Extending Decision-Making Policies in Partially Observable Environments for Active Perception

Tarik Selimović[1], Marijana Peti[1], Frano Petric[1] and Stjepan Bogdan[1]

Abstract— **This paper presents a method for extending decision-making policies in active perception tasks for multi-agent systems within partially observable environments. Multiple agents obtain their policies by training in an environment of a certain size. Those policies are then used in the environments larger in size, that are divided into sub-environments of size similar (or smaller) to one that the agents were trained in. Learned policies are adapted accordingly by proposed Action-Space Reduced Policy (ASRP). By leveraging Multi-Agent Reinforcement Learning (MARL) within a POMDP framework, agents can use their learned policies across environments of differing complexity without requiring retraining. The consensus mechanism allows agents to maintain a common belief state, supporting collaborative decision-making based on observations from all agents involved. Validation of the method is conducted on the scenario of multi-agent exploration missions, demonstrating the use of extended policies and enhanced perception accuracy. Simulation results indicate expected success rates and decision-making times, regardless of the environment's dimensionality. Potential applications for scalable, multi-agent perception systems are discussed, along with directions for future research.**

I. INTRODUCTION

In recent years, significant progress has been made in the use of curiosity-driven, open-ended exploration in multi-agent systems. [1] describes a method that provides a policy and integrates Multi-Agent Reinforcement Learning (MARL) into the framework of a Partially Observable Markov Decision Process (POMDP), supported by consensus protocol for information exchange. The method was proposed to address the challenges of active perception in decentralized environments. The fundamental limitation of the proposed approach is the fact that the dimension of the environment and possible actions is encoded in the number of inputs and outputs of policy, which prevents an easy application of policy to different environments. Since the approach is based on reinforcement learning, one of the ways would be to implement a form of transfer learning as described in [2], [3], which requires more training or fine-tuning. The next approach for generalization can be meta-reinforcement learning [4], which is common when the goal is to learn a policy that is able to adapt to each new task from the task distribution with as little data as possible. Techniques such as Model-Agnostic Meta-Learning (MAML) [5] and RL^2 [6] enable agents to acquire adaptable representations that generalize across multiple environments.

[1]Authors are with University of Zagreb Faculty of Electrical Engineering and Computing, Laboratory for Robotics and Intelligent Control Systems, LARICS, Unska 3, 10000 Zagreb, Croatia; (tarik.selimovic, marijana.peti, frano.petric, stjepan.bogdan)@fer.unizg.hr

Additionally, procedural generation techniques, such as those in ProcGen Benchmark [7], have been introduced to explicitly evaluate RL generalization by exposing agents to an ever-changing set of tasks. Furthermore, unsupervised auxiliary objectives, such as predictive contrastive coding [8], have been used to learn representations that generalize beyond specific training environments.

Compared to the generalization approaches mentioned above, the approach described in this paper, which uses the framework (Co-RL-POMDP) that we presented in [1], is based on the idea of applying the policies trained in one environment to a comparable environment with an extended number of actions and a larger group of agents, without the need for additional training or fine-tuning of the policies.

The main contributions of the paper are:

- Method to extend the decision-making policies to make them applicable to the similar environments encompassing extended number of agents (extended Co-RL-POMDP),
- Application of the policy to the Environment Exploration problem with comparison to classical Co-RL-POMDP.

The remainder of the paper is organized as follows. Section II introduces the problem of active perception in multi-agent systems based on independent learners and consensus. Section III describes the extension of decision-making policies in partially observable environments with mathematical description. Section IV presents a case study in which the described approach is applied, with a discussion of the results obtained. The paper is concluded in Section V.

II. MULTI-AGENT ACTIVE PERCEPTION

Multi-agent active perception involves gathering information in an environment to systematically reduce uncertainty and enhance the agents' understanding of its state.

Consider a scenario in which r agents are tasked with exploring possible insight about the environment by collecting relevant observations and constructing a belief vector $\mathbf{b}$ through a predefined belief model. The primary objective of these agents is to efficiently determine the most appropriate value of the belief vector in the shortest possible time, thereby gaining meaningful information about the state of the environment. The method proposed here is based on our work presented in [1]. In the rest of this section we give a brief description of major elements of the method, graphically depicted in Fig. 1.

