Exploiting Relational Planning and Task-Specific Abstractions for Multiagent Reinforcement Learning in Relational Domains

Ranveer Singh [1], **Nikhilesh Prabhakar**[1], **Sriraam Natarajan** [1], **Prasad Tadepalli** [2]

[1]The University of Texas at Dallas
[2]Oregon State University
{ranveer.singh, nikhilesh.prabhakar, sriraam.natarajan}@utdallas.edu, tadepall@eecs.oregonstate.edu

Abstract

Multiagent Reinforcement Learning poses significant challenges due to the exponential growth of state, object, 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 generalization.

Introduction

Multiple agents learning to reason and act under uncertainty in the presence of a varying number of objects and relations while reasoning about different levels of abstraction has long been a cherished goal of AI. Several venues have tackled a subset of these problems. Reinforcement learning (RL) (Sutton and Barto 2018) and multiagent RL (Albrecht, Christianos, and Schäfer 2024) techniques have long been developed for learning under uncertainty and with multiple agents respectively. They have been extended to hierarchical domains for a long time to allow for reasoning at multiple levels of abstractions (Dietterich 1998; Singh 1992; Sutton, Precup, and Singh 1999).

Statistical Relational Learning and AI (StaRAI) (Getoor and Taskar 2007; Raedt et al. 2016), 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 (Džeroski, DeRaedt, and Driessens 2001) is a difficult task and while some methods exist (van Otterlo 2012), they do not scale for large tasks and are certainly not easily extensible to multiagent settings. More recently, there is a significant interest in leveraging the combination of classical Planning and RL (Kokel et al. 2021; Illanes et al. 2020) that allows solving complex tasks by effectively decomposing higher-level tasks and efficiently learning generalizable lower-level policies.

Inspired by the success in different fields, we propose a method that leverages the power of a relational planner to act as a centralized controller for multiagent learning in noisy, relational domains. Our proposed approach, called *multiagent relational planning and RL (MaRePReL)*, uses planning for task decomposition, centralized control, and agent allocation, StaRAI for reasoning and constructing task-specific representations, and deep RL for effective and efficient learning with these smaller representations.

We make the following key contributions: (1) as far as we are aware, we developed the first multiagent system that generalizes across multiple objects and relations; (2) we developed MaRePReL, an integrated planning and learning formalism that is capable of multiagent learning under uncertainty in relational domains; (3) We present the architecture and discuss its key salient features; and (4) finally, demonstrate the effectiveness and generalization abilities of this approach in a relational, multiagent extension of the famous taxi domain (Dietterich 1998).

The rest of the paper is organized as follows: after presenting the required background, we present the architecture in detail and its key features. Then, we discuss our domains and results before outlining areas for future research.

Background

Multiagent RL (MARL) extends reinforcement learning to systems with multiple agents, where they interact with an environment to maximize cumulative rewards. MARL introduces unique challenges. First, the curse of dimensionality arises due to an exponential increase in state and action spaces as the number of agents grows. Second, is the non-stationary nature of multiagent environments, where the environment evolves independently of an agent due to the actions of the other agents. Finally, MARL requires large amounts of data making it sample inefficient.

The curse of dimensionality has typically been addressed using functional approximation techniques (Bitzer, Howard, and Vijayakumar 2010). Centralized training and decentralized execution frameworks like QMIX and MADDPG have been developed to deal with the non-stationary nature of the multiagent environments (Kraemer and Banerjee 2016; Rashid et al. 2020; Lowe et al. 2017). Sample inefficiency is addressed using generative modeling or mask reconstruction algorithms (Li et al. 2022; Kim et al. 2023).

Hierarchical approaches (HMARL), utilize task decomposition and hierarchical structures in multiagent settings

to define appropriate task abstractions. Task hierarchy abstraction helps reduce the curse of dimensionality and sample inefficiency by filtering out irrelevant parts of the state space. Additionally, the structured task hierarchies facilitate effective communication between agents, enhancing their ability to handle non-stationarity in multiagent systems (Ghavamzadeh, Mahadevan, and Makar 2006).

