Cooperative Navigation via Relational Graphs and State Abstraction

Salwa Mostafa, Mohamed K. Abdel-Aziz, Mehdi Bennis

***Abstract*—We consider a cooperative-navigation problem in a partially observable MADRL framework. We investigate how agents cooperate to learn a communication protocol given a very large state space while generalizing to a new environment. The proposed solution leverages the notion of structured observation and abstraction, in which the raw-pixel observations are converted into a relational graph that is then used for learning abstraction. Abstraction is performed based on compression using a relational graph autoencoder (RGAE) and a multilayer perceptron (MLP) to remove irrelevant information. The results show the effectiveness of the proposed MLP and RGAE in learning better policies with better generalization capabilities. It is also shown that communication among agents is instrumental in improving the navigation task performance.**

***Index Terms*—Reinforcement learning, robot navigation, state abstraction, relational graph, emergent communication.**

I. Introduction

Navigation is a fundamental problem in mobile robots, where the aim is to find the optimal path from a starting point to a target destination. The traditional framework for solving the navigation problem is the Simultaneous Localization and Mapping (SLAM) [1], which depends on building a map for the environment and using localization and path planning to guide the robots. Nevertheless, SLAM has a high sensitivity to sensor noise and requires a high-precision global map, which limits its applicability in the real world. Recently, deep reinforcement learning (DRL) has shown significant improvement in solving navigation tasks due to its powerful representation and adaptive learning in dynamic environments. Nonetheless, there are two main challenges in utilizing DRL for solving the navigation task. First, agents interact in a high-dimensional and noisy observation space while receiving a sparse reward, which limits their ability to generalize to unseen scenarios. Second, agents have partial observations and communicate over a noisy channel under limited channel capacity, which affects their task performance.

To overcome the first challenge, the traditional method to reduce a large state space is to use function approximation [2], which empowers the agent to learn a compressed representation of the environment. However, this function approximation is realized by deep neural networks, which suffer from poor out-of-distribution generalization and limited learning transfer. Besides, large samples are required to learn a good policy. Abstraction is a more efficient methodology to learn a compressed and powerful representation for large state/action spaces by filtering out irrelevant information [3], [4]. However, removing too much information may cause a loss of the essential characteristics of the task. Therefore, the authors in [3] studied an abstraction scheme as a compression scheme and investigated the tradeoff between abstraction and task performance. To address the second challenge, sharing a common grounded communication protocol, where agents exchange their partial observability and cooperate to coordinate their behavior was shown to be a promising solution [5]. Under this direction, many studies [5]–[15] have investigated multi-agent communication protocols, which are mainly categorized into pre-defined [14], [15] and learned [5]–[13] communication protocols. For the pre-defined scenario, the communication language is given by the environment, and agents ground their language in representations of the observed world. However, hard coding the communication protocol in dynamic environments may not lead to learning a good policy. In contrast, for the learned scenario, the communication protocol emerges through a consensus and iterative communication process, whereby agents exchange messages and learn to understand and respond to each other to solve a common goal.

The authors in [6] proposed two communication protocol learning approaches Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL). RIAL adopted deep Q-learning while DIAL backpropagate error derivatives through the communication channels. The two approaches rely on centralized training and decentralized execution, which allows agents to exchange discrete messages during task execution. These approaches are not suitable for many real-world problems, in which centralized training causes network congestion and violates privacy. The study in [7] proposed a communication protocol based on straight-through Gumbel-softmax estimators for sequences of discrete messages. The authors show that longer message sequences and their order allow agents to learn faster protocols. However, the authors relied on a hybrid model, which includes supervision through a trained language model to ground the learned protocol. This limits the application of their approach to other tasks. The study in [8] proposed a learning approach based on inductive biases for positive signaling and positive listening. However, the implementation needs several hyperparameters tuning for different tasks. The authors in [9] proposed an observation autoencoder and communicated the encoded representation to learn a grounded communication protocol. Nevertheless, the proposed scheme is not suitable for real-world scenarios, where agents need to be selective about what

