Teaching Reinforcement Learning Agents via Reinforcement Learning

Kun Yang, Chengshuai Shi, Cong Shen
Department of Electrical and Computer Engineering
University of Virginia
Charlottesville, VA 22904
{ky9tc,cs7ync,cong}@virginia.edu

2023 57th Annual Conference on Information Sciences and Systems (CISS) | 978-1-6654-5181-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/CISS56502.2023.10089695

***Abstract*—In many real-world reinforcement learning (RL) tasks, the agent who takes the actions often only has partial observations of the environment. On the other hand, a principal may have a complete, system-level view but cannot directly take actions to interact with the environment. Motivated by this agent-principal capability mismatch, we study a novel "teaching" problem where the principal attempts to guide the agent's behavior via implicit adjustment on her observed rewards. Rather than solving specific instances of this problem, we develop a general RL framework for the principal to teach any RL agent without knowing the optimal action a priori. The key idea is to view the agent as part of the environment, and to directly set the reward adjustment as actions such that efficient learning and teaching can be simultaneously accomplished at the principal. This framework is fully adaptive to diverse principal and agent settings (such as heterogeneous agent strategies and adjustment costs), and can adopt a variety of RL algorithms to solve the teaching problem with provable performance guarantees. Extensive experimental results on different RL tasks demonstrate that the proposed framework guarantees a stable convergence and achieves the best tradeoff between rewards and costs among various baseline solutions.**

I. Introduction

In a typical reinforcement learning (RL) systems, the learning agent directly interacts with the environment by repetitively taking an action, observing the state transition, and collecting a reward. However, there are also many real-world RL systems where the definition of learning agent is less obvious. In some cases, a principal-client structure exists in place of a single learning agent, where the clients are trying to make their own decisions based on their own observations while the principal's task is to balance all the clients to achieve a system-level goal (implicitly based on the collective and thus more comprehensive observations of many clients). There are several motivating factors behind this principal-client structure instead of considering a single learning agent. First, for some applications, such principal-client structure may be indispensable when the principal does not have the capability to directly take actions on the environment. Second, in some other applications, no single agent can either fully observe the complete environment or completely impact the change of environment state. Last but not the least, the principal-client structure may already exist as a legacy infrastructure, *before* RL is brought in as a new solution. It is desirable to keep the existing protocol while also enjoying the new benefits of RL.

A representative example that highlights the aforementioned properties is the *cellular cognitive radio system*. We can view a principal as the base station (BS) while the clients are user equipments (UEs). The goal of the BS is to learn which channel to use for broadcasting information within its coverage area based on the current network status. However, the BS is fixed at one location and cannot observe the complete environment in its coverage area. On the other hand, it is often the case that many UEs are scattered around in the coverage area. In each time slot, every UE will try to select a channel to communicate with the BS to maximize its own communication quality. However, each of these UEs may only observe the *local* environment in its neighborhood, which provides *partial* information of the entire environment. This may lead to undesirable performance from a network/system perspective, e.g., reducing the network throughput or increasing packet collision. Finally, there are already existing (and mature) cellular signalling protocols in the devices, making it desirable to design an RL solution for the principal-client structure by maintaining the existing solution at the devices.

In this work, we address RL for a principal-client structure by proposing an RL framework for the principal that can efficiently and effectively *teach* any client running RL algorithms, by only (implicitly) adjusting the observed reward of the client. This formulation has some similarity to the *reward attack* problem in RL (see Section II for related works), but with a fundamental difference that the principal does not know the optimal action a priori. Hence, in addition to designing how to teach the clients (which can adopt any RL algorithm) by only adjusting the observed rewards, we face the new challenge of learning the optimal global action simultaneously with teaching. A key idea behind our proposed solution is to view the clients as part of the principal's observations, and formulate it as a Markov Decision Process (MDP). Then, under this new MDP, we train an RL learning agent (i.e., the principal) that can simultaneously achieve the following two goals: (1) find the globally optimal action of the complete environment, and (2) determines the optimal policy to teach each RL client via implicit reward adjustment. We evaluate the proposed solution using standard deep RL (DRL) testing environments, and the results indicate that our algorithm can

The work is partially supported by the National Science Foundation under Grant ECCS-2033671, ECCS-2143559, and CNS-2002902.

perform competitively with existing solutions that know the desired optimal global action in advance.

