Multi-Agent Pattern Formation: a Distributed Model-Free Deep Reinforcement Learning Approach

Elhadji Amadou Oury Diallo
Computer Science and Communications Engineering
Waseda University
Tokyo 169-8555, Japan
Email: diallo.oury@fuji.waseda.jp

Toshiharu Sugawara
Computer Science and Communications Engineering
Waseda University
Tokyo 169-8555, Japan
Email: sugawara@waseda.jp

***Abstract*—In this paper, we investigate how a large-scale system of independently learning agents can collectively form acceptable two-dimensional patterns (*pattern formation*) from any initial configuration. We propose a decentralized multi-agent deep reinforcement learning architecture MAPF-DQN (Multi-Agent Pattern Formation DQN) in which a set of independent and distributed agents capture their local visual field and learn how to act so as to collectively form target shapes. Agents exploit their individual networks with a central replay memory and target networks that are used to store and update the representation of the environment as well as learning the dynamics of the other agents. We then show that agents trained on random patterns using MAPF-DQN can organize themselves into very complex shapes in large-scale environments. Our results suggest that the proposed framework achieves zero-shot generalization on most of the environments independently of the depth of view of agents.**

***Index Terms*—deep reinforcement learning, multi-agent systems, pattern formation, self-organization, swarms**

I. Introduction

Suppose, for an instance, that an instructor needs her n students in the play area to form a 2D shape such as a circle so that, for example, they can play a game. The instructor may draw a guideline on the ground as a rule or even give every student a particular position to move to. Now, imagine a scenario where the instructor does not give such help. Indeed, even without such help, the kids may, in any case, have the option to form an adequately decent estimation of the circle if every one of them moves depending on the movement of others by directly observing their neighborhood region. If successful, this method can be called a distributed solution to the circle formation problem for children [1].

By analogy, we utilized a methodology based on the previous example [1] to control a large-scale multi-agent systems of homogeneous teams. The principal idea is to give every agent the opportunity to execute a straightforward estimation of its states and accordingly plan its actions depending on the actions and states of the remaining agents so that the agents as a team will cooperatively and collectively form the target pattern. This type of distributed control of cooperative systems, in which mobile agents work together to perform cooperative tasks, is one of the most challenging problems in multi-agent and multi-robot systems. The challenge is that such a system is expected to have the ability to self-organize itself by learning cooperative behaviours without any human intervention. It must be emphasized that, self-organization of a real-world (drones, cars, ...) multi-agent system is much harder to achieve as agents are required to learn more complex tasks.

Our framework could be applied to many realistic problems such as coordinated multi-robot exploration [2], multi-robot navigation [3], shape constraints in crowd simulation [4], and multi-robot animation for entertainment [5]. In general, the ability for the multi-agent team to cooperate appears to rely enormously upon (1) the characteristics of the problem to solve, (2) the global properties assigned to the team, and (3) the distinctive capabilities of each agent. Instances of the global properties are the capacity to recognize at any rate their team members, to concur on a common global direction (sense of direction), or to concede to a typical handedness [6].

In this paper, we investigate how a large-scale system of independently learning agents can collectively form acceptable two-dimensional patterns (*pattern formation*) from any initial pattern and configuration. On our assumptions, an agent is to be regarded as a point in the plane that autonomously moves according to a given rule. In general, an agent observes the environment, computes its next position with a given algorithm, and moves to its next position until it finds its goal position. We assume that the agents are homogeneous and anonymous in the sense that they have no identifiers. We also assume that the agents are uniform in the sense that all agents synchronously execute a common algorithm. Each agent has no access to the global state and its actions are solely done in terms of its local observation [7].

This research proposes an end-to-end decentralized learning architecture in which agents (1) do not explicitly communicate; (2) use a centralized replay memory to share knowledge among agents who share the same global goal; and (3) use a centralized target network to take into account the dynamics of others and provide a quantitative estimate of how each agent perceives and is perceived by its team members. We call this method *Multi-Agent Pattern Formation with Deep Q-Network* (MAPF-DQN). The goal is to control the overall shape of a robot team by using only the local information provided by agents' sensors. Interestingly, the positions or goals of the individual agents in the group are not explicitly controlled. An agent should concurrently and independently learn to locate its goal position and consequently plan a smooth trajectory towards it. Our model is invariant to the number of agents per team. This makes it easier to transfer the learned behaviours of one team to another one with different a number of agents. We further show that agents using MAPF-DQN can learn complex cooperative strategies in environments with progressively increasing complexities.

