A Unified Batch Hierarchical Reinforcement Learning Framework for Pedagogical Policy Induction with Deep Bisimulation Metrics

Anonymous submission

No Institute Given

Abstract. Intelligent Tutoring Systems (ITSs) leverage AI to adapt to individual students, and many ITSs employ *pedagogical policies* to decide what instructional action to take next in the face of alternatives. A number of researchers applied Reinforcement Learning (RL) and Deep RL (DRL) to induce effective pedagogical policies. Much of prior work, however, has been developed *independently* for a specific ITS and *cannot directly be applied to another*. In this work, we propose a **M**ulti-**T**ask **L**earning framework that combines Deep **BI**simulation **M**etrics and DRL, named **MTL-BIM**, to induce a unified pedagogical policies for two different ITSs across different domains: logic and probability. Based on empirical classroom results, our unified RL policy performed significantly better than the expert-crafted policies and independently induced DQN policies on both ITSs.

Keywords: Deep Reinforcement Learning · Pedagogical Policy.

1 Introduction

Intelligent Tutoring Systems (ITSs) are interactive e-learning environments that support students' learning by providing individualized instruction, scaffolded practice, and on-demand help; and have shown to be highly effective [16,38]. In order to design an effective ITS, developers must determine *how* to teach the desired content. ***Pedagogical policies*** are the decision-making policies inside an ITS that decide what action to take next in the face of alternatives. While many ITSs exist for STEM domains, the pedagogical policies are often purposely built for *a single ITS in a single domain, and cannot work across ITSs.*

A number of researchers have studied Reinforcement Learning (RL) [36,2] and Deep RL (DRL) for pedagogical policy induction [18]. Prior work has also studied the impact of hierarchical RL policies, to adapt to the varying granularity of some tutoring systems [12]. The sequential decision-making nature of RL and DRL, combined with its ability to learn from a reward function, makes it a perfect fit to induce pedagogical policies for ITSs and optimize the learning process for each student individually. In particular, Batch RL methods can be used to train an RL agent on historical logs of student-tutor interactions, rather than on simulated versions of students. Batch RL methods avoid a source of error that

arises from trying to simulate a complex system such as the behavior of humans while interacting with ITSs. While most of the prior work has mainly focused on inducing effective pedagogical policy using historical interaction logs collected from *an individual* ITS, as far as we know, no prior work has investigated building robust and unified pedagogical policy induction models *across different ITSs.*

In this work, we propose a general ***M**ulti-**T**ask **L**earning* framework using Deep **BI**simulation **M**etrics and DRL **(MTL-BIM)** to induce unified pedagogical policies across multiple ITSs. *Our MTL-BIM allows us to combine different training datasets so the RL agent can train a more robust pedagogical policy through a greater range of experience.* Since ITSs are often designed to teach different subjects and therefore involve different infrastructures, instructional interventions, processes, and different learning objectives, interaction logs often differ dramatically across ITSs. ITSs can also differ significantly in the level of granularity of their interventions: problem-level, step-level, or even micro-step. Such differences make it very challenging for RL-induced policies to be effective across multiple levels of granularity. On the other hand, **Domain Adaptation (DA)** [28] and **Multi-Task Learning (MTL)** [37,26] has emerged as a promising research direction for learning across different relevant domains. DA is a sub-category of Transfer Learning in which the learning is performed using a source domain and the goal is to perform well on a different yet related target domain. DA has been shown to enhance performance in a range of fields including ITSs by sharing feature representations across different domains, wherein training data are obtained from multiple domains [35,31]. In this work, we focus on MTL, which refers to the problem of learning several tasks simultaneously and achieving better results than if those tasks were learned separately.

Our MTL-BIM combines batch DRL with deep bisimulation metrics to learn a shared *latent state representation* across tasks, which can then be used by a single shared policy to act on multiple ITSs. It allows ITS researchers to combine multiple learning systems and their respective datasets to *train a single, unified pedagogical policy, even when the granularity of the problems in the ITS is different.* To evaluate MTL-BIM, we utilize two ITSs, named ITS1 and ITS2 (real names hidden for anonymity), that teach how to solve logic proofs and probability, respectively. ITS1 has a problem-level (high level) granularity, where the agent needs to take a decision for the entire problem. However, ITS2 has problem-level (high level) and step-level (low level) granularities, which allow a hierarchical agent to make decisions in two levels. The empirical results demonstrate the ability of our single policy to improve student learning across both tutors, showing that the students who used our pedagogical policy learned more effectively and efficiently than those who used an expert-designed policy tailored to each tutor separately. Our contributions can be summarized as follows:

- To the best of our knowledge, MTL-BIM is the first work to unify the pedagogical policy induction across two different ITSs in different domains.
- MTL-BIM can unify two ITSs with different levels of decision-making granularity into a single pedagogical policy.