While the agents have the capability to exchange information during this process, it is important to note that the

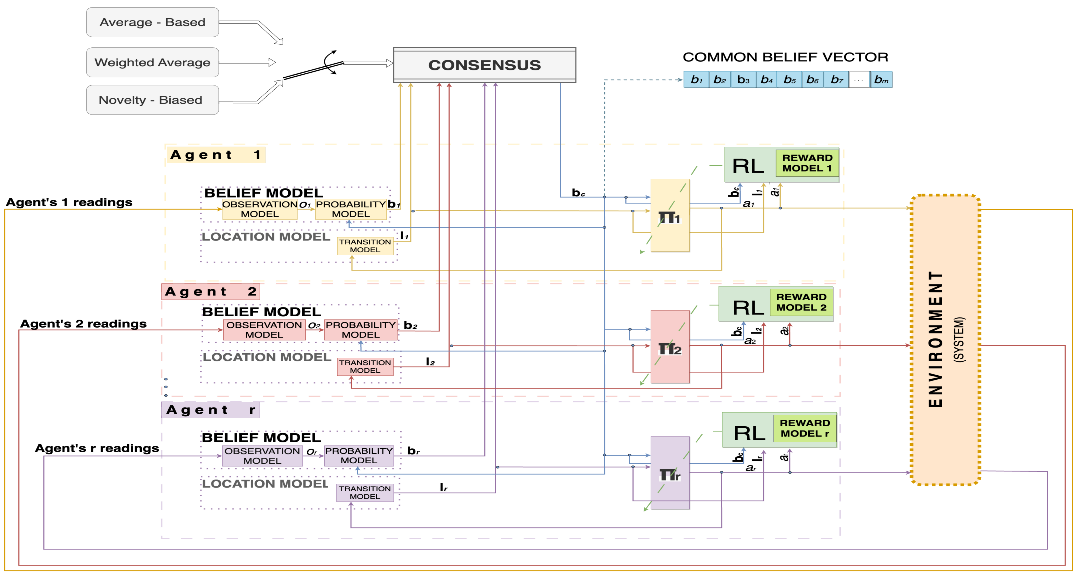


Fig. 1. Closed-loop control diagram of the multi-agent active perception [1]

stability and convergence of the proposed method do not rely on the success of such communication. This ensures that even in scenarios with limited or unreliable communication, the approach remains effective in guiding the agents toward an accurate representation of the environment.

A. Partially Observable Markov Decision Process

The interaction between the agent and the system is formally described using a *Partially Observable Markov Decision Process (POMDP)* [9]. In a POMDP, the agent operates in an environment where it does not have full information about the system's internal state, but instead must rely on observations that provide partial and often noisy information about the true state. This leads to a belief-based decision-making process, where the agent maintains a belief distribution over possible states rather than a precise state estimate. POMDP is described by:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, \gamma) \tag{1}$$

where: - $\mathcal{S}$ is the state space, - $\mathcal{A}$ is the action space, - $\mathcal{O}$ is the observation space, - T is the state transition function $T(s', a, s)$, - R is the reward function $R(s, a)$, - γ is the discount factor.

The state transitions in the system are modeled by a *transition model* $T(s', a, s)$, which defines the probability of transitioning to a new state s' when an action a is taken in the current state s. This transition model captures the dynamics of the environment, allowing the agent to predict the likelihood of different future states given its actions. The transition model can be applied in relative and absolute ways. The *relative transition* describes state transitions as a function of the current state, action, and new state, typically used in scenarios like robot navigation, where the agent's movement depends on its current position and action. Mathematically, it is represented as $T(s', a, s) = P(s'|s, a)$, where P is the probability that the new state s' will be reached from the current state s upon action a For example, in a grid-based world, if the agent is at position (1,1) and takes the action "move up," it transitions to position (0,1). In contrast, the *absolute transition model* describes state transitions as a function of the action and new state, independent of the previous state. This model is used when the agent's action leads to a predefined target state, such as moving to specific fields in a grid. It is represented as $T(s', a) = P(s'|a)$, where P is the probability to reach new state s' upon action a. For example, actions could be "go to position (1,1)" or "go to position (0,1)", where each action probabilistically moves the agent to a specific field regardless of its previous position. In POMDP framework , rewards are determined by a *reward model* $R(s, a)$, which specifies the immediate reward an agent receives when it performs action a in state s. These rewards guide the agent's behavior, motivating it to choose actions that lead to higher rewards over time. In addition to the transition and reward models, the agent receives *observations* that provide partial information about the current state. These observations, represented as o, are related to the state through an *observation model* $O(o, s, a)$, which defines the probability of receiving observation o when the agent takes action a in state s. Since the agent's knowledge of the true state is limited, it must rely on these observations to update its belief about the environment, which in turn influences its future actions and decisions. The goal of the agent in a POMDP is to determine a policy that maps its belief state (a distribution over possible states) to actions, aiming to maximize cumulative rewards despite the uncertainty in observations and state transitions.