Salwa Mostafa is with the Centre for Wireless Communications, the University of Oulu, Oulu, Finland, and the Faculty of Electronic Engineering, Menofia University, Menofia, Arab Republic of Egypt. Mohamed K. Abdel-Aziz is with Nokia Bell Labs, Espoo, Finland. Mehdi Bennis are with the Centre for Wireless Communications, the University of Oulu, FI-90014 Oulu, Finland. (e-mail: salwa.mostafa@el-eng.menofia.edu.eg, mehdi.bennis@oulu.fi, mohamed.3.abdelaziz@nokia-bell-labs.com). (*Corresponding author: Mohamed K. Abdel-Aziz.*)

information to communicate. The work in [10] analyzed an interpretable communication protocol. However, the authors considered a simple and non-dynamic setup, which limits the generalization of their results.

In the aforementioned studies, the agents take actions based on their raw pixel observations, which are typically unstructured. This raw data does not allow them to reason over the received communication messages and generalize to new environments. Moreover, the state space is very large and complex, and agents take a long time to converge. Therefore, this work focuses on the problem of achieving meaningful emergent communication in a fully decentralized interacting multi-agent environment with a communication channel. Inspired by [4], we encode a raw image into a structure at the level of objects, their features, and relations. Then, abstraction allows us to keep the most relevant information to the agent's current state. The proposed framework allows the agents to reason and generalize beyond their partial observations and received communication messages. Our contribution can be summarized as follows:

- We investigated the navigation problem where mobile robots (i.e., agents) have high-dimensional and partially observable observation spaces.
- To solve the problem, we propose a communication protocol learning framework based on structured observation and abstraction to reduce the large state space size and help the agents generalize to new environments. Two types of abstractors are proposed, one is based on compression called RGAE, and the other one keeps the most relevant information to the agent's current state named MLP abstractor.
- Numerical results demonstrate that structured observation and abstraction allow the agents to learn good policies and generalize to new environments. It is also shown that task performance improves through communication.

The rest of the paper is organized as follows. In Section II, we state our system model and reinforcement learning framework. In Section III, we introduce our proposed communication protocol learning framework and explain the policy network in Section IV. Section V provides our simulation model and results. Finally, we conclude the paper in Section VI.

II. System Model

We consider N agents indexed by $\mathcal{N} = \{1, 2, \dots, N\}$ that aim to cooperatively navigate a grid world environment to reach a target destination. Each agent has a partial view of the grid world collected from its sensory camera and sends information to other agents over a shared communication channel during the navigation. We model the interaction among the agents as a decentralized multi-agent reinforcement learning (MARL) with a communication module, which can be expressed as a partially observable Markov decision process (POMDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where

- $\mathcal{S} = \{\mathcal{S}^1, \mathcal{S}^2, \dots, \mathcal{S}^N\}$ is the state space.
- $\mathcal{A} = \{\mathcal{A}^1, \mathcal{A}^2, \dots, \mathcal{A}^N\}$ is the action space, where $\mathcal{A}^{(n)} = \{\text{up } \uparrow, \text{down } \downarrow, \text{right } \rightarrow, \text{left } \leftarrow\}$.
- $\mathcal{C} = \{\mathcal{C}^1, \mathcal{C}^2, \dots, \mathcal{C}^N\}$ is the set of communication messages sent by the agents.
- $\mathcal{O} = \{\mathcal{O}^1, \mathcal{O}^2, \dots, \mathcal{O}^N\}$ is the observation space.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A}^1 \times \mathcal{A}^2 \times \cdots \times \mathcal{A}^N \rightarrow \Delta(S)$ is a non-deterministic transition function that maps the state and joint action space to a probability distributions $\Delta(\mathcal{S})$ over $\mathcal{S}$.
- $R : \mathcal{S} \times \mathcal{A}^1 \times \mathcal{A}^2 \times \cdots \times \mathcal{A}^N \rightarrow \mathcal{R}$ is a reward function, which maps the current states and joint actions to a set of real numbers.
- γ: is the discount factor.

