Hierarchical Deep Reinforcement Learning With Experience Sharing for Metaverse in Education

Ryan Hare, *Member, IEEE*, and Ying Tang, *Senior Member, IEEE*

***Abstract*—Metaverse has gained increasing interest in education, with much of literature focusing on its great potential to enhance both individual and social aspects of learning. However, little work has been done to address the systems and technologies behind providing meaningful Metaverse learning. This article proposes a technical framework to address this research gap, where a hierarchical multiagent reinforcement learning approach with experience sharing is developed to augment the intelligence of nonplayer characters in Metaverse learning for personalization. The utility and benefits of the proposed framework and methodologies are demonstrated in *Gridlock*, a Metaverse learning game, as well as through extensive simulations.**

***Index Terms*—ACP, experience sharing, metaverse learning, reinforcement learning (RL).**

I. Introduction

LEARNING is an inherently individual and social phenomenon where learners acquire knowledge and understanding through both independent exploration and social interactions with others [1]. While self-accumulation of knowledge is important for students to be resilient, many of them frequently need and prefer scaffolding and feedback to take on challenging learning tasks. One particular theme that is emerging as a predominant issue in learning technology is the need for effective instructional tools and methods that not only promote positive social interactions with peers and learning environments, but also offer "just-in-time" instruction tailored to individual student needs.

Metaverse, since Facebook started its meta project, is growing rapidly with increasing interest in a wide range of educational applications [2]. The Metaverse exists in a space considered parallel to the physical world, revolutionizing social interactions. The required artificial intelligence (AI) technology enables learning analytics for personalization, where a great amount of data about learners' interactions with both the virtual and physical worlds is collected. The data are then used to understand how, when, and why the learners made their learning choices. In fact, Metaverse learning resonates much with our earlier development of Parallel Intelligent Education as shown in Fig. 1 [3]. Furthermore, the Metaverse creates a great starting point from which to construct a new generation of cyber–physical–social systems to interface with learners and expand education through human–computer interactions.

Manuscript received 15 November 2022; accepted 25 November 2022. This work was supported in part by the National Science Foundation under Grant 1913809, and in part by the U.S. Department of Education Graduate Assistance in Areas of National Need (GAANN) under Grant P200A180055. This article was recommended by Associate Editor F.-Y. Wang. *(Corresponding author: Ying Tang.)*

Ryan Hare is with the Department of Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028 USA.

Ying Tang is with the Department of Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028 USA, also with the Institute of Smart Education, Qingdao Academy of Intelligent Industries, Qingdao 266000, China, also with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China (e-mail: tang@rowan.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2022.3227919.

Digital Object Identifier 10.1109/TSMC.2022.3227919

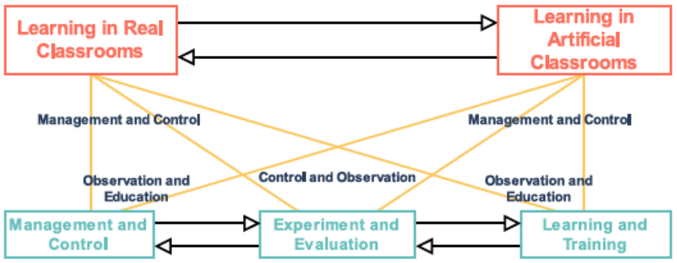
Learning in Real Classrooms
Learning in Artificial Classrooms
Management and Control
Management and Control
Observation and Education
Control and Observation
Observation and Education
Management and Control
Experiment and Evaluation
Learning and Training

Fig. 1. Parallel intelligent education system.

Ideal learning often takes place when the learner is strategically engaged in the construction of meaning and receives feedback from the learning environment on how to be more metacognitively adept [4]. Regardless of whether learning occurred in a physical classroom or an artificial world, there must be an effective way of observing and reasoning a learner's behavior. Based on those observations, appropriate scaffolding can then be provided to improve student learning. The need, importance, and potential benefits of creating a suitable mechanism to measure a student's achievement and area of difficulty in a learning environment can hardly be overstated.

Despite the heated discussions on Metaverse learning, they primarily focus on its roles and potential benefits [5], with little work on technologies for Metaverse learning. For example, a systematic approach from the basic structure to models creating physical/virtual space to parallel intelligence for decision-makings is required to provide meaningful and enjoyable learning experiences in Metaverse. As reported in two recent surveys [6], [7] any learning environment with adaptation capability must be equipped with both "eyes" to track learners' actions and a "brain" to make decisions on how to dynamically adjust its elements or provide assistance.

With the development of mixed reality [8], augmented reality [9], and virtual reality [10], the "eye" component has been made easier through virtual environments, giving developers access to any and all data on learners' actions and performance. The advancements in sensor and networking technology also makes it feasible within a learning environment to track learners' physiological changes through external stimuli, such as ECG [11], EEG [12], [13], webcams [14], [15], [16], and even phone sensors [17]. However, there are few discussions on how to create such a "brain" in Metaverse, particularly to process information as it is collected by eyes and to assess learners' current state within learning environments.

Our prior work started to explore the developmental issues of Metaverse learning, especially in the applications of serious games [3]. While the use of reinforcement learning (RL) in our work demonstrated a promising way to build the brain part, challenges are still ahead. There are many student characteristics and learning features that can be captured by eyes and used to help the brain make decisions. The more features, the higher dimensional data, the more accurate learner profiling. However, for the same reason, the size of the state space increases exponentially, and the large training time required in RL delays responses and impacts overall performance. This research aims to tackle these challenges and make the following contributions.

1) From the ACP perspective (Artificial societies, Computational experiments, and Parallel execution), this work proposes a technical framework for Metaverse learning, with emphasis on the functionalities of AI-based nonplayer characters (NPCs) for personalization.
2) An innovative hierarchical RL (HRL) method with experience sharing is proposed to address the issues of state explosion and convergence time.

The remainder of this article is organized as follows. Section II presents the ACP-based parallel control framework for Metaverse learning. Section III focuses on the proposed HRL with experience sharing. A case study is given in Section IV, followed by our conclusions and future research directions in Section V.