The remainder of this paper is organized as follows. Related works are surveyed in Section II. The proposed RL framework to teach RL clients and the theoretical guarantees are described in Sections III and IV. Experiment results are reported in Section VI. Finally, Section VII concludes the paper.

II. Related Works

The line of research that is mostly closely related to this work is *reward adjustment in RL*. In particular, there has been extensive research on data-poisoning attacks in RL, where an attacker can alter the agent's rewards in order to modify her behavior towards the goal of the attacker. The study ranges from the simple setting of multi-armed bandits (MAB) [1]–[3] to general RL [4]–[6]. Nevertheless, they require that there exists a *pre-specified* target policy known to the attacker, and the existing designs aim at making the agent almost always follow the target policy while only incurring sub-linear attack cost. One notable (albeit still limited) extension is [7], which aims at minimizing the client rewards. None of these prior studies can handle the challenging principal-client structure with the constraints mentioned in Section I. It is worth mentioning that [8] briefly discussed an idea of using an RL framework to carry out attacks, which shares the same philosophy as this work. However, [8] needs to know not only the target policy but also the exact environment dynamics (the environment MDP) as well as the agent's learning algorithm (a Q-learning-type one), which is much more restrictive.

III. System Model and Problem Formulation

To focus on the key elements of our work, in this conference paper, we consider a simplified system where at any time only one principal and one agent (i.e., client) exist in the system. The principal does not change while the agent may vary over time. The *agent* interacts with an episodic MDP by following her own RL strategy. She is oblivious of the existence of a principal and is completely autonomous in terms of the learning process. The *principal*, on the other hand, can monitor the agent's behavior but cannot directly interact with the MDP. To formulate this agent-principal mismatch as discussed in Sec. I, each performed action at the agent will have two different rewards: an individual-level one for the agent and a system-level one for the principal. The agent is oblivious and only concerns about optimizing her individual-level reward. To overcome potential mismatch and optimize the system-level performance, the principal is interested in guiding the agent. However, we enforce the constraint that the only action for the principal is to adjust the agent's local reward before revealing it to the agent. The main design objective is to efficiently utilize the reward adjustment capability at the principal to maximize the system-level reward. An illustrative diagram of the considered system model is given in Fig. 1. In the following, we present the details of the system components.

Raw MDP with two rewards. The environment is modeled as an episodic MDP, which is referred to as the **raw MDP**

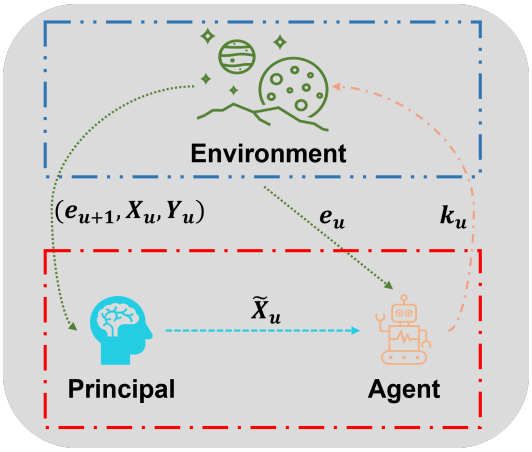


Fig. 1: System illustration. Principal and agent both interact with the MDP environment but with different capabilities and observations.

to facilitate discussion and is denoted by a tuple $\texttt{MDP}_{\text{raw}} := (F, \mathcal{E}, \mathcal{K}, \mathbb{Q}, \rho, \kappa)$. This tuple contains some standard notions in MDP: F is the number of steps in each episode, $\mathcal{E}$ is the state space with $|\mathcal{E}| = E$, $\mathcal{K}$ is the action space with $|\mathcal{K}| = K$, and $\mathbb{Q}$ is the transition matrix where $\mathbb{Q}_f(\cdot|e_f, k_f)$ is the distribution measure of the next state with action k_f in state e_f at step f. The major distinction to standard MDPs is that for action k_f taken on state e_f, two rewards are generated: $X_f := X_f(e_f, k_f) \sim \rho_f(\cdot|e_f, k_f)$ is the individual-level random reward and $Y_f := Y_f(e_f, k_f) \sim \kappa_f(\cdot|e_f, k_f)$ is the system-level random reward, respectively.