II. Related Work

To date, the vast majority of current cooperative multi-agent approaches have been based either on centralized methods in which a lead agent predicts the behaviour of all agents or on a distributed approach in which the agents have a full view and understanding of their environment and dynamics or sometimes on completely independent learning systems that do not have anything in common [8]. The easiest way to implement the decentralized method is to use Q-learning to estimate independent Q-value functions of each agent by considering that the other agents are part of the environment. As the other agents are now also learning, the environment is non-stationary and thus this uncertainty limits such approaches from expanding to more than a few agents.

A handful of the latest works uses the centralized learning and decentralized execution paradigm [9], [10]. This consists of actor-critics methods in which the critic is centralized and uses all the available information from the environment during training. However, throughout execution, agents use their independent network to compute their actions in a completely distributed and independent fashion. For eample, MADDPG [9] uses the joint actions and states with the critic and establishes strategies for each actor by using DDPG [11] and concatenating information from other agents.

Huttenrauch et al. [10] argued that most of the recent multi-agent deep reinforcement learning algorithms are limited when they are applied to swarm systems without special care. One of their argument is that concatenating the information received from different agents is not optimal when you have an environment with a dynamic number of agents. They also argued that this could disregard the inherent permutation invariance of identical agents sine qua non to swarm systems and that doing so would hardly scale to large-scale systems. To remediate these limitations, Huttenrauch et al. [10] proposed a method in which they treat the observation perceived from nearby agents as samples of a random variable and then encode the current distribution of the agents by using mean feature embedding [12].

An alternative to learning systems would be optimization-based algorithms. However, to control the strategies of agents, optimization-based methods [13], [14] often oversimplify the model of agents by assuming unrealistic assumptions in well defined tasks or environments [13], [15], [16] or having a full view of the environment [16].

On the swarm optimization side, Xu et al. [17] tackles the problem of homogeneous multi-agent pattern formation by using a natural swarm algorithm inspired by the particle swarm optimization and by using a virtual pheromone as the messaging protocols. This was proposed to limit the complexity of communication channels as the number of agents increases. In their method, the agents leave virtual pheromone in the environment. The pheromone is updated based on their local observations. However, agents still have to explicitly communicate by broadcasting the pheromone density to other agents located in their communication range.

Regardless, in many real-world applications, most of these hypotheses are infeasible or superfluous. As an example, when the number of agents is large, it then becomes apparent that most of the centralized methods will not scale because agents would have to share their data to the lead agent. Therefore, in case of agents with communication abilities, we cannot continually make sure that the communication channels are maintained up-to-date and not congested.

III. Preliminaries

A. Pattern formation problem

Given a pattern $P = \{p_1, ..., p_n\}$, or a set of points (given by their Cartesian coordinates in the plane) on an environment that is a two-dimensional grid, and $R = \{1, ..., n\}$ a set of n anonymous agents. Let $F(t) = \{f_1(t), ..., f_n(t)\}$ be the formed pattern by agents where $f_i(t) \in \mathbb{R}^2$ are the coordinates of the agent i at time t in the environment and $\mathbb{R}$ is the set of real numbers. The goal of the pattern formation problem is to find a near-optimal and decentralized algorithm to such an extent that from any initial distributions of the agents positions, they will, in the long run, organize themselves to form the target shape.

In general, the initial configuration of the pattern in the environment is unknown to the agents. Instead, they have to explore the environment in its entirety in order to find the set of coordinates of the goal positions before moving towards them. However, they have no access to a global view of the environment. A consequence of the approach just outlined is that agents will eventually maximize their rate of success. We say that the agents successfully form a target pattern P from any initial position of agents $F(0)$ when their final positions $F(T)$ is 85% similar to the target pattern P, i.e. $(F(T) \approx P)$ where T is the terminal timestep of an episode. Note that each agent can only observe a subset of P located in its local view only.

B. Model

Most real-world problems can hardly be modeled as an (PO)MDP because the agents have limited communication and partial or noisy observations provided by their sensors. Clearly, they would have to learn cooperative and coordinated behaviours by using only their local information. For this reason, we use the dec-POMDP framework [18].