II. ACP-Based Parallel Control for Metaverse Learning

The ACP approach has been widely used in several application domains since its inception by Wang [18], such as discrete-time nonlinear control [19], healthcare [20], education [3], and motion recognition [21]. Extended from our prior work, Fig. 2 presents an ACP-based parallel control structure to design the brain in Metaverse learning for personalization. Learners interact with peers, including NPCs, in an authentic world created in the Metaverse. In other words, this structure creates a cyber–physical–social system that uses human-computer interaction to augment learning with both real peers and computer-controlled NPCs. There are three types of AI-based NPCs that make the learning not only more engaging but also individualized and effective.

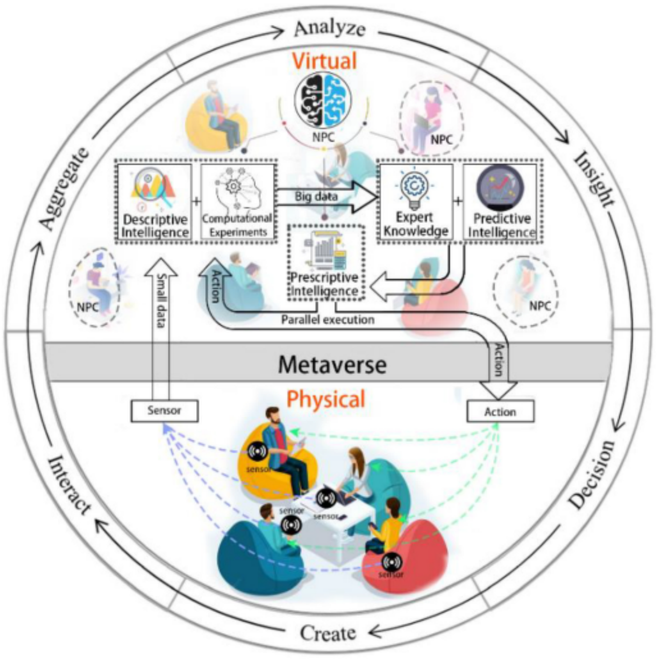

Fig. 2. ACP-based parallel control for metaverse learning.

1) Skilled learner NPC.
2) Apprentice learner NPC.
3) Tutor NPC.

These three NPCs all serve to interact with learners and augment their learning experience in different ways, as discussed below.

A. Skilled Learner NPC

From the perspective of social constructivism, interactions among peers produce both intellectual synergy of many minds to bear on a problem, and the social stimulation of mutual engagement in a common endeavor. With companions that experience and learn in the same contexts for the same education purposes, students tend to learn a great deal through mutual exploration, meaning making, and feedback.

Skilled learner NPCs have more skills and experience than the learner. Their existence in fact creates healthy competition, a good means to improve effect-based learning and attention.

B. Apprentice Learner NPC

On the other hand, another type of companion is the one who in fact needs more help from the learner. According to the Protégé effects [22], deeper learning often occurs when a learner approaches a topic with the intention to teach it later. This process also helps the learner to identify his/her own knowledge gaps. Thus, apprentice learner NPCs are those who the learner can assist or teach to strengthen their own knowledge.

C. Tutor NPC

The tutor NPC mirrors a physical instructor with three types of intelligence resulting from big data, AI, and

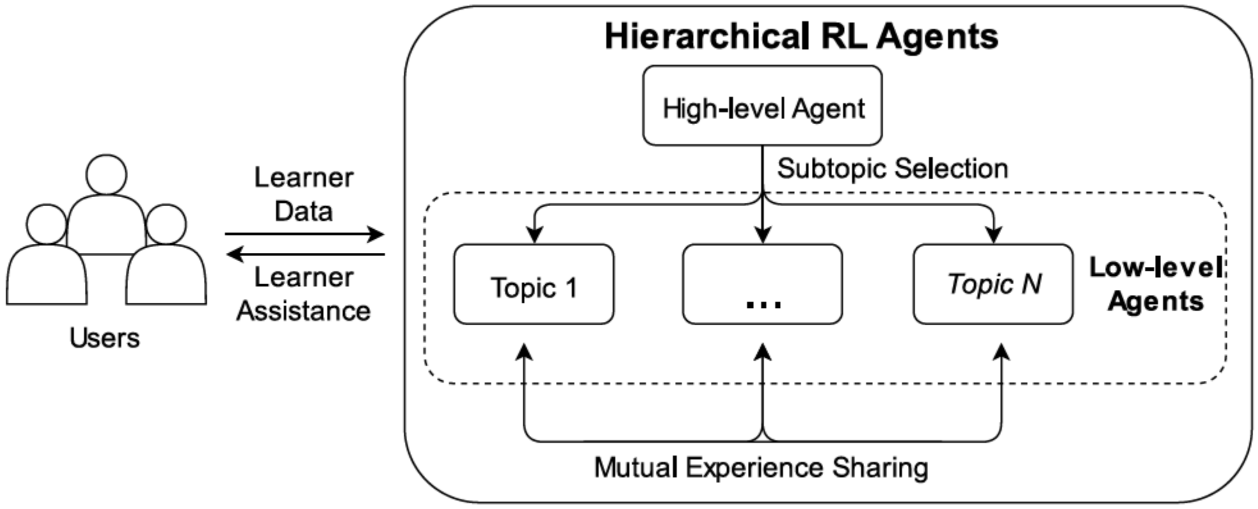

Fig. 3. HRL with mutual experience sharing.

practical experiences of many excellent instructors in reality.

1) Descriptive Intelligence: The NPC interacts with the learner and constantly monitors learning outcomes created by the introduction of various learning materials. By soliciting feedback, the NPC gains information about the effectiveness of learning materials and about a learner's difficulties. The collected information will be the basis to create what is called a student/player model [6], [7]. By analyzing the data to determine the reason behind previous learning mistakes or success in the past, this type of intelligence helps gain insights into how learners will behave in the future.

2) Predictive Intelligence: The most frequent challenge in big data is that data collected in real time is not big enough. Our prior work faced the same dilemma. Computational experiments, as the natural extension and improvement of simulations, offer a solution. At one end, the experiments are controlled trials on the virtual Metaverse learning system that transforms a set of inputs $X = \{x_i | x_i \in X\}$ for individual learners into a set of outputs $Y = \{y_i | y_i \in Y\}$, representing their observable responses. During the operation of these experiments, synthetic data is continuously generated through resampling or Monte Carlo methods [23]. In the meantime, the tutor NPC gains more confidence with the increasing data size to predict what is likely to happen to a learner.