In this model, the agent can directly observe the realized states and perform actions, but can only observe the individual-level rewards X_f. From the agent perspective, she faces a "standard" MDP environment and thus can learn an RL policy based on historical states, actions and individual-level rewards (i.e., e_f, k_f and X_f). We consider that the agent interacts with the raw MDP for G episodes. Importantly, the agent's learning strategy is unknown to the principal.

As for the principal, she has the same observations as the agent, with an additional system-level reward Y_f. The principal is interested in maximizing the *cumulative system-level reward*. However, such reward Y_f is produced by the action k_f *taken by the agent* at state e_f, and the principal does not have the capability to act on the environment. As a result, the principal needs to guide the agent's behavior (i.e., "teach" the agent) to behave towards system-level objectives, while not explicitly interfering with the agent's autonomous learning operation. While different teaching methods can be studied, this work considers a practical while challenging scenario where the principal can only implicitly adjust the agent individual-level reward observations, as specified below.

Teaching via implicit reward adjustment. With action k_f taken by the agent at state e_f, the principal can adjust the environment-generated reward X_f into a perturbed one $\tilde{X}_f$, which is then revealed to the agent, who is assumed to be unaware of the reward adjustment. In other words, the adjustments are implicit and thus oblivious to the agent.

With such adjustments, the principal can shape the agent's observations, especially, the agent now observes a modified trajectory $\{e_f, k_f, \tilde{X}_f, e_{f+1}\}$ instead of the original one $\{e_f, k_f, X_f, e_{f+1}\}$. Intuitively, the agent can be guided towards a more preferable behavior w.r.t. the system-level rewards via careful reward adjustment designs.

However, adjusting rewards is often costly in real-world systems. We thus model changing reward $X_f(e_f, k_f)$ to $\tilde{X}_f(e_f, k_f)$ to incur a corresponding cost $C_f := \alpha|X_f(e_f, k_f) - \tilde{X}_f(e_f, k_f)|$, where $\alpha \geq 0$ is a known parameter that controls the cost level. We are interested in designing a reward adjustment method that can maximize the cumulative system-level reward while incurring as little adjustment costs as possible.

IV. A General RL Framework for RL Teaching

The formulated teaching problem is considerably challenging in the following perspectives. First, as mentioned, the principal needs to balance teaching effectiveness and cost efficiency. Moreover, the principal has to teach with no prior knowledge of the environment. Specifically, she needs to learn about the system dynamics (i.e., reward generation and state transition laws) from the agent's past interactions with the environment, while guiding her behaviors via reward adjustments. The principal also does not know the optimal actions to maximize system-level reward a priori. These challenges distinguish this work from previous works in guiding (or attacking) the agent either in a known environment or with a pre-specified target [1]–[6].

To solve the general problem of teaching RL agent, we take a novel approach by recognizing that this challenging problem can be modeled as *solving a new episodic MDP*, and thus we can apply RL to solving the teaching RL agent problem. Especially, the agent's behaviors can be modeled as part of the environment (of a new MDP) for the principal and her available actions are the reward adjustments, which would impact the agent (thus the principal's rewards). To distinguish from the raw MDP, the newly constructed MDP is referred to as the **teaching MDP** $\texttt{MDP}_{\texttt{teach}} := (U, \mathcal{S}, \mathcal{A}, \mathbb{P}, R)$, whose elements are defined as follows.

Episode length U. Naturally, the length of one episode for teaching is the overall time steps of G episodes in the raw MDP, i.e., $U := FG$. To facilitate the discussion, for step f at episode g in the raw MDP, another time counter is introduced as $u = (g-1)F + f \in [U]$ for the teaching MDP. In the following discussion, time $u \in [U]$ and its corresponding $(f, g) \in [F] \times [G]$ pair would be used interchangeably, e.g, $e_u = e_{f,g}, \mathbb{Q}_u = \mathbb{Q}_f$.

State $\mathcal{S}$. At step $u \in [U]$, the state s_u characterizes all the observations collected by the principal, which includes both the observations from the raw MDP and the characterization of the agent. Thus, in general, the state can be specified as a tuple $s_u := \left(\tilde{H}_u, e_u, k_u, X_u, Y_u, e_{u+1}\right) \in \mathcal{S}$, with $\tilde{H}_u := \{e_\tau, k_\tau, \tilde{X}_\tau : \tau \in [u-1]\}$ as the agent's (modified) observations up to time u. Recall that for time u, k_u is the agent's action, X_u is the environment-generated reward for agent, Y_u is the principal's reward, $\tilde{X}_u$ is the agent's observed/perceived reward (after the principal's implicit adjustment), and e_{u+1} is the next state sampled from $\mathbb{Q}_u(\cdot|e_u, k_u)$.