B. Multi-Agent Reinforcement Learning

In multi-agent systems, policy learning is achieved through Multi-Agent Reinforcement Learning (MARL) algorithms [10], which can be divided into three main categories:

1) Centralized learning: a single policy is trained to control all agents [11]. A centralized solver processes the observations of all agents and determines their actions. However, as the number of agents increases, the number of possible joint actions grows exponentially, making this approach a challenge.

2) Centralized training, decentralized execution (CTDE): Policies are learned centrally but executed independently by each agent [12]. During execution, the agents make their decisions independently and without direct coordination.

3) Independent Learners (IL): In this approach [13], [14], which is used here, each agent learns its own policy and optimizes the rewards independently. This allows agents to make decisions autonomously without relying on a central controller. In IL, each agent creates its own policy to be used in the decision-making process. Although this approach avoids scalability problems, it also introduces some new challenges, such as the non-stationarity of the environment and the lack of explicit coordination. Different approaches to overcome these challenges are described in [15]. Examples from this group of independent learners are IQL, IA2C, IPPO and COMM-IQL described in [16], [17], [18], [19] respectively. They differ in the use of reinforcement learning techniques. IQL and COMM-IQL (a variant of IQL that includes communication) are based on Q-learning. IA2C is a variant of the widely used A2C algorithm, which is based on the Actor-Critic RL algorithm, and IPPO is based on the Proximal Policy Optimization (PPO) RL algorithm. Independent learner policy of agent i, trained in the learning process, is denoted as:

$$\pi_i : s_i \to a_{ij}, \tag{2}$$

where s_i is the state of agent i, π_i represents the policy of agent i and a_{ij} is the j-th action of the i-th agent.

C. Consensus

The fundamental challenge of non-stationarity in independent learners has been addressed through consensus [20], [21]. The consensus protocol allows agents to make their decisions taking into account the observations collected by all agents. By exchanging information during the process, agents can collaborate and reach a shared understanding, represented as a common belief vector $\mathbf{b}_c$, which unequally represents the state of the environment.

In general, the consensus is written as

$$\mathbf{b}_i(k+1) = \sum_{j \in \mathcal{N}_i} \alpha_{ij}(k)[\mathbf{b}_j(k) - \mathbf{b}_i(k)], \tag{3}$$

where α_{ij} represents level of relation between agents i and j, and $\mathcal{N}_i$ defines a set of agents neighboring with agent i. Consensus that we proposed here is Novelty-Biased consensus (NB), introduced in [22].

III. Extending Decision-Making policy

The generated policies are derived from a specific POMDP and cannot be applied to an environment with a different number of states. The POMDP model defines state transitions, the probability of specific observations, and a set of possible actions. Therefore, any change in the number of states or actions renders the existing policies invalid, as it alters the entire POMDP structure. Consequently, applying the algorithm to a new environment requires generating new policies tailored to the updated rules and rewards. This process requires substantial time for training and fine-tuning within the reinforcement learning framework. Our aim is to avoid new training once agents face similar environment, from the same domain of active perception.

The idea that we explore here is formulated on the hypothesis that in the process of active perception (agents do not change environment, just perceive it), once learned policies that contain the desired behavior of the decision maker based on beliefs about the environment, would work regardless of the number of states or actions modeled within the POMDP framework as long as this model can be extended in particular way. The basic requirement for applying the proposed mathematical approach for extending decision-making policies is that the policy is obtained on a POMDP with an absolute transition model. The application of the same method to a POMDP with a relative transition model will be the subject of further research.

A. Sub-Environments

To deploy policies trained in an environment with a specific dimensionality to an environment with a different dimensionality, we first define the *Policy Scope Environment (PSE)*, which refers to the dimensions of the state and action spaces that the policies were trained on.