3) Prescriptive Intelligence: Another important function of computational experiments is to understand how various factors influence the system output. From an understanding of the present factors, the most important factors can then be determined. In Metaverse learning, the tutor NPC needs to analyze data systematically and rigorously from both the physical world and computational experiments. Consequently, it makes the decision on what assistance to provide to a given learner to maximize the positive impact on that learner's education. The decisions eventually guide the learner in actions. At the same time, the actions are fed back to the virtual world forming the parallel execution.

The remainder of this article focuses on the tutor NPC, and particularly on methods that enable predictive and prescriptive analysis for personalized Metaverse learning.

III. Hierarchical Multiagent Reinforcement Learning With Experience Sharing

The design of intelligent NPCs is a key component for meaningful Metaverse learning. The more interactions NPCs have with learners, the better understanding they gain on learner behaviors, and the more appropriate scaffolding they can provide. That said, NPCs very much resemble agents in RL, who explore an unknown environment through trial and error. While our prior work put forth to use RL for building the intelligence of NPCs, the convergence and efficiency issues were not sufficiently addressed, limiting its practical applications.

Learning is a continuous process that is gradual and cumulative. Thus, Metaverse learning can be viewed as a dynamic process with a sequence of tasks that engage learners in a continuous mode of interactions. The completion of a task might lead to other tasks (i.e., sequential relation), and consequently move learners closer to an ultimate learning goal. With these remarks, this article builds upon the previous work and proposes a HRL approach. Instead of having one agent to explore the entire environment, multiple agents are built to share the responsibilities. Considering that humans are more productive to prioritize difficult tasks for multitask learning [24], our method designates a high-level agent to prioritize all tasks, and individual low-level agents to take care of the tasks.

While the content knowledge and the level of cognitive demand vary from task to task, all of them serve the same final goal. With that in mind, there are experiences that can and should be shared among the low-level agents. So, our proposed HRL is further augmented with an experience sharing mechanism. The detailed structure of the proposed method is given in Fig. 3.

Without the loss of generality, deep RL is first described in Section III-A. The proposed HRL structure is given in

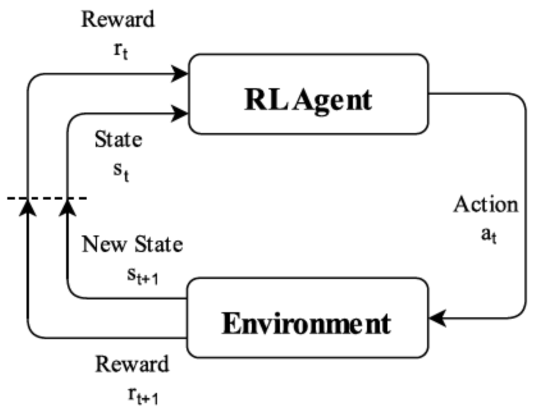

Fig. 4. Standard cycle of learning for an RL agent.

Section III-B, followed by the discussion of the experience sharing strategy in Section III-C.

A. Reinforcement Learning

RL provides a natural scheme for sequential optimal decision making in uncertain and ever-changing environments [25]. Because of the ability of RL agents to learn complex behavior and adapt to new situations, it has numerous applications in self-driving vehicles [26], robotics [27], networking [28], parameter optimization [29], disassembly planning [30], and other control problems [31]. As shown in Fig. 4, at any given time step, t, a trained RL agent observes the environment's current numerical state, s_t, and chooses an action, a_t, from a pool of possible actions. Based on the resulting new state, s_{t+1}, the agent is provided a numerical reward, r_{t+1}, determined from a reward function. The process repeats until the reward is maximized. This evolution is generally modeled as a Markov decision process (MDP) as defined below.

Definition 1: Finite MDP is defined as six-tuple: MDP $= \langle S,\ A,\ R,\ \varphi,\ \gamma,\ \pi \rangle$.

1) $S = \{s_1, s_2, \dots, s_y\}$ is a finite or continuous set of state values.
2) $A = \{a_1,\ a_2, \dots, a_z\}$ is a finite set of possible actions.
3) $R : S \times A \rightarrow \mathbb{R},\ \forall\ s \in S, a \in A$, there is a numerical reward given to the agent when observing state s and selecting action a.
4) $\phi(s_t, a_t, s_{t+1})$ defines the transition probabilities that create a mapping from state to state based on a chosen action.
5) $\gamma \in [0, 1]$ is a discount factor that is used to add or subtract weight from future expected rewards when computing the value of taking a given action.
6) $\pi(s, a) \in [0, 1]$ is the policy that defines the agent's behavior containing a probability to take each action $a \in A$ given that the agent observes state $s \in S$.

The goal of the cycle in MDP is to find a policy that maximizes the state–action-value function $Q(s, a)$. Q, then, is a function of both a state and an action, and can be estimated by the expected value of discounted future rewards r_{t+k+1} with discount factor γ. $\mathbb{E}^\pi$ represents the expected value function, which is computed as the average estimated reward assuming that the agent follows a stochastic policy, π. This full equation is shown as follows:

$$Q^\pi(s, a) = \mathbb{E}^\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t\ = a, \pi\right]. \quad (1)$$

There are many methods to estimate or track the Q-function, such as robust mean-field actor–critic [32], asynchronous actor–critic [33], advantage actor–critic [34], and deep deterministic policy gradient [35]. Double deep Q-learning is one of them that uses neural networks to predict $Q(s, a)$ [36]. Whenever the agent observes a new transition, the neural network is retrained, adjusting the internal network weights θ to minimize the difference between the network's estimated reward and the actual observed reward plus discounted future reward, as estimated by the network. The difference is called the cost function as defined in

$$\text{Cost} = Q(s_t, a_t; \theta_t) - \left[r_{t+1}\ + \gamma \max_a Q(s_{t+1}, a; \vartheta_t)\right]. \quad (2)$$

As shown in (2), a second neural network with weights ϑ is used for future reward predictions. This network is an exact copy of the main network with different weights. Using a copy of the main network with weights ϑ for future predictions helps to stabilize the main network during training. Every ρ time steps, the copy network weights ϑ are updated to reflect the main network weights θ.

