Towards Multi-Agent Model-Based Reinforcement Learning in Discrete Non-Markovian Reward Decision Processes

Alessandro Trapasso[1], Marcello Bavaro[2], Francesco Amigoni[2], Luca Iocchi[1] and Fabio Patrizi[1]

[1]Sapienza University of Rome
[2]Politecnico di Milano

Abstract

Non-Markovian Reward Decision Processes have proven very effective in defining and solving complex tasks for autonomous agents, and recent work has focused on devising relative models and algorithms. When applied to multiple agents, they can effectively define complex multi-agent behaviors. In this paper, we discuss the main advantages in using a Multi-Agent Model-Based Reinforcement Learning approach for solving complex tasks in Multi-agent systems with temporal goals. We use the challenging scenario of Multi-Agent Pickup and Delivery as a case study to illustrate potential benefits of the proposed approach.

Keywords

Multi-Agent Systems, Model-Based Reinforcement Learning, Non-Markovian Reward Decision Processes, Multi-Agent Pickup and Delivery

1. Introduction

Reinforcement Learning (RL) has exhibited exceptional results in different domains, due to its ability to deal with complex environments, even with minimal prior knowledge. A crucial factor in designing RL tasks is the definition of the reward function, i.e., a real-valued function capturing the agent's task. Indeed, when the task complexity grows, defining effective reward functions becomes increasingly challenging, especially if temporal sub-tasks are involved. A body of research has investigated this problem, leading to the introduction of Non-Markovian Rewards (NMRs) and Non-Markovian Reward Decision Processes (NMRDPs) as models for capturing complex tasks, as well as to the adoption of temporal logic and automata-based formalisms for the specification of NMRs – see, e.g., [1, 2, 3, 4].

A fundamental result of these works has been proving that every NMRDP $\mathscr{M}$ can be mapped into an equivalent Markov Decision Process (MDP) $\mathscr{M}'$, representing the NMR as a suitable Deterministic Finite-state Automaton (DFA), modeling the synchronous joint dynamics of the environment and the NMR. It has been shown that standard RL algorithms, in particular model-free, can be used for learning in such a joint model [4, 5]. One problem with this approach is that model-free RL disregards learning the transition and reward functions of the (joint) environment, thus making it impossible for the agent to distinguish between state changes due to transitions of $\mathscr{M}$ and those due to transitions of the NMR, in turn resulting in the agent's impossibility to take full advantage of the knowledge about $\mathscr{M}$, possibly acquired with past experience. Model-based approaches also suffer from a similar problem, if used off-the-shelf. This problem is even more relevant in the Multi-Agent RL (MARL) context [6] where sample efficiency and scalability are key success factors.

AAPEI '24: 1st International Workshop on Adjustable Autonomy and Physical Embodied Intelligence, October 20, 2024, Santiago de Compostela, Spain.

trapasso@diag.uniroma1.it (A. Trapasso); marcello.bavaro@polimi.it (M. Bavaro); francesco.amigoni@polimi.it (F. Amigoni); iocchi@diag.uniroma1.it (L. Iocchi); patrizi@diag.uniroma1.it (F. Patrizi)
https://alee08.github.io/ (A. Trapasso)
0000-0001-5431-6607 (A. Trapasso); 0009-0008-8373-3424 (M. Bavaro); 0000-0001-8146-6213 (F. Amigoni); 0000-0001-9057-8946 (L. Iocchi); 0000-0002-9116-251X (F. Patrizi)

In this short paper, we discuss the use of model-based RL approaches in Multi-Agent Systems (MAS) with non-Markovian rewards, focusing on two important techniques: 1) decoupling and separately learning the Markovian environment transitions and rewards and the non-Markovian reward dynamics; 2) sharing the learned environment dynamics among the agents during training. The use of these techniques leads to solution methods that are significantly more sample-efficient than standard model-free approaches.

More specifically, we consider homogeneous MAS, where the agents act in the same environment but have different non-Markovian tasks. In this context, each agent can learn separately the environment dynamics and the reward dynamics and can share the learned model of the environment dynamics with the other agents during training. This approach can greatly reduce the number of samples needed to perform the individual tasks, thus leading to a significant increase in the efficiency of the overall multi-agent learning process. We demonstrate the applicability and the scalability of the approach with an experimental analysis over the Multi-Agent Pickup and Delivery (MAPD) domain [7], showing the ability of the approach to deal with novel variants of the problem.