Action $\mathcal{A}$. At each step u, a reward X_u is generated for the agent. However, this environment-produced individual-level reward is only directly received at the principal, who modifies it to be $\tilde{X}_u$ and sends to the agent. Thus, the principal's action a_u can be defined as the adjustment value, i.e., $a_u := \tilde{X}_u - X_u \in \mathcal{A}$. Note that in many applications, the feasible reward generated by the raw MDP is often bounded, e.g., $|X_u| \in [B_1, B_2]$. In these cases, the principal's action is also constrained as it cannot exceed the natural limits of the environment, i.e., $a_u \in [B_1 - X_u, B_2 - X_u] \Rightarrow \tilde{X}_u \in [B_1, B_2]$. To simplify the discussion, instead of considering general continuous adjustments, the principal is limited to choosing reward adjustments (i.e., actions) in a finite set with size $|\mathcal{A}| = A$. This consideration is practical as the agent's rewards are often discrete (or discretized) in real world and the principal can only choose other elements in the same set $\mathcal{A}$ without agent noticing external interference.

Transition $\mathbb{P}$. With action a_u performed for state s_u, a new state is generated via the following process, which is denoted as $s_{u+1} = (\tilde{H}_{u+1}, e_{u+1}, k_{u+1}, X_{u+1}, Y_{u+1}, e_{u+2})$. First, the agent's history is supplemented with new observations, i.e., $\tilde{H}_{u+1} \leftarrow \tilde{H}_u \cup \{e_u, k_u, \tilde{X}_u\}$. Then, as the agent's actions are determined by her policy ϕ, with a modified history $\tilde{H}_{u+1}$, action k_{u+1} is to be sampled from $\phi_{u+1}(\cdot|e_{u+1}, \tilde{H}_{u+1})$, i.e., $k_{u+1} \sim \phi_{u+1}(\cdot|e_{u+1}, \tilde{H}_{u+1})$. This action further results in rewards $X_{u+1} \sim \rho_{u+1}(\cdot|e_{u+1}, k_{u+1})$ and $Y_{u+1} \sim \kappa_{u+1}(\cdot|e_{u+1}, k_{u+1})$. Finally, the environment transitions to $e_{u+2} \sim \mathbb{Q}_{u+1}(\cdot|e_{u+1}, k_{u+1})$.

The transition probability can be factorized as

$$\mathbb{P}_u(s_{u+1}|s_u, a_u) = \underbrace{\mathbb{1}\{\tilde{H}_{u+1} = \tilde{H}_u \cup \{e_u, k_u, \tilde{X}_u\}\}}_{\text{agent's history update}}$$
$$\underbrace{\phi_{u+1}(k_{u+1}|e_{u+1}, \tilde{H}_{u+1})}_{\text{agent's action selection}} \underbrace{\mathbb{Q}_{u+1}(e_{u+2}|e_{u+1}, k_{u+1})}_{\text{raw state transition}}$$
$$\underbrace{\rho_{u+1}(X_{u+1}|e_{u+1}, k_{u+1})}_{\text{agent's reward}} \underbrace{\kappa_{u+1}(Y_{u+1}|e_{u+1}, k_{u+1})}_{\text{principal's reward}},$$

where the history $\tilde{H}_{u+1}$ is updated deterministically as $\tilde{H}_u \cup \{e_u, k_u, \tilde{X}_u\}$, while the agent's action selection, the reward generation, and the raw state transition are all stochastic.

Reward R. A natural approach to unify the system-level reward and the cost for the principal is to define an *outcome* as the principal's reward minus the cost: $R_u := Y_u - C_u$, where we recall that $C_u = \alpha|X_u - \tilde{X}_u| = \alpha|a_u|$. Note that this reward is deterministic w.r.t. the state s_u and action a_u in $\texttt{MDP}_{\texttt{teach}}$.

It should be clear now that the original teaching problem can be formulated as solving the teaching MDP $\texttt{MDP}_{\texttt{teach}}$. With the outcome characterizing both the principal's reward and cost, the design goal is now to maximize the cumulative outcome.