Experience replay is often used to further improve the sample efficiency and stability of training [37]. By storing a set of past transition tuples, $\Psi = \{\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle\}$, the agent can randomly sample from the pool whenever new training is needed. Using experience replay improves data efficiency of the network since past experience is reused. It also breaks any correlations between data, as would occur when observing a concurrent sequence of transitions. Finally, experience replay helps prevent the agents from "forgetting" past information by periodically "reminding" the agent what it has seen in the past.

B. Hierarchical RL

HRL [38], [39] is a multiagent extension to RL systems that allows us to divide tasks into two levels handled by high-level and low-level agents, respectively. In our situation, a single high-level agent is used to prioritize all tasks, suggesting which learning tasks a learner completes in what order. As the learner explores the world, one of N low-level agents then provides task-specific support. The use of HRL has its merit in not only strategizing student learning, but also reducing computational intensity and training time via limiting the complexity of the state space. The two-level hierarchical MDP is a decomposition of the MDP into upper and low levels, each of which resembles the original definition of MDP as defined below.

Definition 2: The hierarchical MDP is defined as: HMDP $= \{\text{MDP}^j\}$, where $j = \{h,\ l\}$

1) $S = \{S^h, S^l_n\}$
   a) S^h is the high-level state space.
   b) S^l_n are N low-level state spaces that each deal with a learning task, $n \in [1,\ N]$.

2) $A = \{A^h, A_n^l\}$ is the action space defined separately for high and low-level agents.
3) $R = \{R^h, R^l\}$ the reward function used to reward the high and low-level agents, respectively.
4) $\phi = \{\phi^h, \phi_n^l\}$ for the high- and low-level state spaces.
5) $\gamma = \{\gamma^h, \gamma^l\}$ is defined to vary agent behavior between high and low levels.
   a) When $\gamma = 1$, the agent will weigh all rewards equally.
   b) When $\gamma = 0$, the agent only considers possible reward for the next time step.
6) $\pi = \{\pi^h, \pi_n^l\}$.
   a) π^h is the high-level agent's policy, creating a probability distribution over S^h and A^h.
   b) π_n^l are the low-level agent policies for each of n agents, each creating a distribution over S_n^l and A_n^l.

In this multiagent structure, the high-level agent observes the combination of all data from the learner, selecting which task the learner needs to tackle first. The corresponding low-level agent should then be activated to assist the learner, with the low-level agents each specializing in providing assistance related to one specific task and related content knowledge.

C. Experience Sharing

As stated earlier, rather than deploying one agent to capture generalized behavior in the entire Metaverse, this work instead opts to divide the target learning space into N subtasks, with each contributing to the overall knowledge that the system seeks to deliver. As such, each subtask has an individual MDP and RL agent assigned to complete, with the high-level agent choosing which subtask should be prioritized. While the transition probability function $\phi(s_t, a_t, s_{t+1})$ varies among the low-level agents, they operate similarly to offer assistance to learners with the rest of the system parameters are set. Inspired by the recent success of transfer learning methods, such as shared experience actor–critic [40], focused prioritized experience sharing [41], confidence-based sharing [42], and Q-matrix transfer [43], this research further proposes a weighted mutual experience sharing method to take advantage of this similarity between low-level agents.

Because the environmental dynamics, $\phi(s_t, a_t, s_{t+1})$, are different between agents, other agents' experience is not necessarily ideal. To address this, different weights are introduced to control how samples gathered from other agents are considered when retraining the neural networks. An agent's own experience has a weight of 1.0, while that of other agents varies based on the estimated usefulness. Then, when retraining the predictive neural network for each agent, the agent still prioritizes its own experience while also gaining benefits from other agents' experiences.

To determine the weights, the system needs a similarity metric to estimate how beneficial an agent's experience is. While there is no direct prediction made of ϕ in double deep Q-learning, a neural network is used to predict Q, the expected reward function. By comparing these reward functions, a rough estimate of the similarity of two agents can be achieved. So, to compare two agents, the centered kernel

Algorithm 1 Low-Level Agent Training With Experience Sharing

```
Initialize:
    Low-level agents $\mathcal{A}^n$ for $n \in [1, N]$
    Agent memory $\Psi^n$ for $n \in [1, N]$
Inputs:
    Batch size $b$
1:  While true do
2:      Observe initial state $s_t^i$ for $\mathcal{A}^i$
3:      Get action $a_t^i$ from $s_t^i$ from $\mathcal{A}^i$ using $\epsilon$-greedy policy
4:      Observe new state $s_{t+1}^i$ and reward $r_{t+1}^i$
5:      For agent $\mathcal{A}^j$ such that $j \in \{[1, N] - i\}$ do
6:          Sample batch $B^j \subset \Psi^j$, $|B^j| = b$
7:          Using $B^j$ as input, get per-layer neuron activation
            matrices from $\mathcal{A}^j$ and $\mathcal{A}^i$
8:          Compute weight $w^j$ as the average similarity
            between the per-layer neuron activation matrices
            using Eq. (3).
9:      End for
10:     Get batch $B^i \subset \Psi^i$, $|B^i| = b$ with weights 1.0
11:     Get batches $B^j \subset \Psi^j \forall \{j \in [1, N] | j \neq i\}$, $|B^j| = \frac{b}{N}$ with
        weights $w^j$
12:     Get final batch $B = B^i \cup B^j \forall \{j \in [1, N] | j \neq i\}$
13:     Train agent $\mathcal{A}^i$ using batch $B$ to minimize Eq. (2)
        according to double deep Q-learning
14: End while
```

alignment (CKA) similarity metric [44], as shown in (3), is adopted to directly compare the neural network representations between two agents

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

where K and L are kernel matrices derived from a set of input data, and *HSIC* is the Hilbert–Schmidt independence criterion (HSIC) that is used to compute a statistical dependence between two matrix kernels [45]. Apparently, *CKA* is a normalized version of *HSIC* that is invariant to uniform scaling.

The pseudocode of low-level agent training with knowledge sharing is presented in Algorithm 1. In this algorithm, an agent $\mathcal{A}^i$ is trained on a given iteration, with i specifying the target low-level agent. Note that the network activation matrices used in the comparison represent the internal values at each neuron in the network when given a set of input data. These internal numerical values give the network's representation of the input data, which can then be used for the comparison. Since these matrices are generated for each layer in the network, the similarity comparison is made for each layer, and the final weight is computed as the average of all similarities. It is then expected that the similarity decreases as the agents learn more specific internal weights.