- As far as we know, this is the first work that combines deep bisimulation metrics with model-based DRL for Multi-Task Learning.
- We empirically demonstrate that our MTL-BIM framework can be more effective at improving student learning than expert-designed pedagogical policies, in two different real-world tasks with limited data, using a single, shared policy.

2 MTL-BIM

Problem Setup: In accordance with Domain Adaption terminology, we represent ITSs as domains; and assume two domains: $\mathcal{D}_1$ and $\mathcal{D}_2$, and the interactions in each domain can be modeled by a Markov Decision Process (MDP), described by the 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(s'|s,a)$ is the transition function, and $\mathcal{R}(s)$ is the reward function. The goal of an RL agent is to learn a policy π that selects the action a_t at time step t, which will maximize the expected sum of future rewards: $max_\pi E[\sum_{t'=t}^{T} \gamma^{t'-t} \mathcal{R}(s_{t'}, a_{t'})]$, where T indicates the final time step.
Model-Based Reinforcement Learning: In model-based RL [32], the agent explicitly learns the dynamics of the environment. This is achieved by learning the transition probabilities ($\mathcal{P}(s'|s,a)$) and reward functions $\mathcal{R}(s)$ in the MDP. Our framework is motivated by [8]. In their work, they use a Recurrent State Space Model (RSSM [9]) to learn a latent representation that can be used to plan purely in latent space. Furthermore, they use an "imagined" latent space to simulate the environment and learn the Value and Policy functions.
Bisimulation: Following the notion used in [34], we define bisimulation as a state abstraction defined for MDPs to combine states into clusters with similar properties. Intuitively, two states are bisimilar and can be grouped into the same cluster, if they share the same reward function, and if the distribution of the next bisimilar states are equivalent.

Definition 1 (Bisimulation Relation). *An equivalence relation B is a bisimulation relation if, for all states $s_i, s_j \in \mathcal{S}$ that are equivalent under B (denoted $s_i =_B s_j$) the following conditions hold: $\mathcal{R}(s_i, a) = \mathcal{R}(s_j, a), \quad \forall a \in \mathcal{A}$. $\mathcal{P}(G|s_i, a) = \mathcal{P}(G|s_j, a), \quad \forall a \in \mathcal{A}, \quad \forall G \in \mathcal{S}_B$. Here, $\mathcal{S}_B$ is the partition of $\mathcal{S}$ under the relation B (the set of all groups G of equivalent states) and $\mathcal{P}(G|s,a) = \sum_{s' \in G} \mathcal{P}(s'|s,a)$.*

The main drawback of bisimulation is that it is a very strict criterion for state aggregation. If the reward or transition functions are just slightly different, even an infinitesimal difference, then two states will not be considered bisimilar. For this reason, [23] introduced bisimulation metrics. Bisimulation metrics define a pseudo-metric space $(\mathcal{S}, d)$, where a distance function $d : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ is used to measure how behaviorally similar two states are.

Definition 2 (Bisimulation Metrics). *From Theorem 2.6 in [23] with discount factor $c \in [0,1]$, and the Wasserstein metric W_1, the bisimulation metric is defined as: $d(s_i, s_j) = \max_{a \in \mathcal{A}} (1-c) \cdot |\mathcal{R}^a_{s_i} - \mathcal{R}^a_{s_j}| + c \cdot W_1(\mathcal{P}^a_{s_i}, \mathcal{P}^a_{s_j}; d)$.*

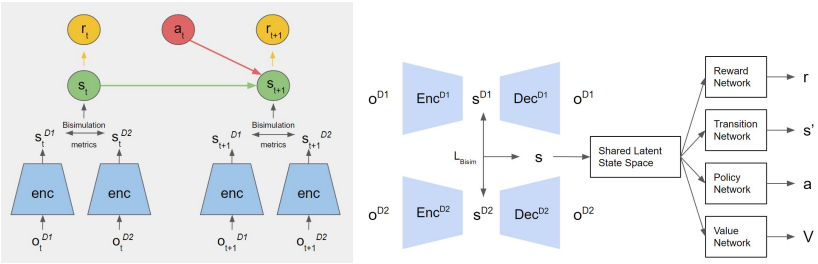

Fig. 1. MTL-BIM Framework and architecture. Left: general architecture. Right: Architecture of our framework for unifying two policies into a single system.