Algorithm 1 RL2RL Framework

Require: Unknown raw MDP $\texttt{MDP}_{\text{raw}}(F, \mathcal{E}, \mathcal{K}, \mathbb{Q}, \rho, \kappa)$, sequentially arriving agents $[M]$, parameter α

Define teaching MDP as $\texttt{MDP}_{\texttt{teach}} := (U, \mathcal{S}, \mathcal{A}, \mathbb{P}, R)$
Initialize policy $\pi^{(0)}$
for $m \in [M]$ **do**
  Initialize history $H_1 = \emptyset$
  for $u \in [U]$ **do**
    Observe raw state e_u on the raw MDP $\texttt{MDP}_{\text{raw}}$
    Observe agent m performs action k_u
    Observe the realized rewards X_u and Y_u
    Construct $s_u = (\tilde{H}_u, e_u, k_u, X_u, Y_u, e_{u+1})$
    Sample action $a_u \sim \pi^{(m)}(\cdot|s_u)$
    Collect reward $R_u = Y_u - \alpha|a_u|$
    Reveal reward $\tilde{X}_u = X_u + a_u$ to agent m
    Update the observation history of the agent $\tilde{H}_{u+1} = \tilde{H}_u \cup \{e_u, k_u, \tilde{X}_u\}$
  end for
  Update policy $\pi^{(m+1)}$
end for

V. Algorithm Design for Teaching

With the general framework established, we now focus on how to solve the formulated teaching MDP problem. In particular, we consider a sequential teaching setting (also known as *population learning* in [6]). Specifically, a set of M agents enters the system sequentially while the same principal teaches all of them also sequentially. The teaching design is discussed in the following, which demonstrates the notable advantage of the flexible algorithm choice for the principal.

A. Flexibility of Principal Strategy

In the ideal case, if the principal has the exact information of the raw MDP $\texttt{MDP}_{\text{teach}}$ and the agent's policy ϕ (and thus, $\texttt{MDP}_{\text{teach}}$), she can solve the optimal policy via standard RL planning techniques (e.g., value iteration). However, the principal considered in this work does not have any prior knowledge about $\texttt{MDP}_{\text{teach}}$, and thus must learn the environment.

Luckily, with the MDP formulation, a variety of RL algorithms can be adopted for this learning process for $\texttt{MDP}_{\text{teach}}$. Especially, the proposed framework does not have any restriction on the RL algorithm adopted by the principal, which provides significant flexibility. This advantage largely benefits the practical implementation of the proposed framework as any suitable RL algorithms can be chosen under other considerations such as the target application, computation and storage requirement. The general *RL to teach RL (RL2RL)* design is summarized in Algorithm 1. We note that this algorithm is executed entirely by the principal.

B. Theoretical Guarantees for RL2RL

Despite being general, we are able to show that the theoretical analysis in RL studies can be generalized for the RL2RL algorithm. Before stating the results, the following regret definition is used as the performance metric.

$$\text{Reg}(M) := \sum\nolimits_{m\in[M]} \left[V_1^*(s_1^{(m)}) - V_1^{\pi^{(m)}}(s_1^{(m)})\right]$$

where we consider that M clients have sequentially entered (and left) the system as described earlier, $\pi^{(m)}$ is the strategy of teaching agent m and $s_1^{(m)}$ is the corresponding initial state. Moreover, $V_1^\pi(s) := \mathbb{E}\left[\sum_{u\in[U]} R_\tau(s_u, a_u)|s_1 = s\right]$ defines the expected cumulative outcomes collected by the principal with policy π while $V_1^*(s) := V_1^{\pi^*}(s)$ is the best performance from the optimal policy $\pi^* := \arg\max_\pi V_1^\pi(s)$.

Inspired by the recent theoretical advances [9]–[11], the following theorem can be proved, with specific designs and analysis deferred to the journal version.

Theorem 1 *There exists an update policy under the RL2RL framework in Algorithm 1 such that with probability at least $1-\delta$, the regret satisfies: Reg$(M) = \tilde{O}(poly(FG|\mathcal{S}|A)\sqrt{M})$, where poly$(\cdot)$ refers to a polynomial term.*

We note that the square-root dependency on M indicates that the per-agent regret, i.e., $\text{Reg}(M)/M$, is vanishing, which implies that the principal can gradually converge to the optimal teaching strategy.