The agents interact over a finite time horizon T. At time step t, each agent n observes a partial view $o_t^{(n)}$ of the underlying state s_t and a set of communicated messages from the previous time step $c_{t-1} = \{c_{t-1}^{(1)}, c_{t-1}^{(2)}, \dots, c_{t-1}^{(N)}\}$. Afterwards, each agent chooses a physical action $a_t^{(n)} \in \mathcal{A}^n$ and a communication action $c_t^{(n)} \in \mathcal{C}^n$ to broadcast and receives a reward $r_t^{(n)} \in R(s_t, a_t)$. Given the joint actions of all N agents $a_t = \{a_t^{(1)}, \dots, a_t^{(N)}\} \in (\mathcal{A}^1, \dots, \mathcal{A}^N)$, the transition function $\mathcal{T}$ maps the current state s_t and the set of agent actions a_t to a probability distribution $\Delta(S)$ over the next state s_{t+1}. We consider a fully cooperative setting in which the objective of each agent is to maximize the total expected return of all agents defined as

$$\max_{\pi : \mathcal{S} \rightarrow \mathcal{A} \times \mathcal{C}} \mathbb{E}\left[\sum_{t=1}^{T} \sum_{n=1}^{N} \gamma^t R(s_t, a_t) \mid (a_t, c_t) \sim \pi^{(n)}, s_t \sim T(s_{t-1}) \right].$$

The total expected return of all agents is maximized using policy gradient methods [16].

III. Proposed Communication Protocol Learning Framework

In this section, we propose a framework for learning a meaningful and generalized communication protocol through structured observations. The main building blocks for our proposed framework are shown in Fig.1. At each time step, the agent receives a partial observation correlated with state s_t, which is an image that represents his partial view of the grid world environment. The *graph constructor* converts the raw pixel observation into a *structured relational graph*[1]. Then, the *relational graph convolutional network* (R-GNC) processes the relational graph to map the feature vectors[2] of each node to a new feature vector that aggregates the messages exchanged from neighborhood nodes under each relation type. Afterward, the processed graph is abstracted through a *graph abstractor* to keep the most relevant information to the current state of the agent. Finally, we pass the output to the RL policy to take the next action and choose a communication action to broadcast in the next time step.

[1] A relational graph is a directed labeled graph, where the vertices represent the entities in the system (i.e. tiles), and the directed edges represent relations (i.e. left, right, ect.).

[2] A feature vector represents the properties of a node (i.e. object type, color, location).

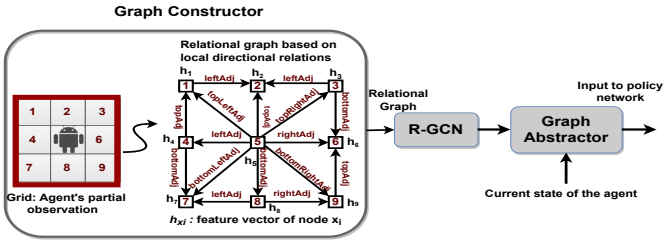


Figure 1: Communication Protocol Learning Framework.

A. Graph Constructor

Since the partial observation is a grid world image, we construct a spatial relational graph to capture the spatial relationship between tiles[3]. A *relational graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ is defined as a directed labeled graph. The set $\mathcal{V}$ represents the set of nodes (i.e. entities/tiles), the set $\mathcal{E} = \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ denotes the set of directed edges (relations) and the set $\mathcal{R}$ represents the set of relation labels (relation type). The relational graph is constructed as follows. Each tile in the grid is represented by a node in the graph. Then, given the coordinates of two different nodes (x_i, y_i) and (x_j, y_j), the edges are defined according to the relational determination rules listed in Table.I. Note that we assume that the tile width and height are equal to z. The determination rule is defined as $r(v_i, v_j) \leftarrow$ *condition*, where $r(v_i, v_j)$ is a relation from node v_i to node v_j with label r and *condition* is a logical statement. If the condition is true, the relation label will be added to the set of relation labels between nodes v_i and v_j. The spatial relations are categorized into three categories: local, remote, and adjacent and aligned. The local directional relations indicate the directional relation between two adjacent nodes. Remote directional relations capture the relative positions between nodes. The aligned relation indicates if two nodes are on the same horizontal or vertical line. Adjacent relations indicate whether two nodes are adjacent to each other [17]. Constructing a relational graph allows us to encode the features of each tile in the grid and perform aggregation among nodes via message passing. Thus, when we remove some nodes via abstraction the information about these nodes has already been passed to other nodes. In so doing, this relational graph has advantages over just using tiles coordinates.