In summary, bisimulation is a notion of how "behaviorally similar" two states are in a transition system (such as an MDP). *Bisimulation metrics* extend and generalize this binary definition to a continuous pseudo-metric. They measure how behaviorally similar two states are using a distance metric. The smaller this value is, the more similar the two states are.

Our Method: Our goal is to unify the states from the two different ITS domains ($\mathcal{D}_1$ and $\mathcal{D}_2$) into a shared latent space, and make their distance in latent space be the same as their bisimulation metric distance, thus making behaviorally similar states be closer to one another in latent space, while making dissimilar states be far away from each other. Our work is inspired by the works of [3] and [24]. In the first work, they use a convenient form of the Wasserstein distance to be able to incorporate the bisimulation metrics into a loss function, and learn an effective representation for RL using the bisimulation metric distance. In the second work, they used bisimulation to perform RL policy transfer inside simple MDPs. In our work, we incorporate deep bisimulation metrics into a model-based DRL algorithms to build a unified framework for pedagogical policy that can improve student learning across two different ITSs.

Fig. 1 (left) shows the general architecture of MTL-BIM, and how we combine a model-based DRL algorithm with bisimulation metrics, to learn a shared latent space that will be used as input for DRL. MTL-BIM can be divided into two main parts: 1) A Variational Auto-Encoder (VAE) [33] based representation learning network, combined with bisimulation metrics, which learns a shared latent representation across the two tasks. 2) A model-based RL algorithm that gets the latent state from part 1 as input, and takes an action in the environment.

For part 1), we use two VAEs, one for each domain, and the latent space size must be the same for both VAEs, so the output layer of the encoder must have the same output size. At any given time step t in a domain $\mathcal{D}$, the encoder converts the observation space into a stochastic latent state, $s_t^{\mathcal{D}} \sim enc(o_t^{\mathcal{D}})$; while the decoder reconstructs the original observation given the latent state, $o_t^{\mathcal{D}} = dec(s_t^{\mathcal{D}})$. The VAE loss is formulated as: $\mathcal{L}_{VAE} = -\mathbb{E}_{z\sim q_\theta(z|x)}[\log p_\phi(x|z)] + \mathbb{KL}(q_\theta(z|x)||p(z))$. During training, we sample a batch of trajectories from each dataset, and we calculate the bisimulation metric between each pair of examples. We define the loss function used to train the encoder using bisimulation metrics

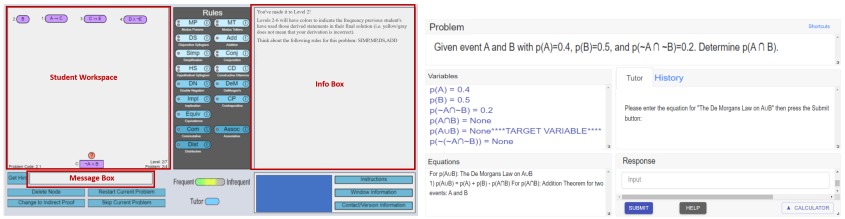

Fig. 2. Left: UI for ITS1 . Right: UI for ITS2 .

as $\mathcal{L}_{bisim} = ||s^{\mathcal{D}_1} - s^{\mathcal{D}_2}||_1 - d(s^{\mathcal{D}_1}, s^{\mathcal{D}_2})$. To train our encoders to learn the desired representation, we make the L1 distance between every two latent states be the same as their bisimulation metric distance, similar to [3]. This way, two states with a very small bisimulation metric distance will be very close to one another in the latent space, while two states with a large bisimulation metric distance will be far apart. The rewards used to calculate the bisimulation metric on definition 2 ($\mathcal{R}^a_{s_i}$) come from the dataset directly, while the transition dynamics ($\mathcal{P}^a_{s_j}$) come from the transition function learned by the model-based RL algorithm.