Related Work. Recent active research lines concern model-based RL and RL with NMRs. Gaon and Brafman [8] describe an application of the R-MAX algorithm for NMRDPs which incrementally extends the MDP state space with the states of a reward automaton to learn an NMR. They use a state-action representation similar to that of the baseline R-MAX used for the experiments in this paper. In comparison, our approach exploits a factorization of the environment's dynamics to improve sample efficiency. A different approach to model-based RL for NMRDPs, based on estimating the parameters of a fractional dynamical model without explicitly representing the NMR is proposed by Gupta et al. [9]; differently, we explicitly represent the NMR. Hierarchical structures have been recently explored [10], enabling the agents to dynamically change strategy to adapt to complex environments. These works mainly focus on learning *options* [11] for reaching goals, not considering NMRs. Another line uses temporal logics over finite traces (LTL_f/LDL_f [12]) to specify and optimize complex RL behaviors trough reward shaping [4, 5]; these works focus on model-free approaches, while we consider model-based ones. Finally, a large body of work aims at learning the NMR model, e.g., [8]; while related, this is out of this paper's scope, which addresses model-based RL for *given* NMR specifications. To the best of our knowledge, no other work has devised model-based RL algorithms for discrete NMRDPs, factorizing the environment's and reward's dynamics to improve sample efficiency.

2. Background

A *Markov Decision Process* (MDP) is a tuple $\mathscr{M} = (S, A, \delta, R_E)$, with: S the finite set of states, A the finite set of actions, $\delta : S \times A \to Pr(S)$ the transition function returning, for $s \in S$ and $a \in A$, a probability distribution $P(s'|s,a)$ over S, and $R_E : S \times A \times S \to \mathbb{R}$ the reward function. A *policy* ρ for an MDP $\mathscr{M}$ is a function $\rho : S \to A$. The *return* $v^\rho(s)$ of a policy ρ from s is $v^\rho(s) = E[\sum_{i=0}^{\infty} \gamma^i r_i]$, with $r_i = R_E(s_i, \rho(s_i), s_{i+1})$, $s_0 = s$, and *discount factor* $\gamma \in [0,1]$. *Reinforcement Learning* (RL) consists in finding an *optimal policy* ρ^* guaranteeing maximal return $v^*(s)$ from every $s \in S$. The Q-function is $Q : S \times A \to \mathbb{R}$ s.t. $Q(s,a)$ is the return obtained by executing a in s, then acting optimally. Given Q, an optimal policy is $\rho^*(s) = \text{argmax}_a Q(s,a)$. In RL, δ, R_E, and Q are unknown. The main solution approaches are *model-free* and *model-based*, with representative algorithms Q-LEARNING [13] and R-MAX [14]. The former seeks ρ^* by learning Q, the latter by learning δ and R_E.

A *Non-Markovian Reward Decision Process* (NMRDP) is a tuple $\mathscr{N} = (\mathscr{M}, R_Q)$, with $\mathscr{M}$ an MDP and $R_Q : S^+ \to \mathbb{R}$ the *Non-Markovian Reward* (NMR) function, returning rewards based on $\mathscr{M}$ histories. Several models for R_Q exist, e.g., Reward Machines (RM) [4] or Restraining Bolts [15]. For simplicity we assume R_Q based on a *Deterministic Finite-state Automaton* (DFA) $\mathscr{A} = (S, Q, q_0, \eta, F)$, with S the input alphabet, Q the finite set of states, $q_0 \in Q$ the initial state, $\eta : Q \times S \to Q$ the transition function, and $F \subseteq Q$ the set of final states; R_Q returns reward $R_Q(q, s, q')$ based on last $\mathscr{A}$'s transition (q, s, q'). We adopt this form for presentation convenience only; experiments are carried out using RMs.