One limitation in this regret guarantee is that there exists a polynomial dependency of the size of the state space $|\mathcal{S}|$, which can be large. Especially, the state s_u contains the agent's adjusted history $\tilde{H}_u$, whose possible combinations have a cardinality growing exponentially in the teaching episode length U. To make the proposed framework more practical, information extractors are introduced to handle the large state space by providing state features. The details of the adopted extractors are explained in the practical evaluations of Sec. VI.

Remark 1 *An additional theoretical advantage of the teaching MDP is that the optimal policy π^* can be rigorously defined. While seemingly insignificant, this can be a major benefit because none of the previous studies in RL reward adjustment [1], [2], [4]–[6] have discussed the optimal policy without ambiguity. In fact, previous results can only guarantee performance up to certain levels (e.g., sub-linear costs).*

VI. Experiment Evaluation

In this section, we evaluate the performance of RL2RL via numerical experiment. We first focus on a (simple) multi-arm bandit (MAB) environment, which allows us to compare our method with a theoretically-optimal baseline. Then, experiment results on grid world example are reported, where we focus on understanding when RL2RL can teach the agents effectively and whether our method can learn new targets. For both experiments, the principal is assumed to be able to modify the reward with three possible actions $\{-1, 0, 1\}$.

A. Bandit Environment

We first consider a stochastic MAB setting, where each arm's reward follows the standard Gaussian distribution. In our

experiments, the agents will face the same five arms with two different rewards for the agents and the principal, respectively. More specifically, for the agents, at the beginning of each experiment, we randomly generate a mean reward (uniformly in $[0, 1]$) for each arm, while for the principal, we randomly choose a target arm and then assign it with a mean value of one, while all other arms having zero mean values. We let the agent run the UCB-1 algorithm (with coefficient 2; see [12]) and let the principal adopt proximal policy optimization (PPO) or soft actor-critic (SAC). To compare the effectiveness of our algorithm, we compare RL2RL with three baselines. We note that all three baselines need the knowledge of the target arm, while our RL2RL does not have this information a priori.

- **Baseline 1.** This is the solution from [3], where the principal will not make any modification when the agent pulls the target arm, and will depress all non-target arms' rewards to the minimum value (i.e., 0). It is proved that this solution enjoys linear pulling time while incurring logarithmic cost.
- **Baseline 2.** This is the same as Baseline 1 except that when the agent pulls the target arm, the principal increases its reward to the maximum value (i.e., 1).
- **Baseline 3.** This is the same as Baseline 2 except that for the non-target arms, the principal does not do any reward adjustment.

Handling the growing state space. Since we formulate the history of the agents as a part of the state in the teaching MDP, one difficulty of applying RL2RL to a practical problem is that the state space will grow as the learning continues. To address this issue, we propose to use some pre-selected *features* to extra information from the history instead of directly using all the raw history as a state for the teaching MDP. For the MAB agents, the most commonly adopted algorithms UCB, Thompson Sampling, and Epslion Greedy all rely on the *sample mean* and *arm pulling frequencies*. These two features are a good representation of the history the agents have encountered. Thus in our experiments, we treat the computation of these two quantities as $\eta(\tilde{H}_u)$, a representation of the history. Using this η, we are able to "store" the history of the agents from the very beginning, which we denote as ∞-length history. We have empirically observed that using this representation of history results in better performance than some naive approaches, e.g., truncating the history to keep only the recent ones.

Results. We first set $\alpha = 1$ and evaluate the performance of RL2RL when the agents are running UCB-1. The results are plotted in Fig. 2. We can see that when the cost of the system is large, RL2RL always outperforms the three baselines. Again we emphasize that this gain is achieved despite that all baselines have access to the target arm and thus face a much easier problem. We also observe similar performance gains when the agents run other bandits algorithms, which we will detail in the journal version.