Fig. 1 (right) shows all the neural networks in the model. At any given time step, the VAE encodes the observation for both $\mathcal{D}_1$ and $\mathcal{D}_2$, and the bisimulation metrics are used to learn the shared latent space. Then the latent states are passed to the model-based DRL part of the algorithm, which follows a similar architecture to that of DREAMER [8], with a Recurrent State Space Model (RSSM) architecture. It trains neural networks to learn a reward function ($r_t \sim p(r_t|h_t, s_t)$), a deterministic state model that predicts the next deterministic state ($h_{t+1} = f(h_t, s_t, a_t)$), a stochastic state model ($s_t \sim p(s_t|h_t)$), a value function ($v_t = V(s_t)$), and a policy function ($a \sim \pi(a_t|s_t)$). The reward loss ($\mathcal{L}_R = \frac{1}{2}||r(s_t) - R(s_t)||^2$), transition loss ($\mathcal{L}_P = \frac{1}{2}||s_{t+1} - P(s_{t+1}|s_t, a_t)||^2$), value loss ($\mathcal{L}_V = \frac{1}{2}||v(s_t) - V(s_t)||^2$), and policy loss ($\mathcal{L}_\pi = -\mathbb{E}(\sum_{t=t}^{t+H} V(s_t))$) in the DREAMER architecture are trained jointly with the bisimulation loss and the VAE loss. Intuitively, by training these losses jointly, we are forcing the encoder network to learn a representation that can learn to maximize the rewards, as well as to ensure that the latent space of the two environments preserves relative distances, according to the the bisimulation metrics.

3 Domains & Policy Induction

Domain $\mathcal{D}_1$: Figure 2 (left) shows the User Interface (UI) for ITS1 ($\mathcal{D}_1$), which is a data-driven, graph-based ITS for multi-step propositional logic problems, that uses pedagogical policies induced via DRL [27,18]. ITS1 decides whether to present each problem as a Problem Solving (PS) or Worked Examples (WE). In PS, students *construct* logic proofs in the left *workspace* by clicking on nodes and the central rule buttons to derive new nodes until the conclusion is reached [19]. In WE, students simply *observe* how the tutor solves the problem.

$\mathcal{D}_1$ **Training Corpus:** 786 historical student-ITS trajectory interactions. Each trajectory was collected by students going through the standard pretest, training (20 problems), posttest procedure on ITS1 . Our state is comprised of the 142 features used in [27,18], including 10 student autonomy features, 29 temporal features, 35 problem-solving skills, 57 general performance, and 11 hint usage features. The goal of our DRL-induced pedagogical policy is to improve student Learning Gain while minimizing training time. Therefore, the reward is calculated as the difference between the posttest and the pretest scores, divided by the training time. This metric is named learning efficiency. The reward is normalized to a range of $[-1, +1]$ during training.

Domain $\mathcal{D}_2$: ITS2 (UI shown in Fig. 2 (right)), is a text-based ITS that teaches students how to solve probability problems using 10 major principles, such as the Complement Theorem and Bayes' Rule. ITS2 makes decisions in two different granularity levels: problem-level (high level) and step-level (low level). The former dictates whether the student sees the problem as a PS or a WE, but it can also decide to provide a Faded WE or FWE. This means that the student and the agent will collaborate when solving the problem, alternating between student-made and tutor-made steps.
$\mathcal{D}_2$'s Training Corpus: contains a total of 1148 student interaction logs. Students go through a standard *pretest-training-posttest*. Our state is represented by 142 features while these features are related to their ITS1 counterparts, but the way they are calculated is different, which makes the state spaces very different in practice. For ITS2 , the reward function is the Normalized Learning Gain (NLG), $NLG = \frac{posttest - pretest}{\sqrt{100 - pretest}}$, where *pretest* and *posttest* represent the student pretest and posttest scores. NLG measures student improvement from the pretest to the posttest, and is normalized to a range of $[-1, +1]$.

Pedagogical Policy Induction: Our goal in this work is to learn a single, unified DRL policy that can work across both ITSs. To induce the shared policy, we trained our neural network by sampling a batch of 32 trajectories from each dataset (ITS1 and ITS2), and pass the data to each encoder separately. We trained the model for 10 epochs of each dataset, with Adam as an optimizer and a learning rate of 10^{-4}.

4 Empirical Evaluation

We performed empirical classroom experiments with students from a large university in the USA. Our goal is to determine whether our unified pedagogical policy can be effective in teaching students the content of each tutor. For this, students in the Discrete Mathematics course had to use both ITS1 and ITS2 , as graded homework assignments. From this point forward, we will refer to these two studies as the *ITS1* and *ITS2* studies.
ITS1 Study Procedure and Participants: When training on ITS1 , all students go through a pretest, a training phase, and a posttest. The pretest contains four problems to evaluate students' incoming knowledge. The training phase

contains five levels, with four problems per level, where the pedagogical policy decides whether to show PS or WE to each student. Finally, the posttest consists of six problems. In Fall of 2020, we conducted an empirical study to evaluate MTL-BIM against a baseline policy that always provides PS, referred to as *PS-Only* since most of ITSs are PS only by default. 114 students finished ITS1 with a stratified random assignment: 59 were assigned to *PS-Only* and 55 were assigned to *MTL-BIM*. Furthermore, MTL-BIM is also compared against two policies that were carried out in the Spring of 2019 (S19): 23 students trained with a human-crafted *Expert* policy designed by an expert with over 15 years of experience and 30 students trained with an RL-based *DQN* [30] policy.