Algorithm 1 also makes use of ϵ-greedy exploration, which is a common exploration policy in RL. With ϵ-greedy exploration, an exploration factor, $\epsilon \in [0, 1]$, is set to determine the chance of the agent selecting a uniformly random action from the action space. Then, with probability $1 - \epsilon$, the agent instead selects the action with the max expected reward, as estimated by the Q-function.

Algorithm 1 covers training for the low-level agents as these agents make use of our proposed experience sharing method. Unlike the low-level agents, the high-level agent operates independently to delegate tasks to the low-level agents. The high-level agent does not make use of any special methodology to undergo training. Thus, in our proposed system, the high-level agent is strained according to the standard approach of double deep Q-learning, as discussed in the original paper [36].

IV. Case Study

This section presents a case study to better understand the proposed methodology. First, the ACP-based framework and associated approach are applied to the design of a Metaverse learning game called *Gridlock*. The pilot testing of *Gridlock* as part of a computer engineering course helps collect a set of real-world learner data. A series of computational experiments is then built upon the data to verify the efficiency and effectiveness of the proposed approach in providing predicative and prescriptive analysis of human behaviors in Metaverse learning. Section IV-A briefly describes *Gridlock*. The simulation setup is presented in Section IV-B, followed by the experimental results in Section IV-C.

A. *Gridlock*

Metaverse learning engages early computer engineering students who assume the role of an engineer within *Gridlock* to design the logic controller for a traffic light. To do so, learners are required to transform the knowledge and skills learned in class into a feasible solution.

Under the ACP-based parallel control framework, learners play the game in the physical world and the same game scenes are running concurrently in the virtual space. The game splits up the design of a traffic light into eight topics, each tasking learners with specific knowledge that is part of the solution. A high-level agent and eight low-level agents are then deployed to build the tutor NPC who solicits learners' feedback and personalizes their learning experiences. More details on the Metaverse game can be found in [3] and on the project website [46].

Within *Gridlock*, agent actions are translated into student assistance. The high-level agent selects from different locations in the virtual environment, each of which pertains to a different subtopic of the game's educational content. The learner is then guided toward the target area and, by extension, the target subtopic. The eight low-level agents, meanwhile, operate within these specific areas. At set points in the learner's play, the relevant low-level agent is queried to select a piece of assistance from an expert-designed pool of assistance actions.

All agent states are determined by the learner's performance variables, which are tracked by monitoring the learner's behavior and administering regular content tests. Specifically, learner performance variables are score on tests, time taken on tests, time taken to move through game segments, emotion estimates captured through webcam images, and additional values that track the learner's game inputs. All variables are normalized in the range [0.0, 1.0], with 1.0 indicating a positive outcome.

Agent reward is determined by fluctuations in the learner's performance variables. As all variables are normalized, any increasing variables result in positive rewards for the agents. Thus, the low-level agents are incentivized to choose actions that most increase the learner's measured performance. Likewise, the high-level agent is rewarded for increases in performance as well, incentivizing the high-level agent to select subtopics where the learner shows poor performance. Finally, all agents are given diminishing returns on their reward on a per-learner basis, ensuring that they minimize the number of actions taken on any one learner and, by extension, ensuring that learners complete the game with as little assistance as possible.

B. Experimental Setup

To simulate human-like learner responses, we use binary decision trees with learner variables. The binary decision trees are generated out to a certain depth and then fit to the data collected during pilot testing. At each node in the tree, one variable and one value are chosen as the splitting criteria. At each leaf node, the possible actions are split into five categories based on the real-world data: "very good," "good," "neutral," "bad," and "very bad." So, by traversing this decision tree based on a generated instance of learner data, the system consistently decides which actions are best in which situations in a way that reflects the real-world data. This setup further reflects real-world applications as each low-level agent has a unique decision tree, creating a unique environment that must be learned.

At each step, depending on the chosen action's category, the learner's data is then shifted either positively or negatively. The magnitude of the shifts is again informed by our observations. For example, very good actions automatically shift the learner's data to a position that indicates mastery of the knowledge for the subtask. This reflects learners who understood a topic after only one piece of assistance. Meanwhile, good actions do not necessarily shift into mastery but do shift in a positive direction, reflecting learners that required two or more pieces of assistance before grasping a topic. Neutral actions apply noise to the data, but do not shift the average in a significant direction. Bad and very bad actions then apply either slight or severe negative shifts and are meant to be avoided by the agents.

To reflect the real-world implementation, all variables were capped in the range [0.0, 1.0], with 1.0 representing perfect performance on any specific metric. All agent tasks and configurations were based on the description given in Section IV-A. Agent rewards are determined by the average percentage change in all the learner's variables, with an additional reward r_c added whenever a subtask was marked as complete. All agents use three fully connected layers, two with 64 neurons each, with an additional layer at the output with a neuron for each possible action. Fully connected layers used the *relu* activation function, while the output layer used a linear activation function. Our loss function is mean-squared-error.

TABLE I
NETWORK AND AGENT HYPERPARAMETERS

Parameter	Description	Value
α	Learning Rate	0.001
γ_h	High-level Discount Factor	0.9
γ_l	Low-level Discount Factor	0.9
k	Update target network every	5
b	Training batch size	200
ϵ	Exploration Factor	$0.8 \rightarrow 1.0$
r_c	Completion Reward	3
	Decision Tree Depth	7
N	Number of Low-level Agents	8
$\|A_l\|$	Number of Actions per Low-level Agent	8

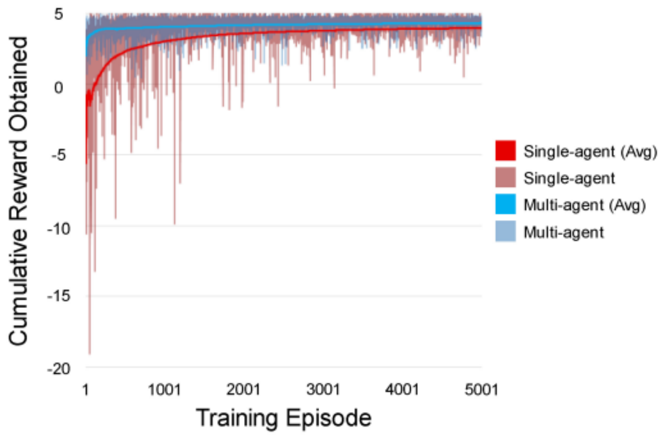

Fig. 5. Single-agent RL versus HRL average cumulative reward obtained per episode.