Next, we extend the experiment to different arm settings and cost levels. The goal is to test the performance of RL2RL when the environment is more favorable to the baselines. The

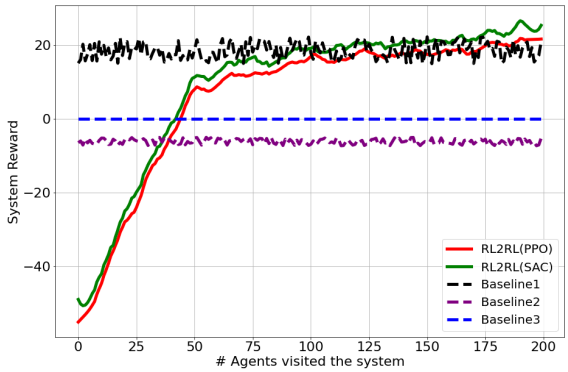


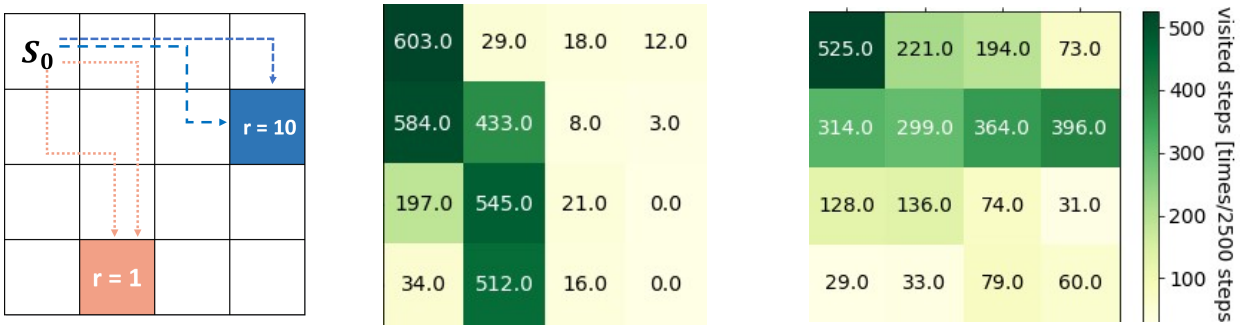
Fig. 2: Learning to teach with agents running UCB-1 and $\alpha = 1$. Each curve is averaged over the same 100 environments.

results are summarized in Table I. Due to space limitation we only report the results when principal adopts PPO. More specifically, we compare the cases where we set the target arm's mean value from $\{0.1, 0.5, 0.9\}$ which is 'not aligned', 'loosely aligned' and 'strongly aligned' with the principal target 1. We also set different cost levels by changing α, and we denote $\alpha = \{0.1, 0.5, 1\}$ as 'small cost', 'mid cost' and 'large cost' respectively. We can see from these results that the baselines, despite knowing the correct target in advance, all have some settings where they perform very poorly. On the contrary, RL2RL cannot be uniformly the best across all environments. However, it provides consistent performance (the best average) and never falls to the negative reward region. We believe this consistency comes from the "learning" capability of RL2RL where the adjustment level is judiciously and dynamically determined.

B. Grid-world Environment

We extend our experiment to an $N \times N$ grid-world environment, where each agent aims to find the best path towards the goal while the principal has an additional goal with a larger reward compare to the agent's one. The individual-level reward is such that the agent receives a constant punishment of -1 for each step it moves, while receiving a reward of 1 when it reaches its own goal. The principal has a system-level reward of 1 if the agent reaches its own goal, and receives a reward of K if the agents visits the principal's (unknown) system-level goal. We note that this specific dual-reward model captures the real-world use case where the principal finds it acceptable if an agent only reaches her individually optimal goal, but would prefer to explore the potentially better global goal. We add some environment randomness so that all agents have a 10% probability to move in a uniformly random direction at each time step. We only report the combination of agent running tabular Q-learning while principal runs PPO. Other methods are deferred to the journal version.

Handling the growing state space. Similar to the bandit case, we use a representation of the history instead of the raw history. The agents adopts Q-learning which keeps a tabular estimation of the Q-values. The Q-table reflects the



(a) Ground truth (b) Before Training (c) After Training

Fig. 3: Training result in grid-world environment. In Fig. 3a, the orange line stands for (unknown) optimal trajectories for agent's target while the blue trajectories represent the (unknown) optimal paths to reach principal's target. In Figs. 3b and 3c the number on each grid stands for how many times the agent visited the corresponding grid. The color becomes darker with more visits.