ITS2 Study Procedure and Participants: All students went through the same four phases: 1) textbook, 2) pretest, 3) training on the ITS, and 4) posttest. The only difference among them was how the pedagogical decisions were made. During textbook, all students read a general description of each principle, reviewed some examples, and solved some training problems. The students then took a pretest which contained a total of 14 single- and multiple-principle problems. Students were not given feedback on their answers, nor were they allowed to go back to earlier questions (this was also true for the posttest). During training, both conditions received the same 12 problems in the same order. Each domain principle was applied at least twice. Finally, all students took the 20-problem posttest: 14 of the problems were isomorphic to the pretest, and the remainders were non-isomorphic, multiple-principle problems. All of the tests were graded in a double-blind manner by a single experienced grader. For comparison purposes, all test scores were normalized to the range of $[0, 1]$. In Fall 2020, we empirically compare our MTL-BIM policy against a hierarchical control *Expert*-designed policy (default setting). A total of 98 students were randomly assigned to the two conditions: 49 *Expert* and 49 *MTL-BIM*. We also compared our MTL-BIM policy against an RL-based DQN [30] policy with 45 students, which was carried out during the Fall of 2018.

5 Results

ITS1 Results: We have four groups: *MTL-BIM* ($N = 55$) and *PS-only* ($N = 59$) from Fall 2020, and *Expert* ($N = 23$) and *DQN* ($N = 30$) from Spring 2019. A one-way ANOVA test showed a marginal difference in the pretest scores among the four: $F(3, 163) = 2.24, p = 0.09$. DQN scored higher than other groups on the pretest: Expert ($M = 0.67, SD = 0.27$), DQN ($M = 0.74, SD = 0.23$), PS-Only ($M = 0.62, SD = 0.20$), and MTL-BIM ($M = 0.63, SD = 0.19$). A one-way ANOVA test showed no significant difference in the amount of training time on ITS1 : $F(3, 163) = 1.47, p = 0.23$ with *MTL-BIM* ($M = 61.9, SD = 35.8$) minutes, PS-only ($M = 78.0, SD = 43.7$), *Expert* ($M = 65.9, SD = 87.7$), and *DQN* ($M = 93.1, SD = 109.7$). For learning performance, we leverage *posttest scores* for how well the students perform after using the ITS, and *learning efficiency*, which divides the posttest scores by the training time. Note that learning efficiency is used as the reward function for our RL agents.

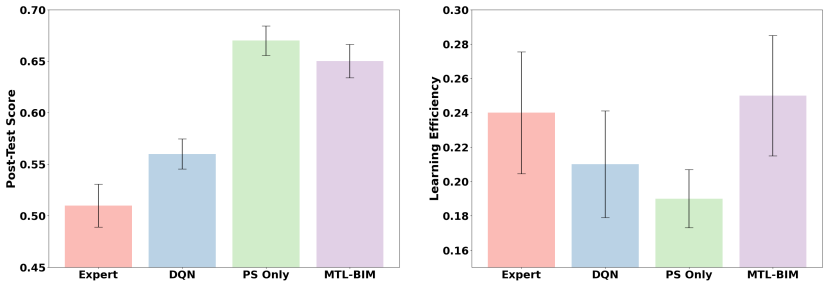

Fig. 3. ITS1 results. **posttest (Left):** MTL-BIM, PS-Only > DQN, Expert. **Learning Efficiency (Right):** MTL-BIM > PS-Only.

Fig. 3 (Left) shows the mean and standard error of the students' posttest scores. A one-way ANCOVA test using the condition as a factor and the pretest score as a covariate showed a significant difference:: $F(3, 162) = 22.32, p < 0.001$. Subsequent contrasts analyses showed *MTL-BIM* scored significantly higher than *Expert* ($t(75) = 5.235, p < 0.0001$) and *DQN* ($t(82) = 4.528, p < 0.001$). Similarly, *PS-Only* scored significantly higher than *Expert* ($t(79) = 6.852, p < 0.001$) and DQN ($t(86) = 6.389, p < 0.0001$). No significant difference was found between *MTL-BIM* and *PS-Only*, nor between *Expert* and *DQN*.