The environment is modeled as a POMDP, with the policy π initially trained in an environment with state space $\mathcal{S}_{\text{tr}}$ and action space $\mathcal{A}_{\text{tr}}$, where the dimensionalities of the state and action spaces are given by:

$$\dim(\mathcal{S}_{\text{tr}}) = \Delta_{\mathcal{S}}, \quad \dim(\mathcal{A}_{\text{tr}}) = \Delta_{\mathcal{A}}, \tag{4}$$

where $\Delta_{\mathcal{S}}$ and $\Delta_{\mathcal{A}}$ represent the dimensions of PSE for the state and action spaces, respectively. When deploying the policy to an environment with a different dimensionality, we consider two cases:

1) Environment with Dimensionality Greater Than PSE: If the environment has a state space whose dimensionality is greater than the dimensions of PSE, the environment is partitioned into multiple sub-environments. The number of sub-environments η, is determined by

$$\eta = \left\lceil \frac{\dim(\mathcal{S}_{\text{env}})}{\Delta_{\mathcal{S}}} \right\rceil \tag{5}$$

where $\dim(\mathcal{S}_{\text{env}})$ is the dimension of the state space of the environment and $\Delta_{\mathcal{S}}$ is the state space dimension of the PSE. The environment is then divided into η smaller sub-environments, each corresponding to a dimension of the state and action space corresponding to the PSE. If the division of the state space does not result in a perfect fit (i.e., when $\dim(\mathcal{S}_{\text{env}}) \mod \Delta_{\mathcal{S}} \neq 0$), the last sub-environment will have a smaller dimension than state space dimension of PSE.

2) (Sub-)Environment with Dimensionality Less Than PSE: If the environment has a state and action space whose dimensionality is smaller than the dimensions of the PSE, we execute the Action-Space Reduced Policy (ASRP). In this case, only the portion of the action space that corresponds to the available actions is implemented in the smaller environment. The policy is restricted to the first e outputs of its decision function, where e is the number of actions in the reduced environment:

$$e = \dim(\mathcal{A}_{\text{env}}), \quad \dim(\mathcal{A}_{\text{env}}) < \Delta_{\mathcal{A}} \tag{6}$$

The action a_{ij} is then selected as:

$$a_{ij} = \pi_i(\bar{\mathbf{s}}_i) = \arg \max_{1 \le j \le e} (\mathbf{q}_i) \tag{7}$$

where $\mathbf{q}_i$ represents the action-value function (Q-values) over the subset of available actions, ensuring that the policy can be effectively applied in environments with smaller action and state spaces, without retraining or modifications.

B. Agent distribution in sub-environments

Uniform (even) distribution is a method in which the agents are distributed as evenly as possible across the sub-environments. This approach ensures that each sub-environment receives at least one agent before further agents are assigned cyclically. The aim is to maintain a balanced distribution and prevent a particular sub-environment from being overloaded by agents. This distribution method is only applicable if the number of agents r is greater than the number of sub-environments η, i.e. $r > \eta$.

The distribution method follows these steps:

1) *Initial Allocation* – Assign one agent to each sub-environment to ensure all are occupied.
2) *Sequential Assignment* – Distribute the remaining $(r-\eta)$ agents one by one in a cyclic (round-robin) manner across the sub-environments until all agents are assigned.

The number of agents assigned to sub-environment $w, (w > 0)$ is given by:

$$N_w^A = \left\lfloor \frac{r}{\eta} \right\rfloor + \begin{cases} 1, & \text{if } w \le (r \mod \eta) \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

After dividing the environment into sub-environments and distributing the agents responsible for active perception, we introduce the agent distribution matrix $\mathbf{D}$, where the agents are represented by the rows and the sub-environments by the columns, reflecting the distribution of agents across the sub-environments:

$$\mathbf{D} = [d_{i,w}], \quad i = 1, \dots, r; \; w = 1, \dots, \eta \tag{9}$$

where:

- $d_{i,w} = 1$ if agent i is assigned to sub-environment w,
- $d_{i,w} = 0$ otherwise.

For every agent i, there is exactly one '1' in the column corresponding to the agent, which ensures that each agent is assigned to exactly one sub-environment. Using the distribution matrix, the number of agents assigned to sub-environment w, denoted N_w^A, is calculated as:

$$N_w^A = \sum_{i=1}^{n} d_{i,w} \quad \forall w = 1, 2, \dots, \eta \tag{10}$$

C. Completing the Active Perception Mission

The mission in a sub-environment is completed when the agents in a sub-environment sufficiently reduce the uncertainty about states. Upon reaching the uncertainty threshold, the agent initiates action *end* (in the proposed approach, regardless of the scope or nature of the problem, there is always an *end* action that completes the agent's task). Since only agents in sub-environment exchange information on belief vector, agents in other sub-environments are not aware that uncertainty threshold is reached and may continue to act according to the policy, although the mission goal has been achieved. To mitigate such a situation, the agent policy takes into account the values of the distribution matrix.

