Combining Planning and Reinforcement Learning for Solving Relational Multiagent Domains

Nikhilesh Prabhakar*
The University of Texas at Dallas
Richardson, Texas, USA
nikhilesh.prabhakar@utdallas.edu

Ranveer Singh*
The University of Texas at Dallas
Richardson, Texas, USA
ranveer.singh@utdallas.edu

Harsha Kokel
IBM Research
San Jose, California, USA
harsha.kokel@ibm.com

Sriraam Natarajan
The University of Texas at Dallas
Richardson, Texas, USA
sriraam.natarajan@utdallas.edu

Prasad Tadepalli
Oregon State University
Corvallis, Oregon, USA
tadepall@eecs.oregonstate.edu

ABSTRACT

Multiagent Reinforcement Learning (MARL) poses significant challenges due to the exponential growth of state and action spaces and the non-stationary nature of multiagent environments. This results in notable sample inefficiency and hinders generalization across diverse tasks. The complexity is further pronounced in relational settings, where domain knowledge is crucial but often underutilized by existing MARL algorithms. To overcome these hurdles, we propose integrating relational planners as centralized controllers with efficient state abstractions and reinforcement learning. This approach proves to be sample-efficient and facilitates effective task transfer and generalization.

KEYWORDS

Multiagent Learning, Relational Reinforcement Learning, Statistical Relational Learning, Abstraction, Planning

1 INTRODUCTION

Building multiple agents capable of learning to reason and act under uncertainty in large and complex environments has long been a cherished goal of AI. Reinforcement learning (RL) [37] and multiagent RL [1] techniques have long been developed for learning under uncertainty and in the presence of multiple agents, respectively. Several previous research efforts have extended these methods to hierarchical domains with multiple levels of state and action abstractions [7, 36, 38].

---

[1]*Equal contribution

Statistical Relational Learning and AI (StaRAI) [11, 32], on the other hand, have dealt with learning in the presence of varying numbers of objects and relations, i.e., in relational domains. However, relational RL [8] is relatively unexplored, and while some methods exist [42], they do not scale for large tasks and are certainly not easily extensible to multiagent settings. A promising direction is exploiting the combination of hierarchical (and relational) planning to explore multiple levels of abstraction and RL to learn lower-level policies [16, 20].

Inspired by the success in these different sub-areas of AI, we propose a method that leverages the power of a relational hierarchical planner to act as a centralized controller for multiagent learning in noisy, relational domains. Our proposed approach, called *Multiagent Relational Planning and Reinforcement Learning (MaRePReL)*, uses planning for task decomposition, centralized control, and agent allocation, StaRAI for constructing task-specific representations, and deep RL for effective and efficient learning with these specialized representations.
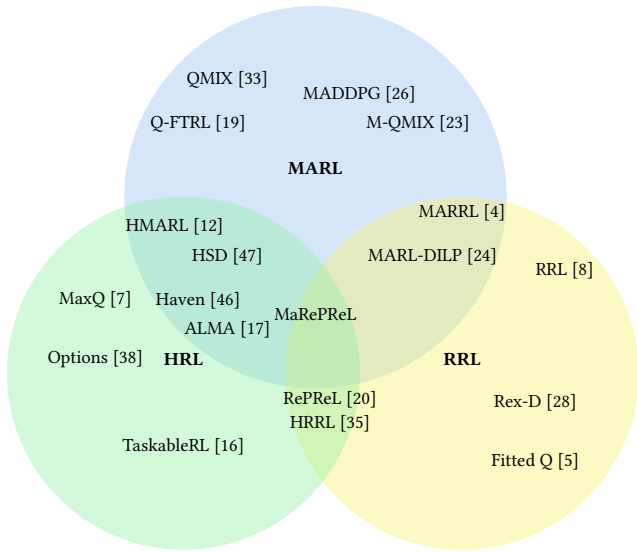
We make the following key contributions: (1) As far as we are aware, we present the first multiagent system for relational multiagent domains that can generalize across multiple objects and relations. As we show in the related work, significant literature exists in multiagent systems, relational learning, and the integration of planning and learning. Ours is the first work to combine all these directions in the context of multiagent systems. (2) To achieve this, we develop MaRePReL, an integrated planning and learning architecture capable of multiagent learning under uncertainty in relational domains. Specifically, MaRePReL's effective learning and reasoning power stems from its representation of relational information, the decomposition of higher-level plans, and the use of deep RL at the lowest level. (3) Finally, we demonstrate our approach's effectiveness and generalization abilities in a few relational multiagent domains. We compare against different deep RL based multiagent baselines, including one that explicitly uses the sub-task information, and illustrate the superiority of our approach.

The rest of the paper is organized as follows: after reviewing the related work and presenting the necessary background, we outline our multiagent framework and discuss the algorithm in greater detail. We then present the experimental evaluation on a few relational multiagent domains before concluding the paper by discussing the areas of future research.

QMIX [33]

MADDPG [26]

Q-FTRL [19]

M-QMIX [23]

MARL

HMARL [12]

MARRL [4]

HSD [47]

MARL-DILP [24]

RRL [8]

MaxQ [7]

Haven [46]

MaRePReL

ALMA [17]

Options [38]   **HRL**

RRL

RePReL [20]

Rex-D [28]

HRRL [35]

TaskableRL [16]

Fitted Q [5]

Figure 1: Our proposed framework w.r.t existing literature on relational, hierarchical, and multiagent RL

2   RELATED WORK

Research in RL in the past three decades has focused on several extensions that make them adaptable to several real-world scenarios. First, Hierarchical Reinforcement Learning (HRL) methods have been introduced to manage complex tasks by decomposing them into smaller, more manageable subtasks [36]. These allow for more efficient learning and problem-solving by utilizing multiple levels of abstraction. Second, Relational Reinforcement Learning (RRL) addresses the complexity of environments where states and actions consist of objects and the relationships between those objects [8]. RRL exploits a higher-order representation of the underlying relational structure to improve learning in such domains. Third, Multi-Agent Reinforcement Learning (MARL) has been developed to handle environments where dynamic changes arise from the presence and actions of other agents, making it particularly useful in competitive or cooperative multi-agent settings [1]. Before introducing our framework, which addresses all three challenges, namely hierarchies, relational structures, and multi-agent domains, we review the relevant literature for these three RL extensions.

Hierarchical Reinforcement Learning (HRL) algorithms have been developed to tackle the complexity of long-horizon tasks by breaking them down into smaller, more manageable subtasks. Frameworks such as the Options [38] and MAXQ [7] facilitate the learning of hierarchical policies across multiple levels of abstraction. By exploiting temporal abstraction, HRL transforms the original long-horizon task into a sequence of shorter-horizon subtasks, where each subtask represents a high-level action that spans a longer period than the lower-level actions carried out by agents deeper in the hierarchy. This hierarchical structure enhances the agent's ability to operate effectively over extended time horizons and significantly improves learning efficiency [31].