Fig. 3 (Right) shows the mean and standard error of the four conditions' learning efficiency. A one-way ANCOVA test using condition as a factor and the pretest efficiency as a covariate showed no significant difference in the learning efficiency: $F(3, 162) = 1.857, p = 0.139$. Subsequent contrasts analyses showed a significant difference in the learning efficiency scores between the *MTL-BIM* and the *PS-Only* conditions ($t(111) = 2.537, p = 0.012$) in that the former scored significantly higher than the latter.

In short, our MTL-BIM framework can induce pedagogical policies that are not only more efficient than the PS-Only policy (learning efficiency) but also more effective than the Expert and DQN policies (posttest scores).

ITS2 Results: In the ITS2 study, we compared the performance of our *MTL-BIM* policy against an *Expert* policy in Fall 2020. We also compared our policy against a *DQN* based policy, which was carried out during the Fall of 2018. A one-way ANOVA test showed no significant difference in the pretest scores among them: $F(2, 140) = 1.622, p = 0.201$. A one-way ANOVA test showed no significant difference in the amount of training time spent on the tutor among them: $F(2, 140) = 1.41, p = 0.25$ with MTL-BIM ($M = 121.3, SD = 47.1$ mins), *Expert* ($M = 118.3, SD = 29.8$), and *DQN* ($M = 108.6, SD = 34.8$).

For ITS2 , NLG is our reward function to train our RL policies. Figure 4 shows the NLG (Left), and the Isomorphic NLG (Right) across the three conditions. A one-way ANOVA test using the condition as a factor showed a significant difference in the NLG: $F(2, 140) = 3.319, p = 0.039$. Subsequent contrasts analyses showed that MTL-BIM scored significantly higher than both Expert

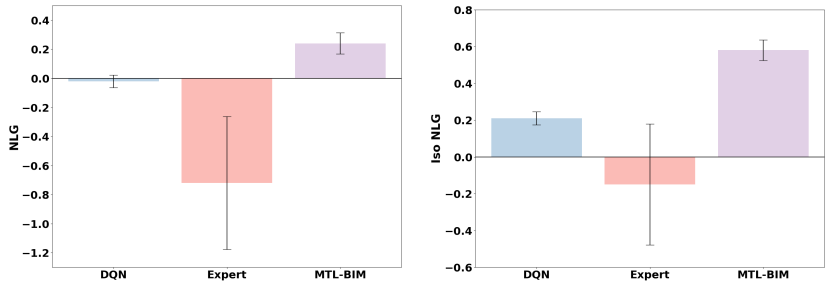

Fig. 4. ITS2 results. **NLG (Left):** MTL-BIM > DQN, Expert. **Isomorphic NLG (Right):** MTL-BIM > DQN, Expert.

Table 1. ITS2 results.

	pretest	posttest Iso	posttest	NLG Iso	NLG
DQN (F18)	0.73 (0.13)	0.85 (0.13)	0.74 (0.14)	0.21 (0.24)	-0.02 (0.29)
Expert (F20)	0.78 (0.13)	0.85 (0.11)	0.78 (0.14)	-0.15 (2.30)	-0.72 (3.20)
MTL-BIM (F20)	0.73 (0.20)	0.88 (0.15)	0.81 (0.18)	0.58 (0.39)	0.24 (0.51)

($t(96) = 2.073, p = 0.040$) and *DQN* ($t(92) = 3.002, p = 0.003$). However, no significant difference was found between the *Expert* and *DQN* conditions. Similarly, a one-way ANOVA test using the condition as a factor showed a significant difference in the isomorphic NLG: $F(2, 140) = 3.47, p = 0.034$. Subsequent contrasts analyses showed *MTL-BIM* significantly outperformed both *Expert* ($t(96) = 2.190, p = 0.031$) and *DQN* ($t(92) = 5.480, p < 0.001$) and no significant difference was found between the *Expert* and *DQN* conditions. In summary, after spending similar amounts of time on the ITS2 , our *MTL-BIM* policy is more effective than *Expert* and the *individually trained DQN* policies, as shown by the NLG and Isomorphic NLG scores.