However, the algorithms discussed so far are tailored for tasks with non-relational representations, limiting their applicability in relational domains like Taxi-world (Dietterich 1999) where the states (and actions) are characterized as a set of relations between objects and their properties. The number of objects for a given domain may not be fixed (Džeroski, DeRaedt, and Driessens 2001).

Planning and RL methods construct a two-level system with a higher-level planner inducing the Markov decision process (MDPs) for the lower-level RL to solve. One such recent framework, RePReL (Kokel et al. 2021), employs a hierarchical relational planner to implement task-specific policies and uses Deep RL to work on hybrid relational domains (Kokel et al. 2023). 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 MDPs that can be solved efficiently.

We extend this framework to multiagent settings by using the insight that the planner could be used as the task scheduler for different agents. In our setting, there is a single planner at the higher level which acts as a centralized controller, and several RL agents that learn the lower-level policies. The key difference to the RePReL framework is to not just use the planner for decomposing the tasks, but in addition, to assign these sub-tasks to specific RL agents.

Multi Agent Relational Planning and Reinforcement Learning (MaRePReL)

We consider the problem of coordinating multiple agents to solve relational tasks. We use a combination of relational, hierarchical planning, and deep reinforcement learning with the overall framework having the following components.

1. **Planner as controller:** Our planner acts as a centralized controller by taking the current state as input and creating a set of agent-specific plans. It consists of the following components:
   (a) **Relational Planner:** Since our approach aims to generalize to an increasing number of tasks and objects, we implement a relational HTN (i.e., hierarchical task network) planner to decompose the goals into a temporally ordered series of sub-goals.
   (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 to divide the ordered plan provided into agent-specific sub-plans using agent constraints for the different tasks.

   The planner-distributor combination serves as the centralized controller.
2. **Abstraction Reasoner:** Following RePRel, we employ the use of Dynamic-First Order Conditional Influence (D-FOCI) statements (Natarajan et al. 2005a) 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 then use. In our work, this step is still hand-crafted as the original work, and leveraging lifted inference (Van den Broeck et al. 2021) to perform this step automatically remains a future direction.
3. **Multiple Reinforcement Learners:** Given the current sub-task from the planner, the RL agent identified by the task distributor uses the smaller state space defined by the abstraction reasoner to learn a generalizable, task-specific policy. The advantage of this is that these policies can be shared by the agents when solving similar tasks (as we demonstrate in our experiments).

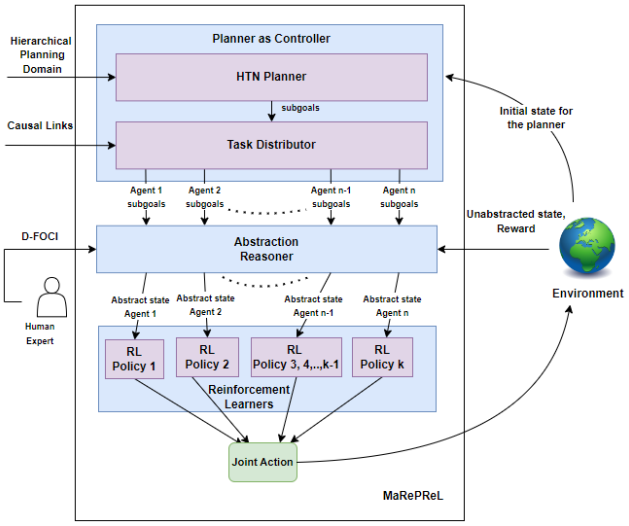


Figure 1: MaRePReL Architecture

The broad overview of our proposed approach is presented in Figure 1. To recall, the planner decomposes the higher-level tasks into appropriate lower-level tasks, and the distributor identifies the appropriate RL agent for the current sub-task, thus making this combination an effective centralized controller. Given the sub-task and the current state (which can be exponentially large), the abstraction reasoner constructs the smaller state space. Finally, the RL agent either learns the policy or executes ones if already learned (for instance, agent $A1$ might have learned the *pickup* sub-task that can be used directly by agent $A2$ when it is required to execute this specific sub-task. We now present this architecture in greater detail.

We present our multiagent environment as a Markov game and build upon the framework of relational Markov games, (Finzi and Lukasiewicz 2004), by expanding its capabilities to address goal-oriented problems.