In an NMRDP $\mathscr{N} = (\mathscr{M}, R_Q)$ with $\mathscr{M} = (S, A, \delta, R_E)$, rewards are based on trajectories. The reward

on trajectory $s_0a_0 \cdots sas' \in (S \times A \times S)^+$ is $R_E(s,a,s') + R_Q(q,s',q')$, with q the state reached by $\mathscr{A}$ on history $s_0 \cdots s$ and q' on $s_0 \cdots ss'$. This yields a different RL problem, for which direct use of existing approaches is inadequate. RL with NMRs can be reduced to the Markovian case, by accessing the state of the NMR's DFA, which thus becomes a component of $\mathscr{M}$'s state. This is detailed in [3], which shows that an NMRDP $\mathscr{N} = (\mathscr{M}, R_Q)$ is equivalent to an MDP $\mathscr{M}' = (S', A, \delta', R')$ modeling the synchronous execution of $\mathscr{M}$ and $\mathscr{A}$, with $S' = S \times Q$, δ' capturing joint transitions of $\mathscr{M}$ and $\mathscr{A}$, and $R' : S' \times A \times S'$ a Markovian reward function. While this reduction makes standard RL solution methods suitable for non-Markovian settings (e.g., [4, 15]), such methods blur the distinction between the MDP and the DFA, possibly yielding inefficiencies due to the agent's inability to take advantage of previously acquired knowledge about states differing only in the DFA component.

This work addresses this problem in a Multi-Agent System (MAS) setting. We extend MDPs to MAS using *Markov Games* consisting of: a set of agents $\Lambda = \{\lambda_1, \dots, \lambda_n\}$, one *action set* for each agent $A_1, \dots, A_n$, state transitions controlled by joint actions, i.e., $\delta : S \times A_1 \times \cdots \times A_n \to Pr(S)$, and agents' individual reward functions $R_i : S \times A_1 \times \cdots \times A_n \to \mathbb{R}$. We denote agent λ_i's joint action as $(a_i, \bar{a}_{-i})$, with a_i the action performed by λ_i and $\bar{a}_{-i}$ those of all other agents. Agents have different goals associated to different DFAs. $\mathscr{A}_i = (S, Q_i, q_{i,0}, \eta_i, F_i)$ denotes the DFA modeling λ_i's goal, with $q_{i,0} \in Q_i$, $\eta_i : Q_i \times S \to Q_i$, and $F_i \subseteq Q_i$. Every agent having its own goal, individual DFA's transitions depend only on the environment state and not on other DFAs.

3. NMRDPs Factorization

The dynamics of an NMRDP $\mathscr{N} = (\mathscr{M}, R_Q)$ is as follows: 1. on reset, the environment is in s_0 and $\mathscr{A}$ performs a dummy transition $q_0 \to q_0$ (consuming s_0); 2. at each step, on state (s,q) and action a, $\mathscr{M}$ progresses to s' with probability $\delta(s,a)$, $\mathscr{A}$ progresses to $q' = \eta(q,s')$ with probability 1, and the returned reward is $R_E(s,a,s') + R_Q(q,s',q')$. The resulting transition function is captured by the distribution $P(s',q'|s,q,a) = P(q'|s',q,s,a)\,P(s'|s,q,a)$, with $P(s',q'|s,q,a)$, $P(q'|s',q,s,a)$, $P(s'|s,q,a)$ the transition probability distributions of $\mathscr{N}$, $\mathscr{A}$, and $\mathscr{M}$, respectively. Since $\mathscr{M}$'s transitions are Markovian, $P(s'|s,q,a) = P(s'|s,a)$, i.e., given s and a, s' is independent of the history that led $\mathscr{A}$ to q. Also, q' depending only on q and s', $P(q'|s',q,s,a) = P(q'|s',q)$. Thus, $\mathscr{N}$'s transition model can be rewritten as $(\diamondsuit)P(s',q'|s,q,a) = P(s'|s,a)P(q'|s',q)$, and the reward function as $(\heartsuit)R(s,q,a,s',q') = R_E(s,a,s') + R_Q(q,s',q')$. Note that $R_Q(q,s',q')$ is independent of s,a, i.e., of how (s',q') is reached from previous state (s,q). Equations $(\diamondsuit)$ and $(\heartsuit)$ factorize the system dynamics decoupling the Markovian environment dynamics ($P(s'|s,a)$, $R_E(s,a,s')$) from the non-Markovian reward expressed by the DFA ($P(q'|s',q)$, $R_Q(q,s',q')$).

Discrete model-based RL can be applied in the joint state space $S \times Q$ to $P(s',q'|s,q,a)$ and $R(s,q,a,s',q')$. An algorithm for this is R-MAX [8, 14], which estimates the model by counting the number of visits $n(s,q,a)$ and setting a threshold δ to determine known transitions, thus enabling the estimation of the model functions. However, this solution does not scale with task complexity, as it does not exploit the Markovian nature of the environment.