TABLE I: Performance comparison with different cost and bandit game settings

Scenario	RL2RL	Baseline1	Baseline2	Baseline3
High Cost & Not Aligned	$\mathbf{5.1 \pm 1.2}$	-21.2 ± 3.5	-11.2 ± 1.5	0.0 ± 0.0
High Cost & Loosely Aligned	21.3 ± 4.2	$\mathbf{30.1 \pm 7.5}$	-8.0 ± 1.2	0.0 ± 0.0
High Cost & Strongly Aligned	49.7 ± 7.3	$\mathbf{58.2 \pm 4.5}$	-4.1 ± 1.1	0.0 ± 0.0
Mid Cost & Not Aligned	25.1 ± 2.1	21.2 ± 4.5	$\mathbf{33.3 \pm 2.5}$	1.5 ± 7.2
Mid Cost & Loosely Aligned	$\mathbf{38.7 \pm 2.5}$	37.5 ± 6.5	38.5 ± 2.1	33.5 ± 2.4
Mid Cost & Strongly Aligned	57.7 ± 7.5	$\mathbf{68.2 \pm 3.5}$	42.5 ± 1.3	40.3 ± 1.5
Low Cost & Not Aligned	65.9 ± 9.2	23.2 ± 8.5	$\mathbf{70.6 \pm 0.5}$	37.6 ± 7.3
Low Cost & Loosely Aligned	70.5 ± 5.5	70.8 ± 5.5	$\mathbf{78.6 \pm 0.8}$	53.5 ± 5.3
Low Cost & Strongly Aligned	75.9 ± 5.3	75.4 ± 3.5	$\mathbf{82.1 \pm 0.5}$	74.6 ± 2.4
Average	$\mathbf{45.5 \pm 5.3}$	40.6 ± 6.3	36.1 ± 1.8	26.9 ± 3.1

historical information and is used as the history extractor. For the principal, she keeps the frequency of state visitation and its average rewards, and feeds the averaged reward map and reaching frequency map to the deep RL algorithm. We can see from Fig. 3 that RL2RL is effective in learning the globally optimal trajectory and teaching it to the agent, resulting in achieving a higher reward that the agent individually cannot achieve.

VII. Conclusion

We have studied a novel RL system with a principal and multiple agents. The agents can take actions that directly operate on the environment but cannot observe the system-level reward, while the principal cannot directly interact with the environment but can observe and wants to maximize the system-level reward. Only implicit agent-level reward adjustment exists as the interaction between the principal and agents. We attacked this challenge by formulating an RL agent teaching problem, and further proposed to adopt RL to solve this new problem. The resulting RL2RL framework is general in terms of the policy selection for both clients and the principal, and we provided theoretical performance guarantee. Experimental results on both bandits and RL settings demonstrate the effectiveness and efficiency of RL2RL in terms of both system-level rewards and adjustment costs.

References

[1] K.-S. Jun, L. Li, Y. Ma, and J. Zhu, "Adversarial attacks on stochastic bandits," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[2] F. Liu and N. Shroff, "Data poisoning attacks on stochastic bandits," in *International Conference on Machine Learning*, pp. 4042–4050, PMLR, 2019.

[3] H. Wang, H. Xu, and H. Wang, "When are linear stochastic bandits attackable?," in *International Conference on Machine Learning*, pp. 23254–23273, PMLR, 2022.

[4] Y. Ma, X. Zhang, W. Sun, and J. Zhu, "Policy poisoning in batch reinforcement learning and control," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[5] A. Rakhsha, G. Radanovic, R. Devidze, X. Zhu, and A. Singla, "Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning," in *International Conference on Machine Learning*, pp. 7974–7984, PMLR, 2020.

[6] A. Rakhsha, X. Zhang, X. Zhu, and A. Singla, "Reward poisoning in reinforcement learning: Attacks against unknown learners in unknown environments," *arXiv preprint arXiv:2102.08492*, 2021.

[7] Y. Sun, D. Huo, and F. Huang, "Vulnerability-aware poisoning mechanism for online rl with unknown dynamics," *arXiv preprint arXiv:2009.00774*, 2020.

[8] X. Zhang, Y. Ma, A. Singla, and X. Zhu, "Adaptive reward-poisoning attacks against reinforcement learning," in *International Conference on Machine Learning*, pp. 11225–11234, PMLR, 2020.

[9] M. G. Azar, I. Osband, and R. Munos, "Minimax regret bounds for reinforcement learning," in *International Conference on Machine Learning*, pp. 263–272, PMLR, 2017.

[10] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is Q-learning provably efficient?," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[11] Z. Zhang, Y. Zhou, and X. Ji, "Almost optimal model-free reinforcement learningvia reference-advantage decomposition," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15198–15207, 2020.

[12] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, pp. 235–256, 2002.