Multiagent Reinforcement Learning (MARL) extends reinforcement learning to systems with multiple agents, where they interact with the environment to maximize cumulative rewards [1]. However, MARL introduces its own unique set of challenges. The first challenge is the curse of dimensionality, where the increasing number of agents leads to an exponential increase in the sizes of state and action spaces. The second challenge is the non-stationary nature of the environment due to the actions taken by the other agents. The final issue for MARL is the sample inefficiency due to the large amount of data required to train such agents [48].

Numerous solutions have been proposed to address the challenges outlined above, falling into two main categories: those adapting the underlying architectures for the RL agents and those considering the overall tasks performed by the agents. In the former category, methods use function approximation techniques to combat the curse of dimensionality [2]. In addition, Centralized Training, and Decentralized Execution (CTDE) methods such as QMIX and MADDPG address the non-stationary nature of the environment [26, 33]. Generative modeling or mask reconstruction algorithms [19, 23] also fall in this category. In the latter category, hierarchical approaches such as HMARL [12] and HSD [47] utilize task decomposition and hierarchical structures in multiagent settings to define task abstractions and improve sample inefficiency by filtering out irrelevant parts of the state space. Additionally, the structured task hierarchies introduced in such methods can facilitate agent communication to address the non-stationary nature of different multiagent environments. While powerful in standard propositional (and continuous) settings, these methods do not address the challenge of a rich, relational structure in the environment.

Relational Reinforcement Learning (RRL) considers the task of learning in environments where states and actions involve relationships between objects and their properties, i.e., relational domains [8]. In these domains, RL agents must explicitly learn to reason about and exploit the relationships between objects [39]. Previously, several works have demonstrated the need for a rich relational representation to be explicitly used inside the learning algorithms as against simply grounding all the objects and obtaining a feature-based representation [32, 40]. A key advantage of relational representations is their ability to support abstractions and facilitate generalization and effective transfer across tasks [11, 27, 41]. However, finding optimal policies in many relational domains is intractable even for moderately large problems [39]. To mitigate this issue, algorithms that incorporate guidance and domain knowledge as constraints on the policy space have been developed [28].

Planning and RL integration have been explored to exploit the power of hierarchical planning with deep RL enabling the use of HRL in continuous domains. While Taskable RL [16] demonstrated significant performance improvement, the underlying planner was still propositional, limiting their applicability to relational problems with varying numbers of objects and relations.

An ideal RL learning algorithm should be able to not only handle the rich relational structure of the domain but also have the ability to represent and reason with the decomposition of complex tasks into smaller ones. In other words, the algorithm must be capable of representing and reasoning with both hierarchies and relational structures. One such recent framework, RePReL [20], employs a hierarchical relational planner to implement task-specific policies and uses Deep RL to work on hybrid relational domains [20, 21]. To interface the higher-level planner with the Deep RL, a *hand-crafted*

abstract reasoner is employed to lift the reasoning process and construct smaller lower-level Markov Decision Processes (MDPs) that can be solved efficiently. This approach has been demonstrated to be successful in domains with varying numbers of objects, complex task structures, and continuous state-action spaces.

While the RePReL framework successfully handled relations and hierarchies in continuous spaces, it can not handle multiagent systems. More precisely, given the three-pronged challenge of complex task structures, rich object-centric environments, and multiagent domains, several advances have been made in each of these specific directions. Also, in the recent past, methods that arise from the combinations of these methods – for instance, HRL with RRL [7, 20, 35], MARL with RRL [4, 24], HRL with MARL [12, 17, 46, 47] – have been proposed. However, no significant research encompasses all three of these challenges (see Figure 1).

It is precisely this gap that we aim to address in this work. Specifically, we extend the RePReL framework to multiagent settings, utilizing the planner as both a scheduler and a centralized controller. Unlike RePReL, where the planner is solely responsible for task decomposition, our proposed framework also distributes tasks among multiple agents. This key enhancement enables our approach to effectively solve relational multiagent domains, as we explain in the next section.

3 MULTIAGENT RELATIONAL PLANNING AND REINFORCEMENT LEARNING (MAREPREL)

We consider the problem of coordinating multiple agents to solve continuous, relational problems. We will first provide a high-level overview of our framework, MaRePReL, combining relational and hierarchical planning with deep reinforcement learning before defining the problem formally. MaRePReL employs the following:

(1) **Planner as controller:** One of our key contributions is to view the (relational) planner as a centralized controller that obtains the current state as input and creates a set of agent-specific plans. In the spirit of two-level systems, the planner not only decomposes the tasks into subtasks but also assigns the subtasks to appropriate agents. To facilitate this, the controller consists of two specific components:

  (a) **Relational Hierarchical Planner:** Recall that the goal is to decompose tasks in the presence of varying numbers of objects and relations between them. Consequently, the first sub-component is a relational, hierarchical planner, which uses a first-order representation to model the objects and relationships in the domain. As one can view hierarchies as a specific form of relations, this planner can decompose the goals into a temporally ordered series of subgoals.

  (b) **Task Distributor:** The planner output is typically the task decomposition and does not bind the tasks to the specific agents. We use a task distributor as part of the relational planner to divide the ordered plan provided into agent-specific sub-plans using agent constraints for the different tasks.

(2) **Abstraction Reasoner:** Following the previous work on single-agent learning (RePReL), we use Dynamic-First Order Conditional Influence (D-FOCI) statements [29] to capture domain knowledge that is then used to reason and construct the relevant parts of the state space that the lower-level RL agents use. This step at the outset is similar to RePReL, where the inference step is still hand-crafted, and leveraging lifted inference [40] to perform this step automatically remains a future direction. However, it must be emphasized that this step is more difficult in the multiagent setting as the optimal allocation of tasks to agents requires considering all agent states. Hence, typical single-agent-based abstraction reasoners do not suffice for this setting as they do not capture the true optimal value functions.

(3) **Multiple Deep RL agents:** Given the current subtask from the planner, the corresponding (deep) RL agent identified by the task distributor learns a generalizable, task-specific policy. Assuming that the abstraction reasoner identifies the relevant part of the state space, learning is both effective and efficient with the additional advantage of being generalizable since the learned policies can be shared among multiple agents (as shown in our experiments).