Here, we focus on homogeneous collaborative MASs where the agents share the same environment transition and reward functions. Given the set of agents $\Lambda = \{\lambda_1, \dots, \lambda_n\}$, for every λ_i, λ_j, we have $P(s'|s,a_i,\bar{a}_{-i}) = P(s'|s,a_j,\bar{a}_{-j})$ and $R_E(s,a_i,\bar{a}_{-i},s') = R_E(s,a_j,\bar{a}_{-j},s')$. Agent-independence is denoted by $P(s'|s,a_*,\bar{a}_{-*})$ and $R_E(s,a_*,\bar{a}_{-*},s')$. On the other hand, agents have different tasks, modeled by individual DFAs $\mathscr{A}_i$. As a result, for every λ_i, the NMRDP can be factorized by $(\clubsuit)P(s',q_i'|s,q_i,a_i,\bar{a}_{-i}) = P(s'|s,a_*,\bar{a}_{-*})P(q_i'|s',q_i)$ and $(\spadesuit)R(s,q_i,a_i,\bar{a}_{-i},s',q_i') = R_E(s,a_*,\bar{a}_{-*},s') + R_Q(q_i,s',q_i')$.

The multi-agent state space S and the joint action space $A_1 \times \cdots \times A_n$ significantly impact performance and scalability. When combined with NMRDPs, state representation is further expanded with the states of the DFAs modeling the tasks. To efficiently tackle this problem, we consider independent DFAs allowing for extending the state of each agent λ_i with $S_i' = S \times Q_i$. In his way, the factorization defined by Equations $(\clubsuit)$ and $(\spadesuit)$ together with the independence $P(s'|s,a_*,\bar{a}_{-*})$ and $R_E(s,a_*,\bar{a}_{-*},s')$ provide for a significantly higher sample efficiency.

Using the above mentioned formulation, it is possible to devise algorithms for sample-efficient MARL in NMRDP. More specifically, the algorithm will implement and exploit the above mentioned techniques: 1) separate learning of the Markovian environment dynamics and the non-Markovian reward dynamics; 2) sharing of the environment dynamics among the agents during training.

In this short paper, we do not describe the details of any algorithm, since the main goal is to focus on the basic techniques described above. However, we provide in the next section some preliminary results when using a simple approach based on incremental individual training, in which each agent λ_i learns the environment dynamics $P(s'|s, a_*, \bar{a}_{-*})$ and $R_E(s, a_*, \bar{a}_{-*}, s')$ and the reward dynamics $P(q_i'|s', q_i)$ and $R_Q(q_i, s', q_i')$ to accomplish its task and shares the agent-independent learned environment dynamics with the other agents. The name MAQR-MAX is used in the next section to refer to such a solution.

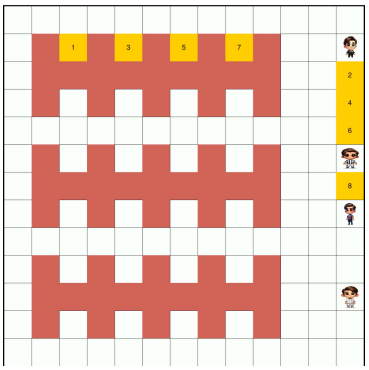
This approach improves sample efficiency, as the environment transitions and automaton transitions are learned separately and, once the environment model is stable, there is no need to keep learning them. When dealing with different (and even more complex) tasks in the same environment expressed by different automata, it is sufficient to update the non-Markovian model, leveraging the previously learned environment estimates. In this way, the agents easily scale to more complex tasks reusing the learning environment models.