Definition 1: *A goal-directed relational Markov Games (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 states,* A^i *is the set of actions for the* i^{th} *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 $a \in A$, $R^i = S \times A \times S \rightarrow R$ is the reward function for the i^{th} agent, representing the instantaneous reward received by the agent on transitioning from one state to next after taking an action, $\gamma \in [0, 1)$ is the discount factor G is the set of goals, the agents are tasked with achieving. 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 1. A problem instance for a GRMG is defined similarly to in GRMDP (Kokel et al. 2021) as a pair $(s \in S, g \in G)$, where s and g are the initial and the goal state, defined by a set of positive or negative literals. A solution provided by our approach is a joint policy that originates from state s and leads to a state satisfying all goals in g.

Relational Planner

The first module of our algorithm is the relational planner. The environment's state can be represented as an abstract planning problem using a planning description language (Höller et al. 2019). 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 an ordered sequence of tasks to each agent, by decomposing the target set of goals G into a set of grounded task-specific operators O. Hierarchical Task Network planners such as SHOP (Nau et al. 1999) can be used to generate a total order plan for a given instance of the environment. The grounded plan along with a set of ordering constraints is used to distribute the tasks to create agent-specific plans. A greedy approach is used to schedule tasks that involve forward chaining (Kvarnström 2011). It does so by examining the causal links between operators and preventing assignments of tasks to agents that cannot execute them. Given causal links $L = (l_1, l_2, \cdots, l_n)$, which defines the partial ordering between operations as a link (O_p, eff, O_q) where eff is the effect of completing task O_p and one of the preconditions for task O_q (Weld 1994). Our task distributor schedules tasks to different agents ensuring the operations that are causally linked are performed by the same agent.

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 $\beta(o)$ which is the necessary effects of the operator. Since the operators only consider the action space of the agent currently using them,

Algorithm 1: MaRePReL algorithm

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

```
Initialize the RL policies π_o and buffer D_o   ∀o ∈ O
for iteration ∈ i do
    for episode ∈ k do
        s ← starting state of the environment
        Π ← P(s, g)
        φ ← Pop the first task for each agent from Π
        while φ is not empty do
            actions ← GetAgentActions(s, φ, π, F, A)
            s, D, φ, PlanValid ←
                PerformStep(s, actions, D, t_R, φ, Π, F, A)
            if not PlanValid then
                Π ← P(s, g) {Recompute the plan}
                φ ← Pop the first task for each agent in Π
            end if
        end while
    end for
    for each option o ∈ O do
        Sample batch D_b from the corresponding buffer D_o
        Update Policy π_o using the buffer D_b
    end for
end for
return π_o   ∀o ∈ O
```

and the operators are shared among the different agents, all with the same underlying, we can define the sub-goal RMDP $\mathcal{M}_o$ for each operator to solve the problem like in RePReL (Kokel et al. 2021).

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 conveyed as conjunctions of literals, and we leverage our understanding of how predicates impact rewards and option goals. To extract this knowledge, we employ the extension to First Order Conditional Influence statements (FOCI) (Natarajan et al. 2005b) to Dynamic FOCI (D-FOCI) statements as used in RePReL (Kokel et al. 2021). D-FOCI statements, represented with an example below are the first-order language rules used to specify the direct conditional influences between literals in the domain. 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 following 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.

Proposed Algorithm

In the MaRePReL learning procedure, outlined in Algorithm 1, we initialize the RL policies and buffers for various operators at the start (**line 1**). We obtain a partial plan for each agent by utilizing our relational multiagent planner implemented through a SHOP (Nau et al. 1999) planner with branch and bound scheduling. 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**). The joint action for the agents is computed based on the current state, tasks, policies, and D-FOCI statements using the **GetAgentActions** method (**line 8**). Upon obtaining the joint action, we perform a step update using the **PerformStep** method. 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 deemed invalid, a new plan is computed, and agent tasks are reassigned (**lines 10-12**). Subsequently, the policies are updated based on the operations buffer. The methods **GetAgentActions** and **PerformStep** This iterative process continues until all subtasks are completed, ensuring the MaRePReL algorithm adapts dynamically to the evolving multiagent environment.

Experimentation and Results

We present our results in the multiagent relational taxi domain and answer the following questions explicitly. Evaluation on more domains is our immediate future direction.