B. Relational Graph Convolutional Network (R-GCN)

After each agent converts its partial view image to a relational graph, the constructed relational graph is processed in a Relational Graph Convolutional Network (R-GCN) [18]. R-GCN maps the feature vector $\boldsymbol{h}$ of each node, which includes the tile type and color to a new feature vector $\boldsymbol{h'}$ conditioned on the relational graph and the parameters $\boldsymbol{W_r}$ attached to each relation label r. The update rule of a feature vector is given by $\boldsymbol{h'}_{x_i} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{x_i\in\mathcal{N}^r_{x_i}} \frac{1}{c_{x_i,r}} \boldsymbol{W_r h_{x_j}} + \boldsymbol{W}_0 \boldsymbol{h_{x_i}}\right)$, where σ is a non-linear function, $\mathcal{N}^r_{x_i}$ is the set of neighborhood nodes to the node x_i with relation type r and $c_{x_i,r}$ is a constant. The update rule takes into consideration the type and direction of an edge and provides relation-specific transformations. It accumulates the feature vectors of neighboring nodes through a normalized sum.

The basis decomposition method is used for regularization to avoid overfitting the model in multi-relational data. The weight parameter $W^{(l)}_r$ is a linear combination of basis transformations $V^{(l)}_b \in R^{d^{(l+1)}\times d^{(l)}}$ with coefficients $a^{(l)}_{rb}$ defined as $W^{(l)}_r = \sum_{b\in B} a^{(l)}_{rb} V^{(l)}_b$

[3]Note that if the partial view is not a grid, we construct a relational graph, where the nodes represent the objects in the partial view and the edges represent the relations between them.

Table I
SPATIAL RELATIONS

Relations	Relational Determination Rules
Local Directional Relations	rightAdj $(v_i, v_j) \leftarrow (x_i = x_j + z) \wedge (y_i = y_j)$ leftAdj $(v_i, v_j) \leftarrow (x_i = x_j - z) \wedge (y_i = y_j)$ topAdj $(v_i, v_j) \leftarrow (y_i = y_j + z) \wedge (x_i = x_j)$ bottomAdj $(v_i, v_j) \leftarrow (y_i = y_j - z) \wedge (x_i = x_j)$ topRightAdj $(v_i, v_j) \leftarrow (x_i = x_j + z) \wedge (y_i = y_j + z)$ topLeftAdj $(v_i, v_j) \leftarrow (x_i = x_j - z) \wedge (y_i = y_j + z)$ bottomRightAdj $(v_i, v_j) \leftarrow (x_i = x_j + z) \wedge (y_i = y_j - z)$ bottomLeftAdj $(v_i, v_j) \leftarrow (x_i = x_j - z) \wedge (y_i = y_j - z)$
Remote Directional Relations	right $(v_i, v_j) \leftarrow x_i > x_j$ left $(v_i, v_j) \leftarrow x_i < x_j$ top $(v_i, v_j) \leftarrow y_i > y_j$ bottom $(v_i, v_j) \leftarrow y_i < y_j$
Alignment and Adjacency Relations	aligned $(v_i, v_j) \leftarrow (x_i = x_j) \vee (y_i = y_j)$ adjacent $(v_i, v_j) \leftarrow (\lvert x_i - x_j\rvert \leq z) \wedge (\lvert y_i - y_j\rvert \leq z)$