4. Case Study: Multi-Agent Pickup and Delivery

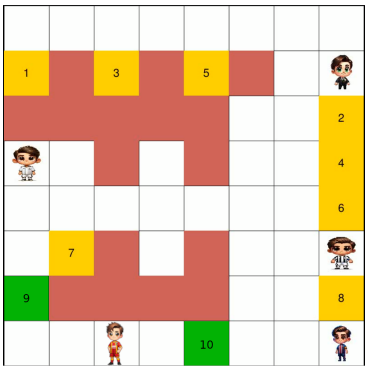
Multi-Agent Pickup and Delivery (MAPD) [7] is an online problem where a team of coordinated agents needs to fulfill a set of dynamically incoming pickup and delivery tasks. MAPD has several real-world applications, such as automated warehouse logistics [16], coordination of autonomous vehicles [17], automated control of non-player characters in video games [18]. Formally, MAPD involves k agents in an environment modeled as an undirected connected graph $G = (V, E)$ whose vertices V represent locations and edges E connections. In this paper, G is a 4-connected regular grid and time is discrete. An agent can either remain in its current location or move to any adjacent one. Actions last one timestep and, here, we assume can fail, leaving the agent in its current location. A *task set* $\mathcal{T}$ contains all unassigned tasks, and new ones can be dynamically added at any timestep. Each task $\tau_j \in \mathcal{T}$ comprises a pickup and a delivery location. When an agent is assigned a task, it plans a path on G to reach, from its current location, first the pickup and then the delivery location. Agents must not collide: distinct agents cannot be in the same location or traverse the same edge in opposite directions, at the same time. We consider a problem variant where $\mathcal{T}$ contains only k tasks, one per agent, all initially known. The aim of MAPD is planning paths to complete all the tasks in the shortest time. We evaluate solution quality as *makespan* (number of timesteps needed to complete all the tasks).

Here, we adopt an RL-based approach, modeling MAPD as a deterministic discrete NMRDP where each agent has its own non-Markovian reward, modeled as a RM, defining the agent's goals and their fulfillment order; these goals require an agent to reach first the pickup and then the delivery location. Each agent learns its own policy using MAQR-MAX.

In the NMRPD model, an agent's state includes its current location, the current timestep, and the progress, tracked by the RM, in fulfilling the assigned tasks. Such compound state is $(s, q) \in S \times Q$, with $s = (x, y, t)$ the agent's position on the grid x, y (a vertex in graph G) at timestep t and q the state of the agent's RM, tracking the progress towards its tasks. RM transitions between states depend on the occurrence of *pickup*, *delivery*, and *collision* events. A pickup event occurs (instantaneously) when the agent reaches the pickup location; delivery events occur similarly, but only if the pickup event has occured already. Collision events occur if two agents are in a same position x, y at same timestep t. The RM rewards the transitions firing on pickup and delivery events, while penalizes those occurring on collisions. More precisely, the RM capturing an agents' non-Markovian goal is s.t., on event e, a transition $(q, e) \rightarrow (q', r_Q)$ takes place from q to q' returning a reward value r_Q.



(a) MAPD 13 × 13.



(b) MAPD 8 × 8.

Task priority	PG	8 x 8 Pickup & Delivery	13 x 13 Pickup & Delivery
Priority 1:		Cell: 5 - 6	Cell: 7 - 8
Priority 2:		Cell: 3 - 4	Cell: 5 - 6
Priority 3:		Cell: 1 - 2	Cell: 3 - 4
Priority 4:		Cell: 7 - 8	Cell: 1 - 2
Priority 5:		Cell: 9 - 10	None

(c) Priority legend.

Figure 1: MAPD for 13×13, 8×8 and legend.

4.1. Experimental Setup

The MAPD environments we use consist of grids with walls and narrow corridors, which challenge the agents' coordination ability. As discussed, the agent's observation space includes its current position, the timestep, and RM state. The action space comprises the four cardinal directions, thus having size 4 (note that we do not consider wait actions). Pickup and delivery actions are implicitly assumed to be automatically and instantaneously executed when the pickup and delivery locations are achieved. Executing actions that would make an agent hit an obstacle or a boundary yields no effect. In order to penalize adjacent agents when swapping positions simultaneously, each agent's trajectory is augmented with an additional timestep for every recorded position (details omitted for space reasons).

Agents are trained incrementally, one after another, taking into account, in the reward signal, the potential occurrence of collisions stemming from the learnt policies of agents trained at previous iterations. Agents are assigned a priority corresponding to the training order: an agent receives a penalty whenever, at training time, occupies the same position as that another agent with higher priority would occupy, at the same timestep, when executing its learned policy.

We have used two distinct environments, an 8 × 8 and a 13 × 13 map (see Figs. 1a and 1b), and tasks, modeled as RMs and assigned before training. In both environments, priorities decrease from 1 to 5. Agents are assigned priorities and tasks as illustrated in Fig. 1c, where the first and second cells reported in column "Pickup & Delivery" are the pickup and delivery locations, respectively. Observe how this approach allows for flexibly defining complex MAPD tasks. For instance, one could conveniently define a problem where the agents must perform multiple pickups and deliveries under order constraints on their visits.