Each time agent i executes action *end* in sub-environment w, the value of corresponding element in the distribution matrix $\mathbf{D}$ changes:

$$d_{i,w} = \begin{cases} 1, & \text{if } a_{ij} \neq \text{end} \\ 0, & \text{if } a_{ij} = \text{end} \end{cases} \tag{11}$$

i.e. N_w^A is decreased by 1.

To ensure a coordinated completion of the mission and thus finish the active perception task, the values of the distribution matrix are communicated between agents in sub-environments and taken into account, in addition to the policy,

$$a_{ij} = \begin{cases} end, & \text{if } \exists w \text{ s.t. } N_w^A < l, w = 1, ..., \eta \\ \pi_i(\bar{\mathbf{s}}_i), & \text{otherwise} \end{cases} \tag{12}$$

where l is the value that specifies the number of agents in a sub-environment that must complete the task before all other agents in other sub-environments perform the action *end*. For example, in the case $l = 2$, with 2 sub-environments and 3 agents initially assigned to each sub-environment (overall 6 agents), the active perception mission will be completed when at least two agents, in either sub-environment, initiate *end* action.

IV. CASE STUDY

The case study we present to demonstrate Extending Decision-Making Policies for Active Perception is a situation in which a team of robots is searching for a fire in a building. In the projected scenario, the architecture of the building is known in advance, with only one location on fire. In each iteration, agents explore specific locations, collect observations (which may be imperfect), and update a common belief vector through consensus. The consensus mechanism gives greater influence to more reliable agents and ensures that the common belief vector reflects the most trustworthy information. The mission is completed when fire location is determined with predefined probability. For a detailed explanation, the definition of the trustworthiness of agents and the mathematics behind it, see [22].

To evaluate the generalization and scalability of the learned policies, we conduct experiments using a Multi-Agent Reinforcement Learning (MARL) framework with independent learners. The simulations are done with different number of locations and agents, while the perception properties of agents, transition and reward model are same as in [1] and [22]. Introduced PSE refers to policies learned in an environment with 10 locations and 3 agents. These policies are then applied using the proposed approach for extending decision-making policies to larger environments with 18, 20, and 25 locations, involving 5, 6, and 8 agents, respectively. Two of those examples (18Lx5A and 25Lx8A) implement ASRP in sub-environments.

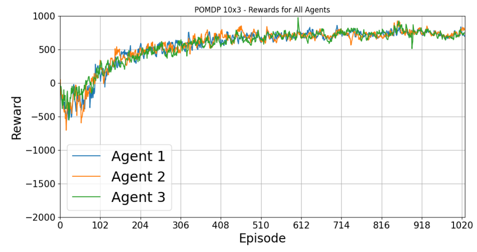
To compare the efficiency of the proposed approach, we trained policies (using the method in [1]) for environments with 18, 20 and 25 locations involving 5, 6 and 8 agents, respectively. Policies for extended environments are trained within a similar time frame as the policies for PSE (3 agents and 10 locations). As expected, due to the significantly higher dimensionality of the problem, a smaller number of episodes was achieved within the training time frame (Fig. 2). The simulations and training for all cases were performed on a MacBook Pro M1 under macOS Monterey and included the Python programming language with TensorFlow and Keras for the machine learning components and POMDPy [23] for modeling the Partially Observable Markov Decision Process.

A. *Results*

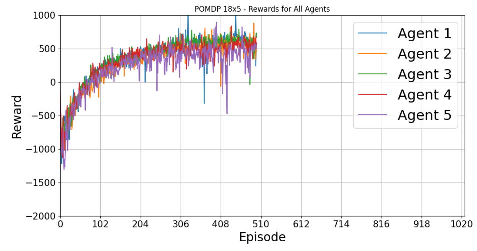
The validation of this approach is done by comparing the number of iterations or average value of actions required for each agent to reach its final belief and the success rate with trained policy for a larger environment with more agents. The success rate and the number of iterations required to complete tasks is evaluated across 100 executions of the mission scenario in each of different environments with a different number of locations and agents. The average values with standard deviations are shown in Table I. For all investigated scenarios the proposed extension policy method outperforms dedicated policies in success rate and the number of iterations. For example, the policy trained for 25 locations with 8 agents (25L x 8A) achieves 76% success rate and completes, in average, after 10.18 iterations (with very large standard deviation). In the same time policy trained on 10 locations with 3 agents, extended to 25Lx8A scenario, achieves 88% success rate and completes in average after 5.94 iterations with significantly lower standard deviation.