C. Graph Abstractor

The relational graph can be very informative, however, the agent needs only the most relevant information to its current state to take action. Therefore, we use a graph abstractor to keep the most relevant information. We consider two types of graph abstractors, one that is based on compressing the input observation called *relational graph autoencoder (RGAE)* [19]. The other abstractor filters the relevant information and is called *multilayer perceptron (MLP) Abstractor*. RGAE provides a degree of abstraction since it is a compressed representation of the input observations. RGAE aims to minimize the reconstruction error of the relational graph to learn an abstract latent representation for the observation space. On the other hand, our proposed MLP graph abstractor consists of an MLP neural network and a Gumbel softmax. The MLP gives a score to each relational edge whether to remove or keep it based on the aggregated graph processed through the R-GCN and the agent's current state. Then, the Gumbel softmax samples according to the obtained score. Finally, nodes that have no edges anymore are removed from the relational graph before passing it to the policy network. Note that we keep the relational directed edges between the current position of the agent and other tiles in the grid without any abstraction so that the agent can determine his next step.

IV. POLICY NETWORK

Each agent under our framework learns an independent network policy. To find an optimal policy, we use the model-free on-policy-based method asynchronous advantage actor-critic (A3C) [20]. The main advantage of using A3C is its

This article has been accepted for publication in IEEE Networking Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LNET.2023.3285295

ability to reliably train the neural network policies without the need for large resources (i.e., replay memory, GPUs) and reduce the training time. In A3C, each agent learns a robust policy during the training process by taking samples from different episodes, which with a high probability are uncorrelated. We use two separate network policies for the physical and communication actions.

A. Physical Action Network

For the physical action network, A3C estimates a policy $\pi(a_t|o_t;\theta)$ and a value function $V(s_t;\theta_v)$, where θ and θ_v are the weight parameters for the policy and value optimizer, respectively. The A3C algorithm interacts with the environment and collects state transitions. The actor takes the abstracted observation o_t and evaluates a reward r_{t+1} and a next state s_{t+1}. The critic evaluates the value of the action taken. Afterward, the current value and policy are updated. The actor weight parameter is updated by $\nabla_\theta \log \pi(a_t|s_t;\theta)A(s_t,a_t;\theta,\theta_v)$, where $A(s_t,a_t;\theta,\theta_v) = r_{t+1} + \gamma V(s_{t+1};\theta_v) - V(s_t;\theta_v)$ is an estimate of the advantage function. The critic weight parameter is updated by minimizing the mean squared error with the Bellman update equation $\partial(R - V(s_t;\theta_v))^2/\partial\theta_v$. Then, it deletes the data samples before starting the next episode.

We use the Adam optimization algorithm to optimize the policy network. The training of the graph abstractor is done while training and optimizing the policy network. The total loss consists of the policy loss and the graph abstractor loss. The policy loss is defined as in [16]. The graph abstractor loss is defined as Graph abstractor loss $= \sum_{r\in\mathcal{R}} \sqrt{\sum_{(i,j)\in \boldsymbol{A_r}} |a_{ij}|^2}$ is the sum of the Frobenius norm of each adjacent matrix $\boldsymbol{A_r}$ representing a relation type in the relational graph.

B. Communication Action Network

For the communication action network, we adopt the A3C model similar to the action network. At each time step, each agent sends a message which is a discrete element from an abstract symbol vocabulary set of size $|\mathcal{C}|$. The symbols are treated as abstract categorical variables, which have no associated meaning that is transmitted by each agent and observed by all other agents. The agent assigns interpretable concepts to the symbols during the training process. Each message symbol c is represented by a one-hot encoding vector $\boldsymbol{c}$. Communication actions require the transmission of discrete symbols. Thus, we use Gumbel-Softmax distribution, which is a continuous relaxation of a discrete categorical distribution. Straight-through Gumbel-softmax estimators allow end-to-end differentiation while using only discrete messages in the forward path.