The broad overview of our proposed approach is presented in Figure 2. The planner decomposes the higher-level tasks into appropriate lower-level tasks using a relational representation of the current state and lifted operators. The distributor identifies the appropriate RL agent for the current subtask, thus making this combination an effective centralized controller. Given the subtask and the current (abstract/relational) state, the abstraction reasoner selects a smaller state representation by identifying the appropriate parts of the state space that are relevant to the current subtask. Finally, the RL agent either learns the policy or executes the ones it already learned (for instance, agent $A1$ might have learned the *pickup* subtask that can be used directly by agent $A2$ to execute this specific subtask). Note that while our experiments assume all the agents to be homogenous, this is not a necessary assumption for our formulation, where specific constraints can be used to allocate the tasks accordingly.

3.1 Problem Formulation

While one could envision using relational partially observable MDPs (RPOMDPs) to model the problem with the current task being the hidden component, issues arise when modeling it as one. First, the abstraction reasoner has to track the current hidden state actively. Second, the use of decentralized RPOMDPs [44] requires creating multiple RMDPs one for each task-agent formalism which would, in general, be larger than the smaller RMDPs created by our formalism using abstractions. Finally, for larger agent-task combinations, the reasoning over belief states requires approximate probabilistic inference over relational states and lifted probabilistic inference, which is outside the current work's scope. Therefore, we consider modeling the problem using Markov games [25]. We build upon the framework of relational Markov games [10], extending it to handle goal-oriented problems. We formalize this extension as a goal-directed relational Markov game (GRMG), defined as follows:

Definition 1: *A goal-directed relational Markov Game (GRMG) is represented as* $M = \langle N, S, A^i_{i \in N}, P, R^i_{i \in N}, \gamma, G \rangle$ *where N is the number of agents, S is the set of (relational) states, A^i is the set of actions for*
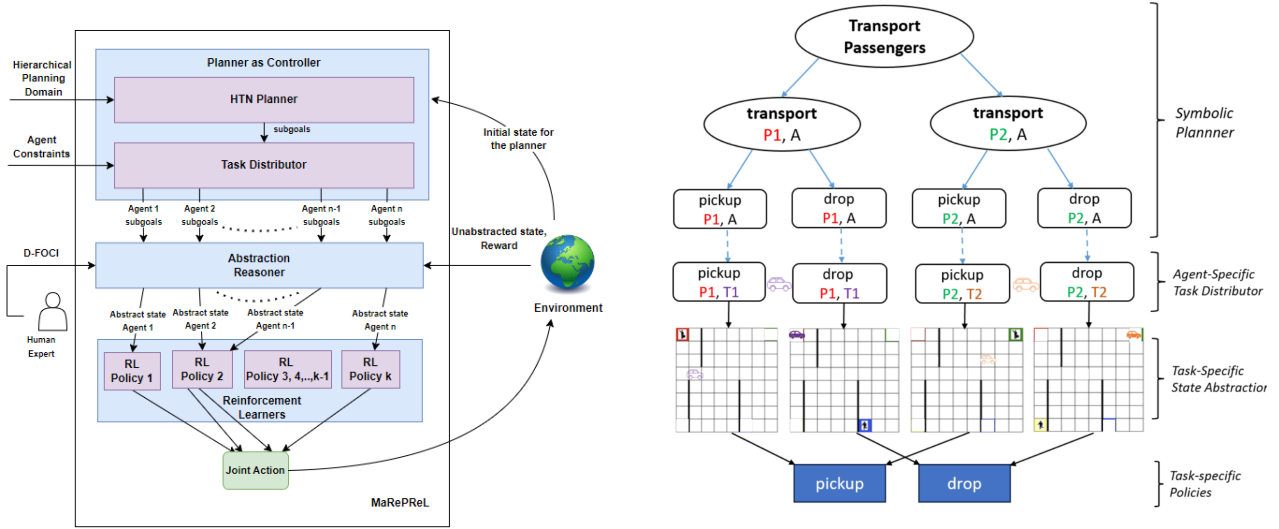
Figure 2: MaRePReL architecture and application in the taxi world for the task of transporting two passengers

the ith agent, and $A := A^1 \times A^2 \times ..... \times A^N$ is the joint action space, $P = Pr(s'|s, a)$ is the transition probability function for transitioning from s to s' where $s, s' \in S$ and $a \in A$, $R^i = S \times A \times S \rightarrow \mathbb{R}$ (set of real numbers) is the reward function for the ith agent, representing the instantaneous reward received by the agent on transitioning from one state to next after taking action, $\gamma \in [0, 1)$ is the discount factor, G is the set of goals, the agents need to achieve. The states S and actions A are defined by the set of objects E, predicates Q, and action types Y.

MaRePReL solves GRMGs using a combination of multiagent planning and RL, as shown in Figure 2. A problem instance for a GRMG is defined similarly to in GRMDP [20] as a pair $(s \in S, g \in G)$, where s and g is the initial state and the goal (a partial state), defined by a set of positive or negative literals. The success probability of a joint policy (say π) is the probability of all the goals $g \in G$ being achieved under the policy π. The expected utility of the joint policy is the expected total discounted reward before the policy terminates, either due to the successful completion of goals or timing out from exceeding the maximum possible length of the trajectory.

3.2 Relational Planner

The environment's state can be represented as an abstract planning problem using a planning description language [13]. The hybrid planning domain $D = \langle Q, O, C, M \rangle$, consists of a set of predicates Q that describes the current state, a finite set of operators O which are the high-level actions executable by the agents, a set of ordering constraints C that is necessary to construct a consistent plan, and methods M that can decompose the goal set into an ordered sequence of operators. A multiagent planning (MAP) problem can be defined as follows:

Definition 2 *For a given domain D, a Multiagent Planning (MAP) problem $P = \langle D, S, G, AG \rangle$, consists of the initial state of the problem S, the set of goals G that need to be completed, and a group of agents AG that need to coordinate together to reach the goal state.*

For the above MAP problem, the planner plays a crucial role by controlling the tasks performed by each agent. It maps the target set of goals G into a set of grounded task-specific operators O and distributes them to different agents. Hierarchical Task Network (HTN) planners such as SHOP [30] can be used to generate a total order plan for a given instance of the environment. The grounded plan (high-level plan) and a set of ordering constraints are used to distribute the tasks to create agent-specific plans (sub-plans). A greedy approach is used to schedule tasks that involve forward chaining [22]. It examines the causal links between operators and prevents tasks from being assigned to agents that cannot execute them. The causal links $L = (l_1, l_2, \cdots, l_n)$ define a partial ordering between operations. Each link is of the form $(O_p, \textit{eff}, O_q)$ where $\textit{eff}$ is the effect of completing task O_p and one of the preconditions for task O_q [45]. Our task distributor ensures that the same agent performs the causally linked operations.