For comparison, the table also shows the results obtained by a greedy method (BM) with the agent always selecting the action with the highest reward based on its location and a belief vector. While BM shows slightly better performance in success rate than the proposed approach, the average number of iterations required to complete a task is higher. Moreover, the time complexity of the BM execution, which is $O(n)$, is significantly higher than that of the proposed approach, which is $O(1)$.

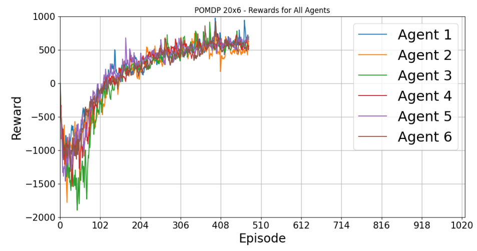
A graphical representation of the success rate, provided in (Fig. 3), clearly illustrates the decrease in the success rate of agents as the complexity of the environment and the number of agents increases, while at the same time the extended decision-making policy maintains a consistent success rate regardless of the increasing complexity of the scenario.



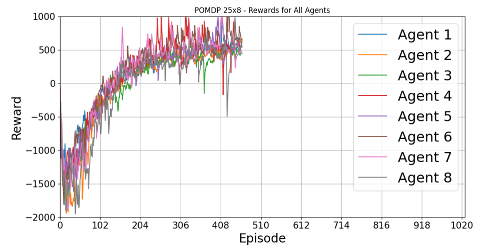
(a) Environment with 10 Locations and 3 Agents



(b) Environment with 18 Locations and 5 Agents



(c) Environment with 20 Locations and 6 Agents



(d) Environment with 25 Locations and 8 Agents

Fig. 2. Reward value throughout policy training across different environments within same timeframe

V. CONCLUSION

The proposed method builds on the work of [1] and extends the decision-making policies for active perception tasks in multi-agent systems operating in partially observable environments. The problem is formulated as a Partially Observable Markov Decision Process (POMDP) with an absolute transition model. This research presents a pipeline for adapting independent learner policies to larger environments within the same domain without requiring additional training.

A key novelty of the approach is the transformation of the environment into sub-environments and the distribution of agents across them. It also introduces a modification of

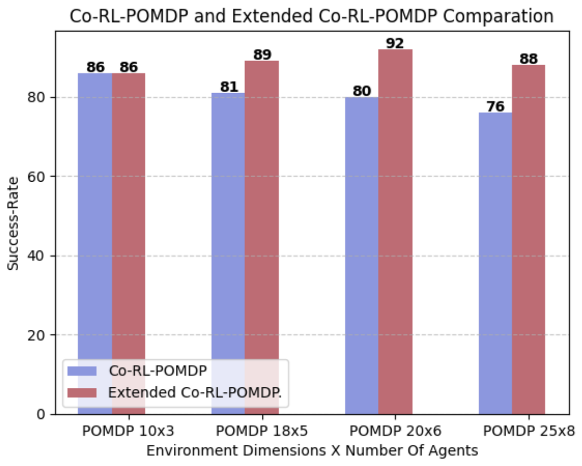


Fig. 3. Graphical comparison of CO-RL-POMDP and Extended CO-RL-POMDP in terms of success rate.

TABLE I

Performance of the Extended CO-RL-POMDP Algorithm ($l = 2$), Across Various Environment Dimensions and Agent Numbers, Compared to CO-RL-POMDP and BM decision-making policy

No. Of Loc.	No of Agents	Policy	Success Rate [%]	Number of Iteration $[iter.]$
10	3	$10L \times 3A$	86 %	6.62 ± 2.93
		BM	90 %	7.08 ± 2.74
18	5	$18L \times 5A$	81 %	8.26 ± 4.41
		$10L \times 3A$	89 %	6.38 ± 2.19
		BM	94 %	7.16 ± 1.89
20	6	$20L \times 6A$	80 %	9.54 ± 7.69
		$10L \times 3A$	92 %	6.49 ± 2.09
		BM	94 %	7.86 ± 4.45
25	8	$25L \times 8A$	76 %	10.18 ± 8.03
		$10L \times 3A$	88 %	5.94 ± 1.03
		BM	90 %	8.54 ± 4.39

agent actions based on the number of agents within each sub-environment.