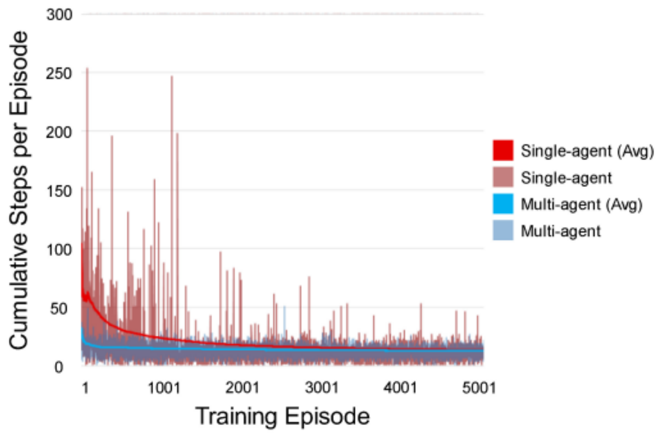

Fig. 6. Single-agent RL versus HRL total steps taken per episode until completion by all agents.

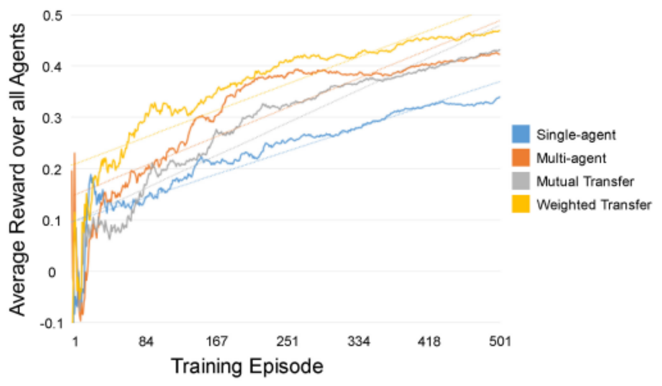

Fig. 7. Comparison of single-agent and multiagent methods, including standard multiagent, mutual experience sharing, and weighted experience sharing.

Additionally, the parameters shown in Table I are used for the double deep Q-learning.

C. Comparative Results

To verify all aspects of the system, two different computational experiments are conducted. In the first experiment, the focus is put on a single-agent approach compared to the proposed multiagent hierarchy using the high-level agent and $N = 8$ low-level agents, with 8 chosen as a balanced number of low-level agents and subtasks. All agent rewards ranged in $[-20, 5]$.

Fig. 5 compares the cumulative reward obtained by both agents per episode over 5000 training episodes. In this case, an episode consists of a learner initially interacting with the system and getting assistance in all subtasks until their data is satisfactory. It is clear to see that the hierarchical approach has visibly improved average reward in the early stages of training. By the Mann–Whitney U test, the cumulative reward shows a significant improvement in the hierarchical agent's performance with an effect size of 0.8101.

Fig. 6 compares the number of steps taken to complete each episode for the single-agent and hierarchical approaches, as well as the rolling average of the number of steps taken. For the hierarchical agent, the number of steps taken is cumulative over the high- and low-level agents. In step size, the hierarchical agent shows a similar level of improvement with an effect size of 0.6113.

The second experiment focuses solely on the N low-level agents that provide task-specific assistance. Like the prior experiment, we chose $N = 8$ as a balanced number of agents to simulate. Beyond that, simulated learner responses are computed by the same method as the first experiment. Agent rewards are capped in the range $[-2, 3]$. In this experiment, four cases are considered, where case 1 uses a single-agent approach, case 2 an N-agent approach, case 3 an N-agent approach with mutual experience sharing, and case 4 an N-agent approach with our proposed method of weighted experience sharing. Note that mutual experience sharing in case 3 is identical to our proposed approach, but all sample weights are fixed to 1.0 and no similarity metric is computed. Finally, all agents are initialized with the same network weights and same random seeds for learner simulations.

The comparison of the average reward per episode over all eight agents among four cases is then presented in Fig. 7. As shown, our proposed method outperforms all others in the average reward. When comparing it with the second-best case (i.e., case 2) using the Mann–Whitney U test, the effect size is 0.3890. Furthermore, all multiagent approaches have better performance than the single-agent method. This verifies our hypothesis that multiagent approaches can learn more specialized behavior, while a single agent would take more experience to learn generalized behavior for all tasks. For our proposed approach, the weighted experience sharing does improve the overall reward of all agents, even in the early stages of training. Without weighing the knowledge from different agents, the system performs even worse than a standard multiagent approach (effect size of 0.1765), which supports our hypothesis that shared experience should be weighted as it does not fully represent the dynamics of the target agent.

V. Conclusion

With the push toward Metaverse as the future learning platform, more research is needed into new technologies, systems, and methods to support Metaverse learning. To address this growing need, this article developed a technical framework for Metaverse learning from the ACP perspective. A special emphasis is put on functionalities of AI-based NPC design for personalized educational experiences in Metaverse. To that end, an innovative HRL approach with experience sharing is proposed, where predictive and prescriptive analysis of learner behavior in Metaverse is effectively conducted. Finally, the framework was successfully applied to the design of *Gridlock*, a learning serious game. The case study empirically demonstrated that both the hierarchical multiagent approach and experience sharing contribute to improving agent learning efficiency and reducing data requirements.

While the ACP-based parallel control framework is implemented in *Gridlock*, the development of computational experiments is still in its infancy. Exploring the essence of deduction and induction to strengthen computational experiments is worthy of future research efforts [47]. Applying our proposed framework and approach to other Metaverse learning applications is always needed to further test their validity and practicality. It is our hope that our work builds a technological foundation that inspires more research on Metaverse learning.