1. **Sample Efficiency:** Does **MaRePReL** improve sample efficiency compared to standard baselines?
2. **Generalization:** Does **MaRePReL** generalize to varying number of objects?

Relational Multiagent Taxi World

For our experiments, we extend the relational taxi domain environment (Dietterich 1999) to a relational multiagent setting. The domain involves two taxis navigating a grid to transport passengers. The goal is to learn a multiagent policy for transporting 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. An important property in this relational multiagent taxi world is that Taxis cannot cross each other or occupy the same location in any state since doing so would cause crashes. We have designed the environment to return huge negative rewards and terminate the episode whenever a crash occurs. This property of the environment introduces non-stationarity in the environment since the actions of one agent are influenced by the actions of other agents, posing challenges for independent policies. We evaluate our algorithms on three tasks of transporting passengers to their respective destinations

1. **Task 1:** Transporting 2 passengers
2. **Task 2:** Transporting 3 passengers
3. **Task 3:** Transporting 4 passengers

We assume that only one passenger is allowed in the taxi and for ease of discretization, that each grid location can have at most one passenger in it. Relaxing these assumptions is beyond the scope of this paper.

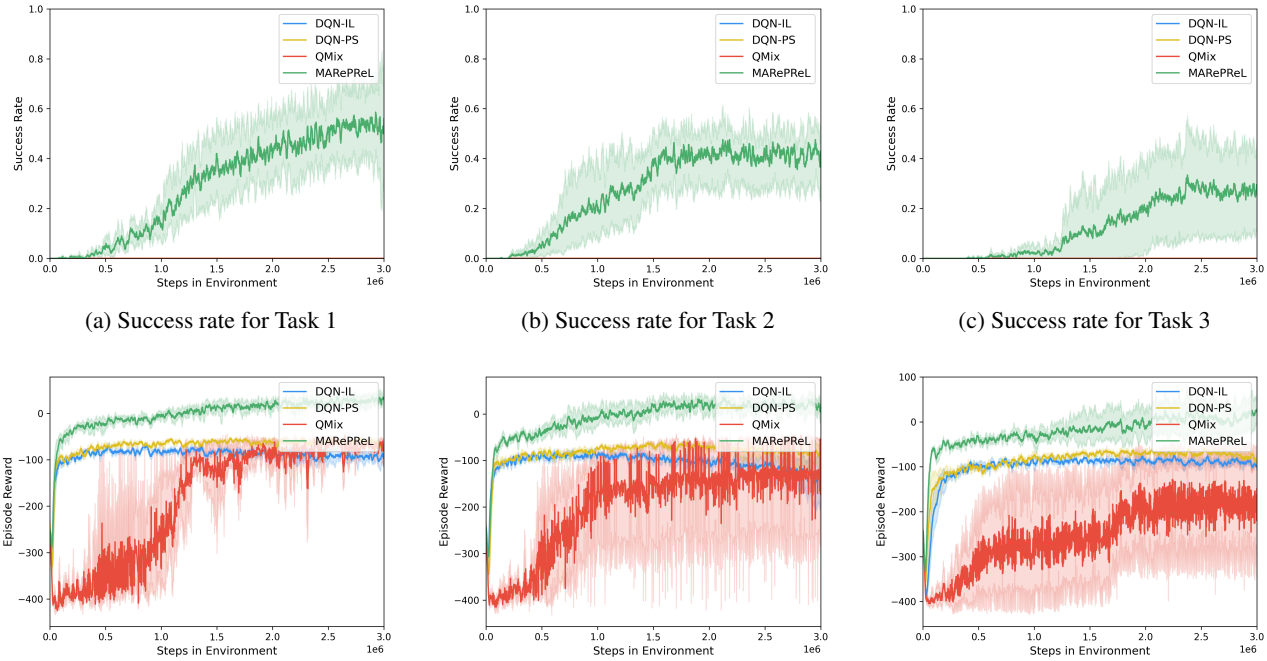
Baselines

We compare MaRePReL against other multiagent baselines: independent DQN (IDQN) (Mnih et al. 2015) (with and without full parameter sharing) and QMIX (Rashid et al. 2020). In IDQN, each agent maintains a decentralized state-action value function, updating its Q-values solely based on local observations and received rewards. For QMIX, a parameterized mixing network computes the joint Q-value.