For each agent a, the task distributor returns a partial plan $\Pi^a = [o_1, o_2, o_3, ..., o_n]$ where o is an operator with $I(o)$ being the precondition of the operator and $\beta(o)$ being the necessary effects of the operator. Since the operators only consider the action space of the agent currently using them, and the operators are shared among the different agents, all working on the same underlying environment, we can define the sub-goal RMDP $\mathcal{M}_o$ for each operator to solve the problem like in RePReL [20].

3.3 Task-specific Abstraction

While the planner decomposes the task and the task distributor identifies the appropriate agent, the resulting state space can still be prohibitively expensive for effective learning. Consequently, the abstraction reasoner becomes crucial in constructing a smaller state space. In GRMG, states are represented as conjunctions of literals. Like RePReL, prior knowledge that describes the relation between rewards, and sub-goals is then described using an extended First Order Conditional Influence statements (FOCI) [29] called Dynamic FOCI (D-FOCI) statements. D-FOCI statements, represented by an example below, are the first-order language rules used to specify the direct conditional influences between literals in the domain.

Algorithm 1 MaRePReL algorithm

Input: Multiagent Planner $\mathcal{P}$, Operators O, Agents A, goal set g, D-FOCI statements F, num of iterations i, num of episodes per iteration k, batch size b, terminal reward t_R
Output: RL policies $\pi = \{\pi_o | \forall o \in O\}$

1: Initialize the RL policies $\pi = \{\pi_o | \forall o \in O\}$ and buffers $\mathcal{D} = \{\mathcal{D}_o | \forall o \in O\}$
2: **for** iteration $\in i$ **do**
3:     **for** episode $\in k$ **do**
4:         $s \leftarrow$ starting state of the environment
5:         $\Pi \leftarrow \mathcal{P}(s, g)$
6:         $\phi \leftarrow$ Pop the first task for each agent from Π
7:         **while** ϕ is not empty **do**
8:             $actions \leftarrow$ **GetAgentActions**(s, ϕ, π, F, A)
9:             $s, \mathcal{D}, \phi, PlanValid \leftarrow$
                RePReLStep$(s, actions, \mathcal{D}, t_R, \phi, \Pi, F, A)$
10:             **if** not $PlanValid$ **then**
11:                 $\Pi \leftarrow \mathcal{P}(s, g)$ ▷ Recompute the plan
12:                 $\phi \leftarrow$ Pop the first task for each agent in Π
13:             **end if**
14:         **end while**
15:     **end for**
16:     **for** each operator $o \in O$ **do**
17:         Sample batch $\mathcal{D}_b$ from the corresponding buffer $\mathcal{D}_o$
18:         Update Policy π_o using the buffer $\mathcal{D}_b$
19:     **end for**
20: **end for**
21: **return** π

The rules defined over the predicates by a domain expert express the relation between domain predicates at a different time step.

$$pickup(P, T) : \{taxi(T, L1), at(P, L)\} \xrightarrow{+1} in\_taxi(P, T)$$

The above rule states that when executing a task $pickup(P, T)$, only the location $L1$ of taxi T and the pickup location L of passenger P influence the state predicate in_taxi. The relational planner provides agent-specific plans that contain the grounded operators. Substituting the variables grounded by our planner in the D-FOCI statements will provide us with the set of literals on which our task-based RL policies can be trained. If the sub-plan for agent $t1$ contains the grounded operator $pickup(p1, t1)$, we can use the substitution $\theta = \{P/p1, T/t1, L/r, L1/l1\}$ to get the grounding,

$$pickup(p1, t1) : \{taxi(t1, l1), at(p1, r)\} \xrightarrow{+1} in\_taxi(p1, t1)$$

which provides us with information on the relevant state literals for the task $pickup(p1, t1)$ as $taxi(t1, l1)$, $at(p, r)$, and $in\_taxi(p1, t1)$. This implies that the task of picking up $p1$, when assigned to $t1$, only needs the information above, and the locations and in-taxi conditions of other passengers and taxis in the domain can be masked while learning an RL policy.

To summarize, the abstraction reasoner performs two specific steps – first, it uses the domain knowledge as first-order logic statements to construct an abstract MDP and then grounds the MDP to construct the smaller ground MDP that an underlying Deep RL agent can solve. Automating these two steps fully, either

using advances inside the lifted inference community or an LLM-based approach, remains an interesting future direction. At the lowest level are different RL agents, which, after acquiring the MDP, proceed to solve them appropriately.

3.4 MaRePReL Algorithm

Algorithm 1 presents the procedure where we initialize the RL policies and buffers for various operators (**line 1**). The policies learned through our approach are a collection of task-specific operations. While one or more agents still have pending subtasks, we continuously collect trajectories from the environment for different operators, storing them in respective operator buffers (**lines 7-15**). In each episode, we first get the starting state from the environment (**line 4**), and obtain the current tasks for each agent by computing their sub-plans using our relational multiagent planner, which is implemented by combining a SHOP [30] planner with branch and bound scheduling (**line 5-6**). While one or more agents have some task remaining, we first compute the joint actions for the agents based on the current state, agent tasks, task policies, and the D-FOCI statements using the **GetAgentActions** method (**line 8**). Upon obtaining the joint action, we perform a step update using the **RePReLStep** method (**line 9**). This step involves updating the state, buffers, plan, and tasks based on the D-FOCI rules for abstractions. Following RePReL's approach, the method returns the updated components along with a flag indicating the validity of the current plan. If the plan is considered invalid — one or more agents cannot perform the tasks assigned to them — then new agent-specific plans (subplans) are computed, and the agents are assigned new tasks (**lines 10-13**). Once the episode has ended, we train a policy for each operator π_o using a batch sampled from the buffer for the operator (**line 16-19**). Once trained, the final policies for the different operators are returned. The methods **GetAgentActions** and **RePReLStep** are further detailed in the supplementary [1]

4 EXPERIMENTAL RESULTS

We present our results across different tasks in three relational multiagent domains that demonstrate the effectiveness of MaRePReL. We answer the following questions explicitly.

(1) Does **MaRePReL** improve **sample efficiency** ?
(2) Does **MaRePReL** efficiently **transfer** from one task to another?
(3) Does **MaRePReL generalize** to varying number of objects?

4.1 Domains

For the first environment, we extend the taxi domain environment [6] to relational multiagent settings. The goal is to transport passengers from their current locations to their destinations. Passengers are located at four different grid positions – R, G, B, and Y -— requiring coordinated efforts from the taxis for pickup and drop-off, with no two passengers having the same pickup or drop locations. Additionally, the taxis cannot cross each other or occupy the same location. Doing so would cause crashes, terminating a huge negative reward and incurring a heavy penalty.