V. Simulation Model and Results

In this section, we evaluate the performance of our proposed framework for solving the cooperation and communication problem to successfully perform a navigation task. The environment is taken from the GridWorld with a cluttered density of obstacle tiles 0.15, where the environment states are randomized at every episode. We consider two agents aiming to reach a common goal in a 15×15 grid world. At each time step, each agent has a 7×7 partial view of the environment-centered at its current location. Each agent gains a reward of 1 if it reaches the goal. A team reward of 1 is received when both agents reach the goal. Each episode has a maximum length of 1024 time steps. The Adam optimizer has a learning rate of 0.0001 and parameters $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$. The R-GCN is implemented with 8 graph convolutional layers. The parameter's setting is chosen based on our empirical experimentation.

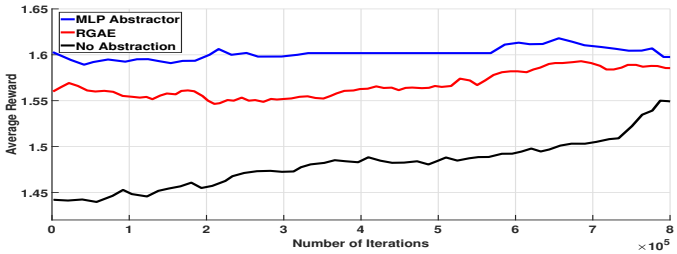
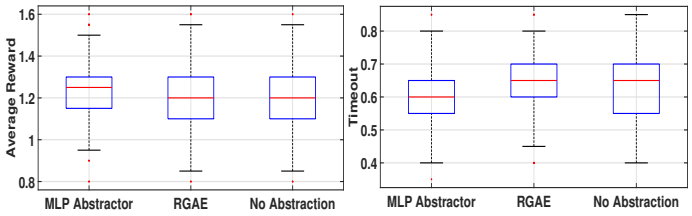
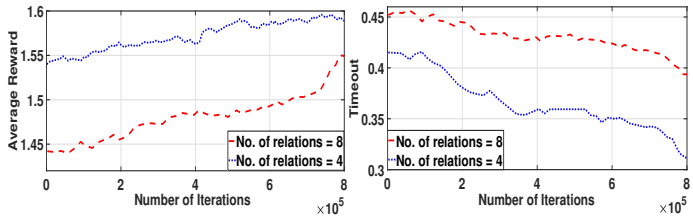


Figure 2: Average reward versus the number of iterations.



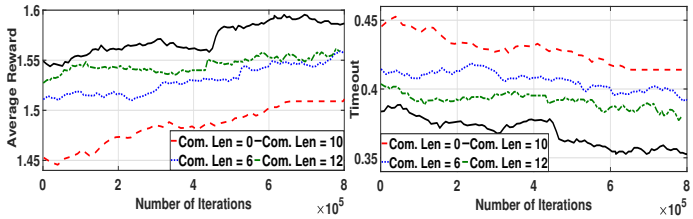
(a) (b)

Figure 3: (a) Effect of the relational graph size on the average reward.(b) Effect of the relational graph size on the timeout.



(a) (b)

Figure 4: (a) Effect of the relational graph size on the average reward.; (b) Effect of the relational graph size on the timeout.



(a) (b)

Figure 5: (a) Average reward versus number of iterations; (b) Timeout versus number of iterations.

Figure 2 shows the performance of our proposed MLP abstractor and the RGAE. It is shown that the MLP abstractor and RGAE outperform the no-abstraction scheme in terms of average reward, which indicates that the agents learn better policies in the abstracted observation space. As expected our proposed MLP abstractor outperforms RGAE since it keeps only the relevant information while RGAE compresses the information. Figures 3a and 3b evaluate the performance of the abstractor's over 3000 episodes in a new environment, where the cluttered density increased to 0.25. It is shown that the MLP abstractor allows the agents to generalize in new environments compared to RGAE and the no-abstraction scheme. The reason is that the MLP abstractor keeps the most relevant information to each agent state, which allows the agents to learn and reason better in new unseen scenarios.

Figure 4a and 4b demonstrate the effect of relational graph size on task performance. We consider the eight and four relations representing the local and remote directional relations in Table I. As we can see, the relational graph with four relations outperforms the relational graph with eight relations in terms of the average reward and timeout, which supports the idea of abstraction. The reason is that keeping the most relevant information to the agent's current state helps him to reason and learn better policies. Note that the timeout characterizes the percentage of time the task fails. Specifically, the task fails when the maximum episode length is reached and one of the agents did not arrive at the goal.