4.2. Experimental Results

In this section, we report the results of some preliminary experiments aiming at discussing potential benefits of algorithms exploiting Model-Based Multi-Agent RL techniques described before.

We carried out tests to evaluate sample efficiency and environment exploration rate, using a CPU i7-11800H 2.30GHz and 16 GB of RAM. Actions have 0.6 success (move to intended cell) and 0.4 failure (remain still) probability. The threshold for a pair (s, a) to be *known* is 100. Every 50 episodes, the current optimal policy is evaluated over 100 executions.

Fig. 2 shows the performance of MAQR-MAX on the 8×8 MAPD domain (Fig. 1b) with 4 agents, in the basic scenario where agents share no information (transition probabilities) about the environment. Fig. 3 illustrates the results for the same scenario, when agents benefit from the transition probability function learned by previous agents. A significant improvement in sample efficiency is obtained, in particular, by Agents 3 and 4, which show a clear decrease in convergence time and an increase in solution quality (timesteps taken to complete the task).

Fig. 4 shows results for the same experiment, now with Agent 1 inheriting Agent 4's learned transitions

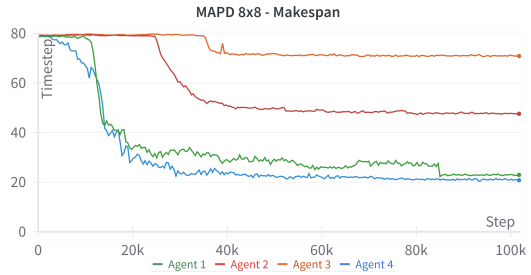


Figure 2: MAPD 8×8 without Transfer Learning.

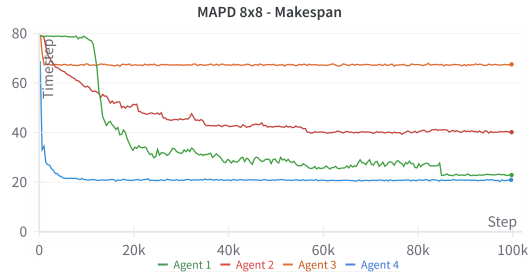
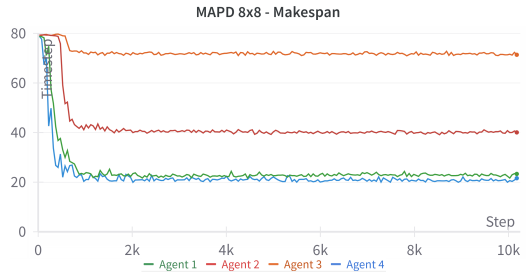


Figure 3: MAPD 8×8 with Transfer Learning.



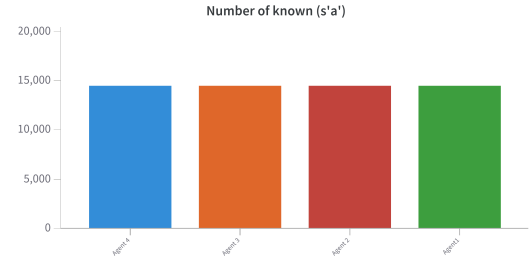


Figure 4: Agent 1 starts with 70% known transitions.

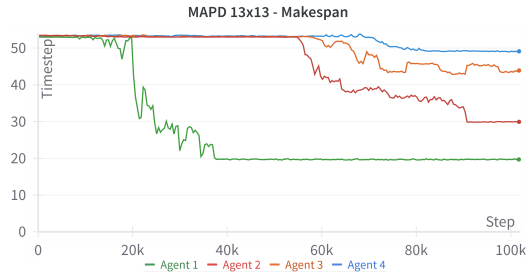


Figure 5: MAPD 13×13 without Transfer Learning.

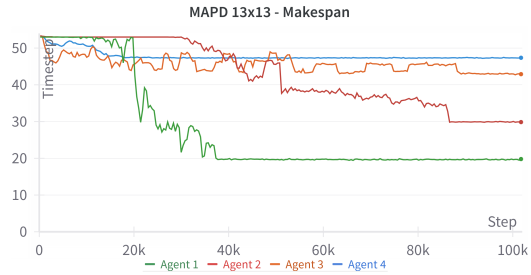


Figure 6: MAPD 13×13 with Transfer Learning.