For the second environment, we extend the office world domain [15] to accommodate multiple agents. The agents are presented

---

Domain	D-FOCI statements	Operators	Relevant Predicates
Multiagent Taxi	taxi-at(T, L_1), move(T, Dir) $\xrightarrow{+1}$ taxi-at(T, L_2) taxi-at(T, L), move(T, Dir) $\xrightarrow{+1}$ R taxi-at$(T1, L1)$, taxi-at$(T2, L2)$, move$(T1, Dir1)$, move$(T2, Dir2)$ $\xrightarrow{+1}$ R **pickup(P, T)** : taxi-at$(T1, L1)$, at(P, L), in-taxi(P) $\xrightarrow{+1}$ in-taxi(P) in-taxi$(P) \rightarrow$ R **drop(P, T)** : taxi-at$(T1, L1)$, in-taxi(P), dest(P, L), at-dest(P) $\xrightarrow{+1}$ at-dest(P) at-dest$(P) \rightarrow$ R	pickup(P, T) drop(P, T)	taxi-at$(T, L1)$, taxi-at$(T2, L2)$, at(P, L), in-taxi(P), move(T, Dir), move$(T2, Dir2)$ taxi-at$(T, L1)$, taxi-at$(T2, L2)$, dest(P, L), in-taxi(P), move(T, Dir), move$(T2, Dir2)$
Multiagent Office	agent-at(A, L_1), move(A, Dir) $\xrightarrow{+1}$ agent-at(A, L_2) agent-at(A, L_1), move(A, Dir) $\rightarrow$ R **visitOrPickup(X, A)** : agent-at$(A, L1)$, at(X, L), with-agent(A, X) $\xrightarrow{+1}$ with-agent(A, X) with-agent$(A, X) \rightarrow$ R **deliver(X, A)** : agent-at$(A, L1)$, with-agent(A, X), office(L), delivered(X) $\xrightarrow{+1}$ delivered delivered$(X) \rightarrow$ R	visitOrPickup(X, A) deliver(X, A)	agent-at(A, L_1), move(A, Dir), at(X, L), with-agent(A, X) agent-at(A, L_1), move(A, Dir), with-agent(A, X), office(L), delivered(X)
Multiagent Dungeon	player-at$(P, L1)$, orientation(P), move(P, Dir) $\xrightarrow{+1}$ player-at$(P, L2)$ player-at(P, L), orientation(P), move$(P, Dir) \rightarrow$ R **attackEnemy(P, E)** : player-at$(P, L1)$, enemy-at$(E, L2)$, player-health(P), player-defense(P) $\xrightarrow{+1}$ player-health(P) player-health$(P) \rightarrow$ R player-at$(P, L1)$, enemy-at$(E, L2)$, enemy-health(E) $\xrightarrow{+1}$ enemy-health(E) enemy-health$(E) \rightarrow$ R **getKeyInDoor(P, K)** : hasKey(P, K), player-at$(P, L1)$, enemy-at$(E, L2)$ $\xrightarrow{+1}$ hasKey(P, K) hasKey(P, K), doorUnlocked(K) $\xrightarrow{+1}$ doorUnlocked(K) doorUnlocked$(K) \rightarrow$ R	attackEnemy(P, E) getKeyInDoor(P, K)	player-at$(P, L1)$, orientation(P), move(P, Dir), enemy-at$(E, L2)$, player-health(P), player-defense(P), enemy-health(E) player-at$(P, L1)$, orientation(P), move(P, Dir), hasKey(P, K), enemy-at$(E, L2)$, DoorUnlocked(K)

Table 1: Example of D-FOCI statements for relational multiagent domains

with a set of tasks they need to complete together. A positive reward is provided to the agents when they complete the tasks, but a bump penalty is given to the agents when they move into the same cell.

For the third environment, we've handcrafted a dungeon grid world inspired by Unity's Dungeon Escape [18]. The agents must defeat different enemies, collect the keys required to unlock the door, and escape the dungeon. The enemies, if attacked, will target the agents back. The agents become incapacitated and unable to take action once their health reaches zero. The environment terminates once all the keys are in the door, or all the agents are dead.

Table 1 lists the D-FOCI statements for the three environments with additional environment details in the supplementary.

4.2 Baselines

We evaluate MaRePReL against several standard MARL algorithms, including Deep Q-Networks with parameter sharing (DQN-PS), Deep Q-Networks as independent learners (DQN-IL), and QMIX [33]. In DQN, each agent maintains its decentralized state-action value function, updating Q-values based solely on local observations

and individual rewards. In contrast, QMIX utilizes a parameterized mixing network to compute a joint Q-value, combining information from all agents. However, it is essential to note that QMIX is not included in the benchmark for the multi-agent dungeon environment due to a key limitation: QMIX requires a fixed number of agents to function properly, which is not guaranteed in the dungeon environment as the number of agents change when an agent dies.

Additionally, we introduce a new baseline called DQN-PE (DQN with Plan Embeddings), inspired by HTN-MTRL [14]. In DQN-PE, the observation for each agent is augmented with a vector embedding of its current sub-plan. This approach leverages the same HTN planner as MaRePReL and encodes the sub-plan string using a BERT embedding model [34]. The string is divided into chunks, mean pooled, and reduced to a $\mathbb{R}^4$ embedding. This baseline allows us to assess whether providing agents with task-specific information in the form of the agent's sub-plan embeddings is sufficient, or whether task hierarchies and state abstractions are required like in MaRePReL. As far as we know, no relational multiagent baselines

Environment	Task 1	Task 2	Task 3
Multiagent Taxi	Transport 2 passengers	Transport 3 passengers	Transport 4 passengers
Multiagent Office	Visiting A, B, C and D	Pickup mail and coffee	Deliver mail and coffee to office
Multiagent Dungeon	Defeat 1 skeleton and escape	Defeat 1 dragon and escape	Defeat 1 skeleton, 1 dragon and escape