References

[1] F.-Y. Wang, Y. Tang, X. Liu, and Y. Yuan, "Social education: Opportunities and challenges in cyber-physical-social space," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 2, pp. 191–196, Apr. 2019, doi: 10.1109/TCSS.2019.2905941.

[2] A. Tlili et al., "Is metaverse in education a blessing or a curse: A combined content and bibliometric analysis," *Smart Learn. Environ.*, vol. 9, no. 1, pp. 1–31, 2022, doi: 10.1186/s40561-022-00205-x.

[3] J. Liang, Y. Tang, R. Hare, B. Wu, and F.-Y. Wang, "A learning-embedded attributed petri net to optimize student learning in a serious game," *IEEE Trans. Comput. Social Syst.*, early access, Dec. 23, 2022, doi: 10.1109/TCSS.2021.3132355.

[4] C. Franzwa, Y. Tang, A. Johnson, and T. Bielefeldt, "Balancing fun and learning in a serious game design," *Int. J. Game-Based Learn.*, vol. 4, no. 4, pp. 37–57, 2014, doi: 10.4018/ijgbl.2014100103.

[5] G.-J. Hwang and S.-Y. Chien, "Definition, roles, and potential research issues of the metaverse in education: An artificial intelligence perspective," *Comput. Educ. Artif. Intell.*, vol. 3, May 2022, Art. no. 100082, doi: 10.1016/j.caeai.2022.100082.

[6] J. Liang et al., "Student modeling and analysis in adaptive instructional systems," *IEEE Access*, vol. 10, pp. 59359–59372, 2022, doi: 10.1109/ACCESS.2022.3178744.

[7] R. Hare and Y. Tang, "Player modeling and adaptation methods within adaptive serious games," *IEEE Trans. Comput. Social Syst.*, early access, Sep. 19, 2022, doi: 10.1109/TCSS.2022.3203926.

[8] A. D. Cheok et al., "Metazoa Ludens: Mixed-reality interaction and play for small pets and humans," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 41, no. 5, pp. 876–891, Sep. 2011, doi: 10.1109/TSMCA.2011.2108998.

[9] J. Tedjokusumo, S. Z. Y. Zhou, and S. Winkler, "Immersive multi-player games with tangible and physical interaction," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 40, no. 1, pp. 147–157, Jan. 2010, doi: 10.1109/TSMCA.2009.2028432.

[10] H. Fei et al., "Cyberphysical system with virtual reality for intelligent motion recognition and training," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 2, pp. 347–363, Feb. 2017, doi: 10.1109/TSMC.2016.2560127.

[11] B. Pourbabaee, M. J. Roshtkhari, and K. Khorasani, "Deep convolutional neural networks and learning ECG features for screening paroxysmal atrial fibrillation patients," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 12, pp. 2095–2104, Dec. 2018, doi: 10.1109/TSMC.2017.2705582.

[12] S. Issa, Q. Peng, and X. You, "Emotion classification using EEG brain signals and the broad learning system," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 12, pp. 7382–7391, Dec. 2021, doi: 10.1109/TSMC.2020.2969686.

[13] P. Chen, Z. Gao, M. Yin, J. Wu, K. Ma, and C. Grebogi, "Multiattention adaptation network for motor imagery recognition," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 8, pp. 5127–5139, Aug. 2022, doi: 10.1109/TSMC.2021.3114145.

[14] Q.-V. Tran, S.-F. Su, W. Sun, and M.-Q. Tran, "Adaptive pulsatile plane for robust noncontact heart rate monitoring," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 9, pp. 5587–5599, Sep. 2021, doi: 10.1109/TSMC.2019.2957159.

[15] A. Halder et al., "General and interval type-2 fuzzy face-space approach to emotion recognition," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 3, pp. 587–605, May 2013, doi: 10.1109/TSMCA.2012.2207107.

[16] W. Zhang, J. Wang, and F. Lan, "Dynamic hand gesture recognition based on short-term sampling neural networks," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 1, pp. 110–120, Jan. 2021, doi: 10.1109/JAS.2020.1003465.

[17] X. Heng, Z. Wang, and J. Wang, "Human activity recognition based on transformed accelerometer data from a mobile phone," *Int. J. Commun. Syst.*, vol. 29, no. 13, pp. 1981–1991, 2014, doi: 10.1002/dac.2888.

[18] F.-Y. Wang, "Toward a paradigm shift in social computing: The ACP approach," *IEEE Intell. Syst.*, vol. 22, no. 5, pp. 65–67, Sep./Oct. 2007, doi: 10.1109/MIS.2007.4338496.

[19] Q. Wei, L. Wang, J. Lu, and F.-Y. Wang, "Discrete-time self-learning parallel control," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 1, pp. 192–204, Jan. 2022, doi: 10.1109/TSMC.2020.2995646.

[20] W. Duan et al., "An ACP approach to public health emergency management: Using a campus outbreak of H1N1 influenza as a case study," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1028–1041, Sep. 2013, doi: 10.1109/TSMC.2013.2256855.

[21] L. Jin, J. Li, Z. Sun, J. Lu, and F.-Y. Wang, "Neural dynamics for computing perturbed nonlinear equations applied to ACP-based lower limb motion intention recognition," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 8, pp. 5105–5113, Aug. 2022, doi: 10.1109/TSMC.2021.3114213.

[22] C. C. Chase, D. B. Chin, M. A. Oppezzo, and D. L. Schwartz, "Teachable agents and the Protégé effect: Increasing the effort towards learning," *J. Sci. Educ. Technol.*, vol. 18, no. 4, pp. 334–352, 2009, doi: 10.1007/s10956-009-9180-4.

[23] H. Yunhui, M. Sallak, and W. Schon, "Estimation of imprecise reliability of systems using random sets and Monte Carlo resampling procedures," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 11, pp. 2844–2855, Nov. 2017, doi: 10.1109/TSMC.2016.2523928.

[24] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei, "Dynamic task prioritization for multitask learning," in *Computer Vision* (Lecture Notes in Computer Science). Cham, Switzerland: Springer Int., 2018, pp. 282–299, doi: 10.1007/978-3-030-01270-0_17.

[25] C. Liu, X. Xu, and D. Hu, "Multiobjective reinforcement learning: A comprehensive overview," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 3, pp. 385–398, Mar. 2015, doi: 10.1109/TSMC.2014.2358639.