Results

Sample Efficiency: MaRePReL is compared against QMIX and IDQN. All three algorithms are limited to 3 million environment steps. Unlike MaRePReL, IDQN and QMIX converge to suboptimal rewards, as shown in **Figure 3**. While analyzing the success rate (**Figure 2**), our algorithm did not yet converge to an optimal policy, but it exhibits progress in learning the task. In contrast, IDQN and QMIX show near 0 success rates. The difficulty in learning can be attributed to the non-stationarity of the environment and the high penalty of crashes. MaRePReL demonstrates statistically significant higher sample efficiency than the compared algorithms, affirmatively answering $Q1$.

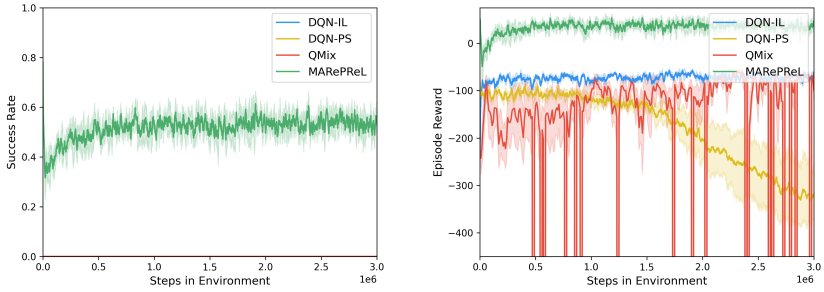
Generalization: In this set of experiments, the policies are not randomly initialized. Instead, we learn the policy from Task 1 (transport two passengers) and apply it to Task 2 (transport three passengers). MaRePReL significantly improves sample efficiency, achieving Task 2's success rate and episode reward in less than half a million steps, compared to 3 million steps for the non-transferred policy. Other algorithms do not show improved performance since they have



(a) Success rate for Task 1 (b) Success rate for Task 2 (c) Success rate for Task 3

(d) Episode reward for Task 1 (e) Episode reward for Task 2 (f) Episode reward for Task 3

Figure 2: The plots shown above compare the success rate(a-c), and average episode reward mean (d-f) of MaRePReL with the DQN (Independent Learners), DQN (Parameter Sharing), and QMIX baselines across Task 1, 2, and 3



(a) Success rate for Transfer task (b) Episode reward for Transfer task

Figure 3: The plots show the success rate and the episode reward for task 2 across different algorithms after learning policies for task 1 and transferring the learned policies for task 2

no initial success rate. Regarding rewards, QMIX quickly converges to a similar suboptimal reward. At the same time, DQN without parameter sharing remains unchanged, and DQN with parameter sharing exhibits a significant decrease later in the run.

Discussion and Future Work

As can be observed from the experiments, MaRePReL significantly outperforms traditional MARL approaches, including IDQN (with and without parameter sharing) and QMIX. Our results (admittedly from a single domain) demonstrate the effectiveness of combining a relational planner with an agent-specific task distributor at the higher level and deep reinforcement learning at the lower level. Significant improvements can be observed in both learning and generalization. However, the scalability of our methodology with an increasing number of agents remains untested, and we plan to expand our testing to other multiagent domains, comparing it with additional baselines.

Our approach 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. Our current formalism applies only to problems featuring a fully observable state space. 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 (Samvelyan et al. 2019; Christianos, Schäfer, and Albrecht 2020) 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. Our hand-coded abstraction reasoner can be replaced by an effective lifted inference method. Finally, constructing a full end-to-end differentiable system is an interesting direction for future research.

Acknowledgements

NP, RS, and SN gratefully acknowledge the support of ARO award W911NF2010224. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily those of the ARO or the U.S. government

References

Albrecht, S. V.; Christianos, F.; and Schäfer, L. 2024. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press.

Bitzer, S.; Howard, M.; and Vijayakumar, S. 2010. Using dimensionality reduction to exploit constraints in reinforcement learning. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3219–3225.

Christianos, F.; Schäfer, L.; and Albrecht, S. 2020. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 10707–10717. Curran Associates, Inc.