Table 2: Tasks to perform in different environments

(a) Taxi Task 1

(b) Taxi Task 2

(c) Taxi Task 3

(d) Office Task 1

(e) Office Task 2

(f) Office Task 3

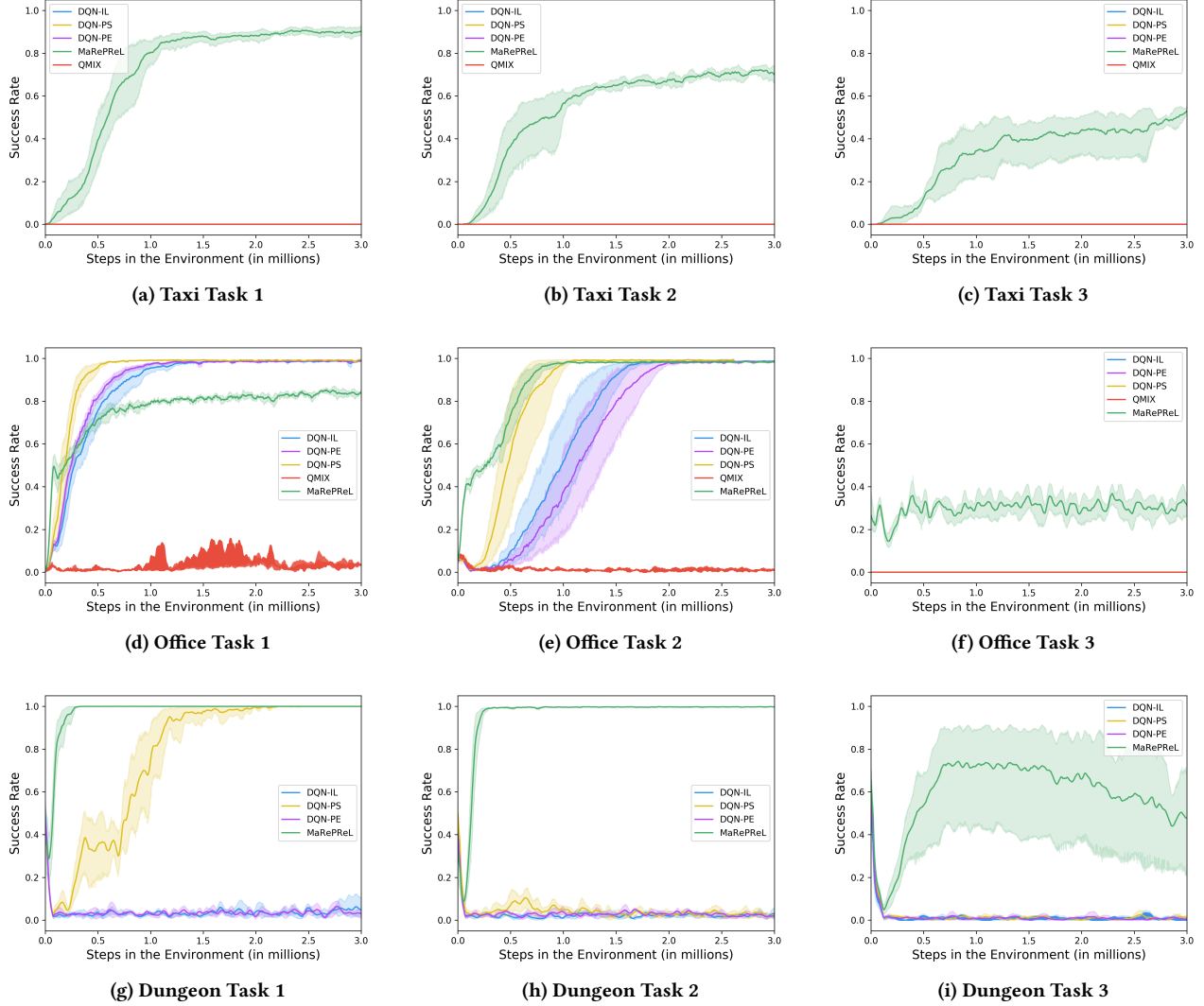(g) Dungeon Task 1

(h) Dungeon Task 2

(i) Dungeon Task 3

Figure 3: The success rates across different methodologies for the Taxi, Office World, and Dungeon domains

are readily available. One could design a relational multiagent baseline by considering hierarchies as a special form of relations (in the lines of the work inside RRL community [5, 43]), but extending them to a multiagent scenario is non-trivial and outside the scope.

4.3 Results

We evaluate sample efficiency, transfer ability and generalization capability of our method against the other baselines across the different tasks (Table 2). The results, averaged over five trials, are presented in Figure 3 (training from scratch), and 4 (training starting with a previously learned policy), where the bold line represents the mean and the shaded region illustrates the variance of the success rate across trials after 3 million environment steps.

Sample Efficiency: In the taxi domain, MaRePReL, unlike DQN and QMIX, was able to learn how to complete tasks 1, 2, and 3,

(a) Taxi Task 1 to Task 3   (b) Office Task 1 to Task 2   (c) Dungeon Task 2 to Task 3
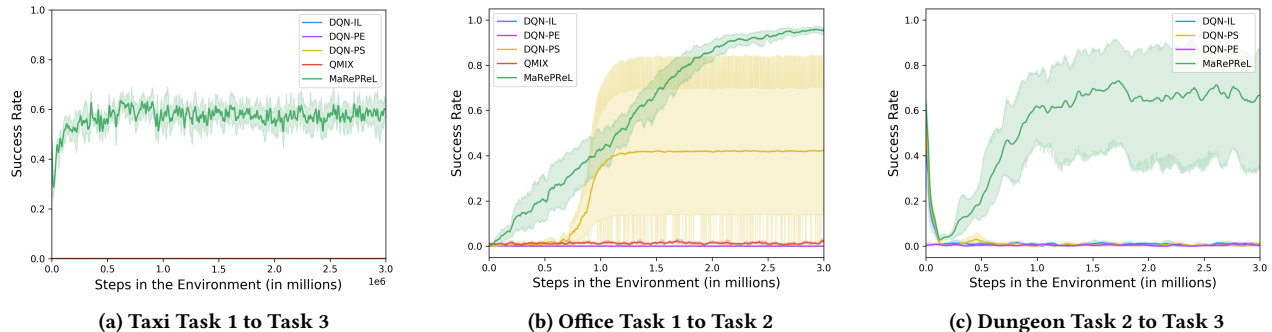
Figure 4: The success rate in case of transferring from policy for one task to another across different methodologies for the Taxi, Office World, and Dungeon Domains

whereas DQN-IL, DQN-PS, DQN-PE and QMIX have a near-zero success rate even after training for 3 million steps (Figure 3 a-c). In the Office World domain, DQN-PS learned an optimal policy for Tasks 1 and 2 (Figure 3 d-e). This can be attributed to the static nature of the goals in the environments. However, as the complexity of the tasks increased, such as in Task 3 (Figure 3 f), none of the baselines could perform while MaRePReL still was able to show early success. In the Dungeon domain, MaRePReL demonstrates robust performance, converging to optimal solutions for Task 1 and 2 while showing a steeper learning curve for Task 3. The baseline of DQN-PS exhibits convergence for Task 1 after one million steps, although notably slower compared to MaRePReL. However, DQN-IL, DQN-PS, and DQN-PE struggled with learning Tasks 2 and 3, showing a near-zero success rate in both cases (Figure 3 g-i). QMIX can't be used in the dungeon environment due to its changing number of agents. Therefore, **Q1** can be answered affirmatively.