Analysis of MTL-BIM Policy Actions: We performed a simple log analysis for our Fall 2020 study, to understand how the "unified" MTL-BIM policy behaved across the two different ITSs. Table 2 shows the number of pedagogical decisions that MTL-BIM made on each tutor. Note that two problem levels decisions, WE and PS, are made on ITS1 while three hierarchical decisions, WE, PS, and FWE are made on ITS2 . Overall, MTL-BIM made very different patterns of pedagogical decisions across the two ITSs. For ITS1 , the MTL-BIM policy decided to show PS 66% of the time, and WE 34% of the time, while for ITS2 , the MTL-BIM policy provided PS 20% of the time, WE 0.2% of the time, and FWE 78.8% of the time. That is, our proposed MTL-BIM framework can adapt from a non-hierarchical system with two actions (ITS1) to a hierarchical system with three actions (ITS2), and learn to be effective at improving student learning in both of them.

Table 2. Comparison of MTL-BIM policy actions between ITS1 and ITS2 .

	PS Count	WE Count	FWE Count
ITS1	10.27 (1.35)	5.09 (0.67)	-
ITS2	2.08 (2.5)	0.02 (0.14)	7.89 (2.56)

6 Related Work

Prior work has investigated using RL techniques to learn pedagogical policies for ITSs with different goals in mind. For instance, [4] used Q-learning to induce policies for efficient student learning. More recently, [36] applied a Partially Observable Markov Decision Process (POMDP) to train online policies for faster training. [21] employed offline policy iteration to induce a policy that improves learning gains. Lately, DRL algorithms have also been used with similar purpose, enabling the policy to scale to continuous, high dimensional state- and action-spaces, and to larger datasets. [18] trained an offline DRL algorithm to learn a pedagogical policy that improves student learning when compared to an expert-crafted policy. The attempts to unify multiple ITSs have been very limited. [1] tried to merge the pedagogical policies for two different tutors, but did not use an RL-based approach. The work by [20] managed to combine different sub-tasks in an educational game using MTL, and managed to more accurately predict posttest scores.

Improving generalization in Machine Learning algorithms is a very active area of research. Transfer Learning, Domain Adaptation, Multi-Task Learning, Meta-Learning, or Multi-Domain Learning have been some of the solutions that have been developed to try to solve the generalization problem. Domain adaptation (DA) approaches have shown great promise. These approaches aim to find a common space between domains to generate domain-invariant representations [28,22,11,14,17]. In RL, the generalization ability of agents has been widely studied through Multi-Task RL approaches [37,5,6,26,13]. [15] studied the generalization ability of RL agents and proposed a method that improves generalization by training the agent on a mixture of observations, which acts as regularization. [25] proposed a Multi-Task Reinforcement Learning approach with explicit soft modularization, to improve the optimization process of learning several tasks while sharing parameters across tasks. The work by [29] presents an approach to Multi-Task Deep RL using attention to automatically learn the relationships between tasks. Finally, prior work has addressed the Multi-Task RL problem by learning a shared representation across tasks, which is what we have done in this work. The work by [10] learns a shared latent representation of the state-action space across all the tasks. [7] also learned a shared representation across tasks that allow them to outperform the single-task versions of the algorithms in standard RL benchmarks.

7 Conclusion and Discussion

Our goal in this work is to build a robust and unified pedagogical policy *across different ITSs*. We developed MTL-BIM, a framework to learn a shared latent state representation using deep bisimulation metrics, in combination with a model-based Deep Reinforcement Learning algorithm, and used it to induce a unified pedagogical policy for two different ITSs. We showed that MTL-BIM can learn to act effectively in both ITSs, and learn a unified policy by combining both training datasets. Our results show that, in ITS1 , the students who used our MTL-BIM policy learned more than the students who used both the Expert and the DQN policies, and they also learned more efficiently than the students assigned to the PS-Only policy. In ITS2 , our results show that the MTL-BIM students improved significantly more than the students in both the Expert and the DQN policies, both in terms of general NLG and isomorphic NLG. We believe this work provides a step forward in learning shared pedagogical policies across multiple ITSs. This method allows a single policy to learn from multiple datasets at the same time, to then optimize student learning across both tutors. However, we believe more testing is required and our method should be further evaluated on other ITSs.