Dietterich, T. 1999. State Abstraction in MAXQ Hierarchical Reinforcement Learning. In Solla, S.; Leen, T.; and Müller, K., eds., *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Dietterich, T. G. 1998. The MAXQ Method for Hierarchical Reinforcement Learning. In *ICML*, volume 98, 118–126.

Džeroski, S.; DeRaedt, L.; and Driessens, K. 2001. Relational reinforcement learning. *Machine learning*, 43: 7–52.

Finzi, A.; and Lukasiewicz, T. 2004. Relational Markov Games. In Alferes, J. J.; and Leite, J., eds., *Logics in Artificial Intelligence*, 320–333. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-30227-8.

Getoor, L.; and Taskar, B. 2007. *Introduction to statistical relational learning*. MIT press.

Ghavamzadeh, M.; Mahadevan, S.; and Makar, R. 2006. Hierarchical Multi-Agent Reinforcement Learning. *Autonomous Agents and Multi-Agent Systems*, 13(2): 197–229.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2019. HDDL–A Language to Describe Hierarchical Planning Problems. *International Workshop on HTN Planning (ICAPS), 2019*.

Illanes, L.; Yan, X.; Icarte, R. T.; and McIlraith, S. A. 2020. Symbolic plans as high-level instructions for reinforcement learning. In *Proceedings of the International Conference on automated planning and Scheduling*, volume 30, 540–550.

Kim, J. I.; Lee, Y. J.; Heo, J.; Park, J.; Kim, J.; Lim, S. R.; Jeong, J.; and Kim, S. B. 2023. Sample-efficient multi-agent reinforcement learning with masked reconstruction. *PLOS ONE*, 18(9): 1–14.

Kokel, H.; Manoharan, A.; Natarajan, S.; Ravindran, B.; and Tadepalli, P. 2021. Reprel: Integrating relational planning and reinforcement learning for effective abstraction. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, 533–541.

Kokel, H.; Natarajan, S.; Ravindran, B.; and Tadepalli, P. 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): 16877–16892.

Kraemer, L.; and Banerjee, B. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190: 82–94.

Kvarnström, J. 2011. Planning for loosely coupled agents using partial order forward-chaining. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 21, 138–145.

Li, G.; Chi, Y.; Wei, Y.; and Chen, Y. 2022. Minimax-Optimal Multi-Agent RL in Markov Games With a Generative Model. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 15353–15367. Curran Associates, Inc.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Proceedings of the 31st International Conference on Neural Information*

Processing Systems, NIPS'17, 6382–6393. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518: 529–533.

Natarajan, S.; Tadepalli, P.; Altendorf, E.; Dietterich, T. G.; Fern, A.; and Restificar, A. 2005a. Learning First-Order Probabilistic Models with Combining Rules. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, 609–616. New York, NY, USA: Association for Computing Machinery. ISBN 1595931805.

Natarajan, S.; Tadepalli, P.; Altendorf, E.; Dietterich, T. G.; Fern, A.; and Restificar, A. 2005b. Learning first-order probabilistic models with combining rules. In *Proceedings of the 22nd international conference on Machine learning*, 609–616.

Nau, D. S.; Cao, Y.; Lotem, A.; and Muñoz-Avila, H. 1999. SHOP: Simple Hierarchical Ordered Planner. In *International Joint Conference on Artificial Intelligence*.

Raedt, L. D.; Kersting, K.; Natarajan, S.; and Poole, D. 2016. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis lectures on artificial intelligence and machine learning*, 10(2): 1–189.

Rashid, T.; Samvelyan, M.; De Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2020. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *J. Mach. Learn. Res.*, 21(1).

Samvelyan, M.; Rashid, T.; De Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G.; Hung, C.-M.; Torr, P. H.; Foerster, J.; and Whiteson, S. 2019. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*.

Singh, S. P. 1992. Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the National Conference on Artificial Intelligence*, 10, 202. Citeseer.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book. ISBN 0262039249.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211.

Van den Broeck, G.; Kersting, K.; Natarajan, S.; and Poole, D. 2021. *An Introduction to Lifted Probabilistic Inference*. MIT Press.

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

Weld, D. S. 1994. An introduction to least commitment planning. *AI magazine*, 15(4): 27–27.