Figure 5a and 5b demonstrate the effect of communication on performance. As we can see, the average reward is increased and the timeout is decreased for communication lengths of 6, 10, and 12 bits compared to no communication. This shows that communication helps the agents to learn better policies and reach their goals faster. However, when the communication length becomes too large such as communication length 12, the performance of the task starts to degrade as the vocabulary size increases to 2^{12} messages, and it takes the agents more time to interpret the meanings of the messages Therefore, we conclude that the agents learn better policies in an abstract observation space, where only the important and relevant information of its state is kept. Moreover, communication improves task performance as it allows agents to exchange messages about their partial view of the environment.

VI. Conclusion

In this work, we investigated the problem of learning how to cooperate and communicate in a partially observable distributed environment. We proposed a communication protocol learning framework based on structured observation to help agents reason and generalize over the communicated messages. We showed that abstraction can help agents learn better policies and generalize to new environments. In future work, we plan to extend our framework to structured protocols, where sequences of symbols are exchanged between agents.

VII. ACKNOWLEDGMENT

The work is funded by the European Union under Grant Agreement 101096379 through the project CENTRIC, Grant Agreement 101096034 through the project SNS JU VERGE, and Grant Agreement 957218 through the project EU-ICT IntellIoT. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. We would like to thank Nokia Bell Labs (Espoo) Finland for the discussion and collaboration.

References

[1] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[2] S. S. Du, S. M. Kakade, R. Wang, and L. F. Yang, "Is a good representation sufficient for sample efficient reinforcement learning?," *arXiv preprint arXiv:1910.03016*, 2019.

[3] D. Abel, D. Arumugam, K. Asadi, Y. Jinnai, M. L. Littman, and L. L. Wong, "State abstraction as compression in apprenticeship learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3134–3142, 2019.

[4] M. Shanahan and M. Mitchell, "Abstraction for deep reinforcement learning," *arXiv preprint arXiv:2202.05839*, 2022.

[5] A. Lazaridou and M. Baroni, "Emergent multi-agent communication in the deep learning era," *arXiv preprint arXiv:2006.02419*, 2020.

[6] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[7] S. Havrylov and I. Titov, "Emergence of language with multi-agent games: Learning to communicate with sequences of symbols," *Advances in neural information processing systems*, vol. 30, 2017.

[8] T. Eccles, Y. Bachrach, G. Lever, A. Lazaridou, and T. Graepel, "Biases for emergent communication in multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 32, 2019.

[9] T. Lin, J. Huh, C. Stauffer, S. N. Lim, and P. Isola, "Learning to ground multi-agent communication with autoencoders," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15230–15242, 2021.

[10] I. Kajić, E. Aygün, and D. Precup, "Learning to cooperate: Emergent communication in multi-agent navigation," *arXiv preprint arXiv:2004.01097*, 2020.

[11] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[12] M. P. Mota, A. Valcarce, J.-M. Gorce, and J. Hoydis, "The emergence of wireless mac protocols with multi-agent reinforcement learning," in *2021 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, 2021.

[13] L. Miuccio, S. Riolo, S. Samarakoony, D. Panno, and M. Bennis, "Learning generalized wireless mac communication protocols via abstraction," *arXiv preprint arXiv:2206.06331*, 2022.

[14] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

[15] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous agents and multi-agent systems*, vol. 11, no. 3, pp. 387–434, 2005.

[16] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[17] Z. Jiang, P. Minervini, M. Jiang, and T. Rocktaschel, "Grid-to-graph: Flexible spatial relational inductive biases for reinforcement learning," *arXiv preprint arXiv:2102.04220*, 2021.

[18] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*, pp. 593–607, Springer, 2018.

[19] S. Zhang, Z. Zhang, F. Zhuang, Z. Shi, and X. Han, "Compressing knowledge graph embedding with relational graph auto-encoder," in *2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pp. 366–370, 2020.

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR, 2016.