References

1. Abdelshiheed, M., et al.: Metacognition and motivation: The role of time-awareness in preparation for future learning. In: CogSci (2020)
2. Singla et al., A.: Reinforcement learning for education: Opportunities and challenges. arXiv preprint arXiv:2107.08828 (2021)
3. Zhang et al., A.: Learning invariant representations for reinforcement learning without reconstruction. arXiv preprint arXiv:2006.10742 (2020)
4. Iglesias et al., A.: Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems. Knowledge-Based Systems **22**(4), 266–270 (2009)
5. Rusu et al., A.A.: Policy distillation. arXiv preprint arXiv:1511.06295 (2015)
6. Rajeswaran et al., A.: Epopt: Learning robust neural network policies using model ensembles. arXiv preprint arXiv:1610.01283 (2016)
7. D'Eramo et al., C.: Sharing knowledge in multi-task deep reinforcement learning. In: ICLR (2019)
8. Hafner et al., D.: Dream to control: Learning behaviors by latent imagination. arXiv preprint arXiv:1912.01603 (2019)
9. Hafner et al., D.: Learning latent dynamics for planning from pixels. In: ICML. pp. 2555–2565 (2019)
10. Borsa et al., D.: Learning shared representations in multi-task reinforcement learning. arXiv preprint arXiv:1603.02041 (2016)
11. Tzeng et al., E.: Simultaneous deep transfer across domains and tasks. In: ICCV. pp. 4068–4076 (2015)
12. Zhou et al., G.: Hierarchical reinforcement learning for pedagogical policy induction. In: AIED 2019. vol. 11625, pp. 544–556. Springer (2019)
13. Higgins et al., I.: Darla: Improving zero-shot transfer in reinforcement learning. In: ICML. pp. 1480–1490. PMLR (2017)

14. Xing et al, J.: Domain adaptation in reinforcement learning via latent unified state representation. In: AAAI 2021. pp. 10452–10459. AAAI Press (2021)
15. Wang et al., K.: Improving generalization in reinforcement learning with mixture regularization. arXiv preprint arXiv:2010.10814 (2020)
16. Koedinger et al., K.R.: Intelligent tutoring goes to school in the big city. IJAIED **8**, 30–43 (1997)
17. Wu et al., K.: Deep reinforcement learning boosted partial domain adaptation. In: IJCAI 2021. pp. 3192–3199 (2021)
18. Ausin et al., M.S.: Exploring the impact of simple explanations and agency on batch deep reinforcement learning induced pedagogical policies. In: AIED. pp. 472–485. Springer (2020)
19. Maniktala et al., M.: Avoiding help avoidance: Using interface design changes to promote unsolicited hint usage in an intelligent tutor. IJAIED (2020)
20. Geden et al., M.: Predictive student modeling in educational games with multi-task learning. In: AAAI. vol. 34, pp. 654–661 (2020)
21. Chi et al., M.: Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. UMAP **21**(1-2), 137–180 (2011)
22. Long et al., M.: Learning transferable features with deep adaptation networks. In: ICML. pp. 97–105 (2015)
23. Ferns et al., N.: Bisimulation metrics for continuous mdps. SIAM **40**(6), 1662–1714 (2011)
24. Castro et al., P.S.: Using bisimulation for policy transfer in mdps. In: AAAI (2010)
25. Yang et al., R.: Multi-task reinforcement learning with soft modularization. arXiv preprint arXiv:2003.13661 (2020)
26. Sodhani et al., S.: Multi-task reinforcement learning with context-based representations. arXiv preprint arXiv:2102.06177 (2021)
27. Shen et al., S.: Empirically evaluating the effectiveness of pomdp vs. mdp towards the pedagogical strategies induction. In: AIED. pp. 327–331. Springer (2018)
28. Pan et al., S.J.: Domain adaptation via transfer component analysis. IEEE Transactions on Neural Networks **22**(2), 199–210 (2010)
29. Bram et al., T.: Attentive multi-task deep reinforcement learning. Preprint arXiv:1907.02874 (2019)
30. Mnih et al., V.: Playing atari with deep reinforcement learning. NIPS (2013)
31. Mao et al., Y.: Cross-lingual adversarial domain adaptation for novice programming. In: AAAI (2022)
32. Ha, D., Schmidhuber, J.: World models. arXiv preprint arXiv:1803.10122 (2018)
33. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. Preprint arXiv:1312.6114 (2013)
34. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. Information and computation **94**(1), 1–28 (1991)
35. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR. pp. 4293–4302 (2016)
36. Rafferty, A.N., Brunskill, E., et al.: Faster teaching via pomdp planning. Cognitive science **40**(6), 1290–1332 (2016)
37. Tanaka, F., Yamamura, M.: Multitask reinforcement learning on the distribution of mdps. In: IEEE International Symposium on CIRA. vol. 3, pp. 1108–1113 (2003)
38. Vanlehn, K.: The behavior of tutoring systems. IJAIED **16**(3), 227–265 (2006)