay, P. Charland, J. B. Sarrasin, and S. Masson, "Impact of serious games on science learning achievement compared with more conventional instruction: an overview and a meta-analysis," *Studies in Science Education*, vol. 55, no. 2, pp. 169–214, 2019. [Online]. Available: https://doi.org/10.1080/03057267.2019.1722420

[7] A. Streicher and J. Smeddinck, *Personalized and Adaptive Serious Games*, 10 2016, vol. 9970, pp. 332–377.

[8] S. Liu, Y. Chen, H. Huang, L. Xiao, and X. Hei, "Towards smart educational recommendations with reinforcement learning in classroom," in *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 2018, pp. 1079–1084.

[9] K. d. O. Andrade, G. Fernandes, G. A. Caurin, A. A. Siqueira, R. A. Romero, and R. d. L. Pereira, "Dynamic player modelling in serious games applied to rehabilitation robotics," in *2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol*, 2014, pp. 211–216.

[10] S. L. Javadi, B. Masoumi, and M. R. Meybodi, "Improving student's modeling framework in a tutorial-like system based on pursuit learning automata and reinforcement learning," in *International Conference on Education and e-Learning Innovations*, 2012, pp. 1–6.

[11] K. Georgila, M. G. Core, B. D. Nye, S. Karumbaiah, D. Auerbach, and M. Ram, "Using reinforcement learning to optimize the policies of an intelligent tutoring system for interpersonal skills training," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2019, p. 737–745.

[12] C. Watkins, "Learning from delayed rewards," 01 1989.

[13] R. Hare and Y. Tang, "Hierarchical deep reinforcement learning with experience sharing for metaverse in education," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 4, pp. 2047–2055, 2023.

[14] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," 2022.

[15] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," 2022.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[17] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," 2015.

[18] S. Schaal, "Learning from demonstration," in *Advances in Neural Information Processing Systems*, M. Mozer, M. Jordan, and T. Petsche, Eds., vol. 9. MIT Press, 1996. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1996/file/68d13cf26c4b4f4f932e3eff990093ba-Paper.pdf

[19] X. Zhang and H. Ma, "Pretraining deep actor-critic reinforcement learning algorithms with expert demonstrations," 2018.

[20] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys, "Deep q-learning from demonstrations," 2017.

[21] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," 2019.

[22] T. Wang, X. Dai, and Y. Liu, "Learning with hilbert–schmidt independence criterion: A review and new perspectives," *Knowledge-Based Systems*, vol. 234, p. 107567, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121008297

[23] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, "A survey of behavior trees in robotics and ai," 2020.