[26] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, and Z. Sun, "a reinforcement learning approach to autonomous decision making of intelligent vehicles on highways," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 10, pp. 3884–3897, Oct. 2020, doi: 10.1109/TSMC.2018.2870983.

[27] Y. Wang, J. Sun, H. He, and C. Sun, "Deterministic policy gradient with integral compensator for robust quadrotor control," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 10, pp. 3713–3725, Oct. 2020, doi: 10.1109/TSMC.2018.2884725.

[28] X. You, X. Li, Y. Xu, H. Feng, J. Zhao, and H. Yan, "Toward packet routing with fully distributed multiagent deep reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 2, pp. 855–868, Feb. 2022, doi: 10.1109/TSMC.2020.3012832.

[29] T.-Y. Mu, A. Al-Fuqaha, and K. Salah, "Automating the configuration of MapReduce: A Reinforcement learning scheme," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 11, pp. 4183–4196, Nov. 2020, doi: 10.1109/TSMC.2019.2951789.

[30] Z. Bi, X. Guo, J. Wang, S. Qin, L. Qi, and J. Zhao, "A Q-learning-based selective disassembly sequence planning method," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2022, pp. 3216–3221, doi: 10.1109/SMC53654.2022.9945073.

[31] D. Liu, S. Xue, B. Zhao, B. Luo, and Q. Wei, "Adaptive dynamic programming for control: A survey and recent advances," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 1, pp. 142–160, Jan. 2021, doi: 10.1109/TSMC.2020.3042876.

[32] Z. Zhou and G. Liu, "RoMFAC: A robust mean-field actor-critic reinforcement learning against adversarial perturbations on states," 2022, *arXiv:2205.07229*.

[33] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," 2016, *arXiv:1602.01783*.

[34] W. Cai, X. Guo, J. Wang, S. Qin, J. Zhao, and Y. Tan, "An improved advantage actor-critic algorithm for disassembly line balancing problems considering tools deterioration," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2022, pp. 3336–3341, doi: 10.1109/SMC53654.2022.9945173.

[35] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[36] H. V. Hasselt, G. Arthur, and S. David, "Deep reinforcement learning with double Q-learning," 2016, *arXiv:1509.06461*.

[37] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[38] Z. Yang, K. Merrick, L. Jin, and H. A. Abbass, "Hierarchical deep reinforcement learning for continuous action control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5174–5184, Nov. 2018, doi: 10.1109/TNNLS.2018.2805379.

[39] L. Huo, Z. Wang, M. Xu, and Y. Song, "A task-agnostic regularizer for diverse subpolicy discovery in hierarchical reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Sep. 30, 2022, doi: 10.1109/TSMC.2022.3209070.

[40] F. Christianos, L. Schäfer, and S. V. Albrecht, "Shared experience actor-critic for multi-agent reinforcement learning," 2021, *arXiv:2006.07169*.

[41] S. L. Oliveira, G. de Oliveira Ramos, and R. C. Ghedini, "Experience sharing between cooperative reinforcement learning agents," 2019, *arXiv:1911.02191*.

[42] T.-L. Vuong et al., "Sharing experience in multitask reinforcement learning," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 1–7.

[43] J. Sharma, P.-A. Andersen, O.-C. Granmo, and M. Goodwin, "Deep Q-learning with Q-matrix transfer learning for novel fire evacuation environment," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 12, pp. 7363–7381, Dec. 2021, doi: 10.1109/TSMC.2020.2967936.

[44] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," 2019, *arXiv:1905.00414*.

[45] T. Wang, X. Dai, and Y. Liu, "Learning with Hilbert–Schmidt independence criterion: A review and new perspectives," *Knowl.-Based Syst.*, vol. 234, Dec. 2021, Art. no. 107567, doi: 10.1016/j.knosys.2021.107567.

[46] Y. Tang and R. Hare, "Gridlock." AGAII. Accessed: Nov. 13, 2022. [Online]. Available: http://agaii.org/projects/gridlock

[47] X. Xue, Y. Xiang-Ning, Z. De-Yu, X. Wang, Z. Zhang-Bin, and F.-Y. Wang, "Computational experiments: Past, present and future," 2022, *arXiv:2202.13690*.

Ryan Hare (Member, IEEE) received the B.S. degree in electrical and computer engineering from Rowan University, Glassboro, NJ, USA, in 2019, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

His current research interests include augmenting student learning with adaptive educational systems, improving student engagement with educational games, and using machine learning methods such as reinforcement learning to create more intelligent adaptive tutoring systems.

Ying Tang (Senior Member, IEEE) received the B.S. and M.S. degrees from Northeastern University, Shenyang, China, in 1996 and 1998, respectively, and the Ph.D. degree from the New Jersey Institute of Technology, Newark, NJ, USA, in 2001.

She is a Full Professor of Electrical and Computer Engineering with Rowan University, Glassboro, NJ, USA. She is also with the Institute of Smart Education, Qingdao Academy of Intelligent Industries, Qingdao, China; also with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China; and also with the School of Information Science and Technology, Maritime University, Dalian, China. Her work has resulted in one USA patent, and over 200 peer-reviewed publications, including 71 journal articles, two edited books, and six book/encyclopedia chapters. Her current research interests lie in the area of discrete event systems and visualization, including virtual reality/augmented reality, modeling and adaptive control for computer-integrated systems, intelligent serious games, green manufacturing and automation, blockchain, and Petri Nets.

Dr. Tang served as an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING from 2009 to 2014 and is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, and an Editorial Board Member of the *International Journal of Remanufacturing*. She is a Guest Editor for the Special Issue on Behavioral Modeling, Learning, and Adaptation in Cyber-Physical Social Intelligence in IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, the Special Issue of Intelligent Energy Solutions to Sustainable Production and Service Automation in IEEE TRANSACTIONS ON ENGINEERING SCIENCE AND AUTOMATION, and the Special Issue of Advances in Green Manufacturing and Optimization in Processes. She is the Founding Chair of the Technical Committee on Intelligent Solutions to Human-Aware Sustainability for IEEE Systems, Man and Cybernetic, and the Founding Chair of Technical Committee on Sustainable Production Automation for IEEE Robotic and Automation.