(70.33%). Agents 1 and 2 converge more rapidly, demonstrating scalability. The increasing fraction of *known* transitions after each training phase further underscores the importance of knowledge transfer.

We have also tested the approach on a 13×13 scenario (Figs. 5 and 6), to preliminarily test scalability. A behavior similar to that of the 8×8 scenario is observed, with the agents achieving a smaller makespan and requiring fewer episodes to learn the optimal policy than in the case without transfer learning. These results demonstrate the feasibility of our approach. Future refinements may further improve performance and scalability.

5. Conclusion

We have presented a formalization of Multi-Agent Model-Based Reinforcement Learning for NMRDPs that, by decoupling the Markovian transitions in the environment from the non-Markovian evolution of the reward and by sharing the learned environment model among the agents during training, allows the definition of sample-efficient RL algorithms for solving complex multi-agent problems. The general applicability of the proposed solution has been demonstrated by successfully addressing the Multi-Agent Pickup and Delivery problem. A more detailed investigation of algorithms exploiting these properties is left as future work.

Acknowledgements

Work partly supported by the PNRR MUR project PE0000013-FAIR and the Sapienza project MARLeN (Multi-layer Abstraction for Reinforcement Learning with Non-Markovian Rewards).

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] F. Bacchus, C. Boutilier, A. J. Grove, Rewarding behaviors, in: Proc. AAAI, 1996, pp. 1160–1167.
[2] S. Thiébaux, C. Gretton, J. K. Slaney, D. Price, F. Kabanza, Decision-theoretic planning with non-markovian rewards, J. Artif. Intell. Res. 25 (2006) 17–74.
[3] R. Brafman, G. De Giacomo, F. Patrizi, LTLf/LDLf non-Markovian rewards, in: Proc. AAAI, 2018, pp. 1771–1778.
[4] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, Reward machines: Exploiting reward function structure in reinforcement learning, J. Artif. Intell. Res. 73 (2022) 173–208.
[5] G. D. Giacomo, L. Iocchi, M. Favorito, F. Patrizi, Reinforcement learning for LTLf/LDLf goals, CoRR (2018). URL: http://arxiv.org/abs/1807.06333.
[6] S. Albrecht, F. Christianos, L. Schäfer, Multi-Agent Reinforcement Learning: Foundations and Modern Approaches, MIT Press, 2024.
[7] H. Ma, J. Li, T. Kumar, S. Koenig, Lifelong multi-agent path finding for online pickup and delivery tasks, in: Proc. AAMAS, 2017, pp. 837–845.
[8] M. Gaon, R. I. Brafman, Reinforcement learning with non-Markovian rewards, in: Proc. AAAI, 2020, pp. 3980–3987.
[9] G. Gupta, C. Yin, J. V. Deshmukh, P. Bogdan, Non-Markovian reinforcement learning using fractional dynamics, in: Proc. CDC, 2021, pp. 1542–1547.
[10] P. Bacon, J. Harb, D. Precup, The option-critic architecture, CoRR (2016). URL: http://arxiv.org/abs/1609.05140.
[11] R. S. Sutton, D. Precup, S. Singh, Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning, J. Artif. Intell. 112 (1999) 181–211.
[12] G. D. Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: Proc, IJCAI, 2013, pp. 854–860.
[13] C. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (1992) 279–292.
[14] R. I. Brafman, M. Tennenholtz, R-max - a general polynomial time algorithm for near-optimal reinforcement learning, J. Mach. Learn. Res. 3 (2003) 213–231.
[15] G. De Giacomo, L. Iocchi, M. Favorito, F. Patrizi, Restraining bolts for reinforcement learning agents, in: Proc, ICAPS, volume 34, 2020, pp. 13659–13662.
[16] P. R. Wurman, R. D'Andrea, M. Mountz, Coordinating hundreds of cooperative, autonomous vehicles in warehouses, AI Mag. 29 (2008) 9–20.
[17] M. Veloso, J. Biswas, B. Coltin, S. Rosenthal, Cobots: Robust symbiotic autonomous mobile service robots, in: Proc. IJCAI, 2015, pp. 4423–4429.
[18] H. Ma, J. Yang, L. Cohen, T. S. Kumar, S. Koenig, Feasibility study: Moving non-homogeneous teams in congested video game environments, in: Proc. AIIDE, 2017, pp. 270–272.