The proposed method is evaluated by comparing it with a newly generated policy, obtained by applying the algorithm [1] to a larger environment, and with a greedy method applied to the same environment. In contrast to the proposed approach, generating a new policy requires re-applying the algorithm to take into account the updated dimension of rules and rewards of the new environment. The results obtained in this paper are promising for the future research, which will investigate how to enable the agents to work in dynamic environments, to switch between sub-environments and quickly complete the task of active perception.

ACKNOWLEDGMENT

This work has been supported in part by the scientific project Strengthening Research and Innovation Excellence in Autonomous Aerial Systems - AeroSTREAM supported by European Commission HORIZONWIDERA-2021-ACCESS-05 Programme through project under G. A. number 101071270. The work of doctoral student Marijana Peti has been supported in part by the "Young researchers' career development project–training of doctoral students" of the Croatian Science Foundation.

REFERENCES

[1] T. Selimović, M. Peti, and S. Bogdan, "Multi-agent active perception based on reinforcement learning and pomdp," *IEEE Access*, vol. 12, pp. 48004–48016, 2024.

[2] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," 2020.

[3] A. A. Taiga, R. Agarwal, J. Farebrother, A. Courville, and M. G. Bellemare, "Investigating multi-task pretraining and generalization in reinforcement learning," in *The Eleventh International Conference on Learning Representations*, 2023.

[4] J. Beck, R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, and S. Whiteson, "A survey of meta-reinforcement learning," 2023.

[5] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," 2017.

[6] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl^2: Fast reinforcement learning via slow reinforcement learning," 2016.

[7] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," 2020.

[8] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2019.

[9] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee, "Pomdps for robotic tasks with mixed observability," in *Robotics: Science and Systems*, 2009.

[10] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spanò, "Multi-agent reinforcement learning: A review of challenges and applications," *Applied Sciences*, vol. 11, no. 11, 2021.

[11] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," 07 1998.

[12] Y. Zhou, S. Liu, Y. Qing, K. Chen, T. Zheng, Y. Huang, J. Song, and M. Song, "Is centralized training with decentralized execution framework centralized enough for marl?," 2023.

[13] J. Hu and M. P. Wellman, "Multiagent reinforcement learning in stochastic games," in *International Conference on Machine Learning*, 1999.

[14] D. Schwung, S. Yuwono, A. Schwung, and S. X. Ding, "Decentralized learning of energy optimal production policies using plc-informed reinforcement learning," *Computers & Chemical Engineering*, vol. 152, p. 107382, 2021.

[15] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," 2019.

[16] M. Tan, "Multi-agent reinforcement learning: Independent versus cooperative agents," in *International Conference on Machine Learning*, 1997.

[17] K. He, P. Doshi, and B. Banerjee, "Latent interactive a2c for improved rl in open many-agent systems," 2023.

[18] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. S. Torr, M. Sun, and S. Whiteson, "Is independent learning all you need in the starcraft multi-agent challenge?," 2020.

[19] R. Pina, V. D. Silva, C. Artaud, and X. Liu, "Fully independent communication in multi-agent reinforcement learning," 2024.

[20] T. Li, L. W. Krakow, and S. Gopalswamy, "Optimizing consensus-based multi-target tracking with multiagent rollout control policies," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, aug 2021.

[21] X. Xu, R. Li, Z. Zhao, and H. Zhang, "Communication-efficient consensus mechanism for federated reinforcement learning," 2022.

[22] F. Petric, M. Peti, and S. Bogdan, "Multi-agent coordination based on pomdps and consensus for active perception," in *Intelligent Autonomous Systems 17*, pp. 690–705, Springer Nature Switzerland, 2023.

[23] K. Zheng and S. Tellex, "pomdp_py: A framework to build and solve pomdp problems," in *ICAPS 2020 Workshop on Planning and Robotics (PlanRob)*, 2020. Arxiv link: "`https://arxiv.org/pdf/2004.10099.pdf`".