Transfer: In the Office World domain, we transfer policies from Task 1 (Visiting A, B, C, D) to Task 2 (Get Mail and Coffee), and in the Dungeon domain, from Task 2 (Dungeon with 1 Dragon) to Task 3 (Dungeon with 1 Dragon and 1 Skeleton). In the Office World experiment, MaRePReL successfully adapted to the new task and nearly achieved 100% success, whereas the baselines struggled— QMIX, DQN-IL, and DQN-PE failed, and DQN-PE had a low success rate (Figure 4b). In the Dungeon experiment, when tasks shared common goals, transferring with MaRePReL facilitated convergence, while neither QMIX nor any DQN baselines showed success (Figure 4c). These results demonstrate the transfer abilities of a relational model, a fact well-known within the relational RL community. The key power of relational models (in our case, relational planner and relational abstract reasoner) lies in their ability to achieve efficient transfer and effective generalization. Therefore **Q2** can be answered affirmatively.

Generalization: In this case, the policies are not randomly initialized; instead, the policies learned for tasks with fewer objects are applied to the new task with more objects. For the taxi world domain, a policy trained on the task of transporting two passengers is applied to the task of transporting four passengers (Figure 4a). MaRePReL significantly improves sample efficiency, achieving Task 3's success rate in less than half a million steps, compared

to 3 million steps for the non-transferred policy. Similarly, one can notice that MaRePReL generalizes to a task presenting an increasing number of enemies to defeat in the Dungeon environment. Other baselines do not demonstrate any initial success or show any performance improvement. Therefore, **Q3** can be answered affirmatively

5 DISCUSSION AND FUTURE WORK

We demonstrated empirically that MaRePReL significantly outperforms traditional MARL approaches, including DQN (independent learners or parameter sharing or with sub-plan embeddings) and QMIX. Our results clearly show the effectiveness of combining a relational planner with an agent-specific task distributor at the higher level and deep RL at the lower level. Significant improvements can be observed in both learning, transfer, and generalization.

Our framework has a few limitations. As the number of operators and agents increases, the search space for the relational planner grows exponentially, posing challenges for generalization, an important future direction. Our current formalism applies only to problems featuring a fully observable state space, and extending it to partially observable spaces necessitates integration with efficient (lifted) probabilistic inference. Moreover, the cooperation shown between agents is loosely coupled as they work in parallel to complete the tasks assigned by a centralized planner. It is possible to extend our approach to tackle challenges in domains that demand coordination among multiple agents [3, 9] by incorporating a partial-order planner along with wait operators. This extension would allow all agents to achieve a state that fulfills the preconditions before performing the joint task. Finally, constructing a fully differentiable system is an interesting direction for future research.

REFERENCES

[1] Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. 2024. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches.* MIT Press, Boston. https://www.marl-book.com

[2] Sebastian Bitzer, Matthew Howard, and Sethu Vijayakumar. 2010. Using dimensionality reduction to exploit constraints in reinforcement learning. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, IEEE, Taipei, Taiwan, 3219–3225.

[3] Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. 2020. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., Virtual, 10707–10717. https://proceedings.neurips.cc/paper/2020/file/7967cc8e3ab559e68cc944c44b1cf3e8-Paper.pdf

[4] Tom Croonenborghs, Karl Tuyls, Jan Ramon, and Maurice Bruynooghe. 2006. Multi-agent relational reinforcement learning: Explorations in multi-state coordination tasks. In *Learning and Adaption in Multi-Agent Systems: First International Workshop, LAMAS 2005, Utrecht, The Netherlands, July 25, 2005, Revised Selected Papers*. Springer, LAMAS, Utrecht, The Netherlands, 192–206.

[5] Srijita Das, Sriraam Natarajan, Kaushik Roy, Ronald Parr, and Kristian Kersting. 2020. Fitted Q-Learning for Relational Domains. arXiv:2006.05595 [cs.LG] https://arxiv.org/abs/2006.05595

[6] Thomas Dietterich. 1999. State Abstraction in MAXQ Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller (Eds.), Vol. 12. MIT Press, Denver. https://proceedings.neurips.cc/paper_files/paper/1999/file/e5a4d6bf330f23a8707bb0d6001dfbe8-Paper.pdf

[7] Thomas G Dietterich. 1998. The MAXQ Method for Hierarchical Reinforcement Learning.. In *ICML*, Vol. 98. ICML, Madison, Wisconsin, 118–126.

[8] Sašo Džeroski, Luc DeRaedt, and Kurt Driessens. 2001. Relational reinforcement learning. *Machine learning* 43 (2001), 7–52.

[9] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. 2023. SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., New Orleans, 37567–37593. https://proceedings.neurips.cc/paper_files/paper/2023/file/764c18ad230f9e7bf6a77ffc2312c55e-Paper-Datasets_and_Benchmarks.pdf

[10] Alberto Finzi and Thomas Lukasiewicz. 2004. Relational Markov Games. In *Logics in Artificial Intelligence*, Jóse Júlio Alferes and João Leite (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 320–333.

[11] Lise Getoor and Ben Taskar. 2007. *Introduction to statistical relational learning*. MIT press, Boston.

[12] Mohammad Ghavamzadeh, Sridhar Mahadevan, and Rajbala Makar. 2006. Hierarchical Multi-Agent Reinforcement Learning. *Autonomous Agents and Multi-Agent Systems* 13, 2 (sep 2006), 197–229. https://doi.org/10.1007/s10458-006-7035-4

[13] Daniel Höller, Gregor Behnke, Pascal Bercher, Susanne Biundo, Humbert Fiorino, Damien Pellier, and Ron Alford. 2020. HDDL: An extension to PDDL for expressing hierarchical planning problems. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. AAAI, New York City, 9883–9891.

[14] Yuyong Hu and Hankz Hankui Zhuo. 2024. Multi-Task Reinforcement Learning with Cost-based HTN Planning. In *2024 5th International Conference on Computer Engineering and Application (ICCEA)*. IEEE, Hangzhou, China, 155–160. https://doi.org/10.1109/ICCEA62105.2024.10603549

[15] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. 2018. Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholm, Sweden, 2107–2116. https://proceedings.mlr.press/v80/icarte18a.html

[16] León Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A McIlraith. 2020. Symbolic plans as high-level instructions for reinforcement learning. In *Proceedings of the international conference on automated planning and scheduling*, Vol. 30. ICAPS, Online, 540–550.

[17] Shariq Iqbal, Robby Costales, and Fei Sha. 2022. ALMA: Hierarchical Learning for Composite Multi-Agent Tasks. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., New Orleans, 7155–7166. https://proceedings.neurips.cc/paper_files/paper/2022/file/2f27964513a28d034530bfdd117ea31d-Paper-Conference.pdf

[18] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. 2020. Unity: A General Platform for Intelligent Agents. arXiv:1809.02627 [cs.LG] https://arxiv.org/abs/1809.02627

[19] Jung In Kim, Young Jae Lee, Jongkook Heo, Jinhyeok Park, Jaehoon Kim, Sae Rin Lim, Jinyong Jeong, and Seoung Bum Kim. 2023. Sample-efficient multi-agent reinforcement learning with masked reconstruction. *PLOS ONE* 18, 9 (09 2023), 1–14. https://doi.org/10.1371/journal.pone.0291545

[20] Harsha Kokel, Arjun Manoharan, Sriraam Natarajan, Balaraman Ravindran, and Prasad Tadepalli. 2021. Reprel: Integrating relational planning and reinforcement learning for effective abstraction. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 31. ICAPS, Guangzhou, China, 533–541.

[21] Harsha Kokel, Sriraam Natarajan, Balaraman Ravindran, and Prasad Tadepalli. 2023. RePReL: a unified framework for integrating relational planning and reinforcement learning for effective abstraction in discrete and continuous domains. *Neural Computing and Applications* 35, 23 (2023), 16877–16892.

[22] Jonas Kvarnström. 2011. Planning for loosely coupled agents using partial order forward-chaining. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 21. ICAPS, Freiburg, Germany, 138–145.

[23] Gen Li, Yuejie Chi, Yuting Wei, and Yuxin Chen. 2022. Minimax-Optimal Multi-Agent RL in Markov Games With a Generative Model. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., New Orleans, 15353–15367. https://proceedings.neurips.cc/paper_files/paper/2022/file/62b4fea131cfd5b7504eae356b75bbd8-Paper-Conference.pdf

[24] Guangxia Li, Gang Xiao, Junbo Zhang, Jia Liu, and Yulong Shen. 2022. Towards relational multi-agent reinforcement learning via inductive logic programming. In *International Conference on Artificial Neural Networks*. Springer, ICANN, Bristol, UK, 99–110.

[25] Michael L. Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning* (New Brunswick, NJ, USA) *(ICML'94)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 157–163.

[26] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6382–6393.

[27] Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, and Luc De Raedt. 2024. From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence* 328 (2024), 104062. https://doi.org/10.1016/j.artint.2023.104062

[28] David Martínez, Guillem Alenyà, and Carme Torras. 2017. Relational reinforcement learning with guided demonstrations. *Artificial Intelligence* 247 (2017), 295–312. https://doi.org/10.1016/j.artint.2015.02.006 Special Issue on AI and Robotics.

[29] Sriraam Natarajan, Prasad Tadepalli, Eric Altendorf, Thomas G. Dietterich, Alan Fern, and Angelo Restificar. 2005. Learning First-Order Probabilistic Models with Combining Rules. In *Proceedings of the 22nd International Conference on Machine Learning* (Bonn, Germany) *(ICML '05)*. Association for Computing Machinery, New York, NY, USA, 609–616. https://doi.org/10.1145/1102351.1102428

[30] Dana S. Nau, Yue Cao, Amnon Lotem, and Hector Muñoz-Avila. 1999. SHOP: Simple Hierarchical Ordered Planner. In *International Joint Conference on Artificial Intelligence*. IJCAI, Stockholm, Sweden, 968–973. https://api.semanticscholar.org/CorpusID:2329216

[31] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. 2021. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–35.

[32] Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. 2016. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis lectures on artificial intelligence and machine learning* 10, 2 (2016), 1–189.

[33] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholm, Sweden, 4295–4304. https://proceedings.mlr.press/v80/rashid18a.html

[34] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3982–3992. https://doi.org/10.18653/v1/D19-1410

[35] Silvana Roncagliolo and Prasad Tadepalli. 2004. Function approximation in hierarchical relational reinforcement learning. In *Proceedings of the ICML-2004 Workshop on Relational Reinforcement Learning*. ICML, Banff, Alberta, 1–5.

[36] Satinder P Singh. 1992. Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the tenth national conference on Artificial intelligence*. AAAI, San Jose, California, 202–207.

[37] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.

[38] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112, 1-2 (1999), 181–211.

[39] Prasad Tadepalli, Robert Givan, and Kurt Driessens. 2004. Relational reinforcement learning: An overview. In *Proceedings of the ICML*, Vol. 4. ICML, Banff, Alberta, 1–9.

[40] Guy Van den Broeck, Kristian Kersting, Sriraam Natarajan, and David Poole. 2021. *An Introduction to Lifted Probabilistic Inference*. MIT Press, Boston.

[41] M. van Otterlo. 2005. *A Survey of Reinforcement Learning in Relational Domains.* Number 05-31 in CTIT Technical Report Series. Centre for Telematics and Information Technology (CTIT), Netherlands. Imported from HMI.

[42] Martijn van Otterlo. 2012. Solving relational and first-order logical markov decision processes: A survey. In *Reinforcement learning: State-of-the-art.* Springer, Berlin, Heidelberg, 253–292.

[43] Chenggang Wang, Saket Joshi, and Roni Khardon. 2008. First order decision diagrams for relational MDPs. *Journal of Artificial Intelligence Research* 31 (2008), 431–472.

[44] Chenggang Wang and Roni Khardon. 2010. Relational partially observable MDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 24. AAAI Press, Atlanta, Georgia, USA, 1153–1158.

[45] Daniel S Weld. 1994. An introduction to least commitment planning. *AI magazine* 15, 4 (1994), 27–27.

[46] Zhiwei Xu, Yunpeng Bai, Bin Zhang, Dapeng Li, and Guoliang Fan. 2023. HAVEN: Hierarchical Cooperative Multi-Agent Reinforcement Learning with Dual Coordination Mechanism. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 10 (Jun. 2023), 11735–11743. https://doi.org/10.1609/aaai.v37i10.26386

[47] Jiachen Yang, Igor Borovikov, and Hongyuan Zha. 2020. Hierarchical Cooperative Multi-Agent Reinforcement Learning with Skill Discovery. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems* (Auckland, New Zealand) *(AAMAS '20).* International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1566–1574.

[48] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control* 325 (2021), 321–384.