Definition 1: A decentralized partially observable Markov decision process (dec-POMDP) [18] is defined as a tuple $\langle \mathcal{D}, \mathcal{S}, \boldsymbol{\mathcal{A}}, \mathcal{T}, \mathcal{R}, \boldsymbol{\Omega}, \mathcal{O}, h, \mathcal{I} \rangle$, where

- $R = \{1, ..., n\}$ is a set of n agents;
- $\mathcal{S}$ a finite set of states s in which the environment can be;
- $\boldsymbol{\mathcal{A}}$ is the final set of joint actions of agents, where $\boldsymbol{\mathcal{A}} = \mathcal{A}^1 \times ... \times \mathcal{A}^n$ and A^i is the action space of agent i;
- $\mathcal{T}$ is a probabilistic transition function;
- $\mathcal{R}$ is the immediate reward function of an agent;
- $\boldsymbol{\Omega}$ is the finite set of joint observations, where $\boldsymbol{\Omega} = \mathcal{O}^1 \times ... \times \mathcal{O}^n$ and $\mathcal{O}^i$ is the set of observations available to agent i;
- $\mathcal{O}$ is the observation probability function;
- h is the horizon of the problem; and
- $\mathcal{I}$ is the initial state distribution at time $t = 0$.

At every step, the environment transits from s_t to s_{t+1} with a probability $p(s_{t+1}|s_t, \mathbf{a}_t) \in \mathcal{T}$ when all agents execute a joint action $\mathbf{a}_t = \langle a_t^1, \dots a_t^n \rangle$. Then agent i receives a reward $r_t^i = \mathcal{R}(s_{t+1}|s_t, a_t^i)$. The observation o can be approximated by a *zth order history* approach which uses the last z observations and actions [19]. This is very important because the agents no longer have a Markovian signal; they can't neither observe the state nor estimate the belief b as in POMDPs. Therefore, this approach can manage any latent state information compared to using directly the current observation as the input of an agent. Note that doing so makes our agents *non-oblivious* because each agent r_i uses the same algorithm ψ and the past observations and actions of ψ. This gives the agents the ability to remember important information such as the positions of teammates and/or the target pattern.

C. Deep Reinforcement Learning

In reinforcement learning, agent i learns an optimal control policy π_i. At each step t, i observes the current state s, chooses an action $a \in \mathcal{A}^i$ using π_i, receives a reward r_t^i. Then, the environment transits to a new state s_{t+1}. The goal is to maximize a cumulative discounted future reward at time t_0 as

$$R_t = \sum_{t=t_0}^{T} \gamma^{t-t_0} r_t, \tag{1}$$

where T is the terminal time step and $\gamma \in [0, 1]$ is the discount factor that weights the importance of rewards. The action-value of a given policy π^i represents the utility of an action a at a state s, where the utility is defined as the expected and discounted future reward,

$$Q^\pi(s, a) = \mathbb{E}\left[R_t | s_t = s, a_t = a\right]. \tag{2}$$

The optimal Q^* is defined as

$$Q^*(s, a) = \max_\pi Q^\pi(s, a). \tag{3}$$

Q-learning [20] is an approach that iteratively estimates the Q-function using the Bellman optimality

$$Q^*(s, a) = \mathbb{E}\left[r + \gamma \max_a Q^*(s_{t+1}, a) | s, a\right]. \tag{4}$$

We can use a parameterized value function $Q(s, a; \theta_t)$ when the task is complex. Then, we update the parameters by using the following formula:

$$\theta_{t+1} = \theta_t + \alpha(y_t^Q(s_t, a_t; \theta_t)) \nabla_{\theta_t} Q(s_t, a_t; \theta_t), \tag{5}$$

where α is a scalar step size and y_t^Q is the target value function.

$$\begin{aligned} y_t^Q &= R_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta_t) && \text{(6a)} \\ &= R_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_t); \theta_t) && \text{(6b)} \end{aligned}$$

DQN [21], [22] is an extension of Q-learning that uses a stack of neural network to estimate $Q(s, a; \theta)$, where θ are the weights of the network. We use a separate network to estimate the target Q-values, y_t^{DQN}. The target network has the same architecture as the DQN network but with frozen parameters θ^- that are updated after every τ from the online network. This leads to more stable training because it keeps the target θ^- fixed for a while. Now, we can can rewrite Eq. 6 as:

$$\begin{aligned} y_t^{DQN} &= R_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta_t) && \text{(7a)} \\ &= R_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta); \theta^-) && \text{(7b)} \end{aligned}$$

Then, we finally update the neural network weights by:

$$\theta_{t+1} = \theta_t + \alpha(y_t^{DQN} - Q(s_t, a_t; \theta_t)) \nabla_{\theta_t} Q(s_t, a_t; \theta_t) \tag{8}$$

IV. Methods

A. General multi-team architecture

We propose a decentralized system (Figs. 1 and 2) that can handle a dynamic number of homogeneous teams for the problem of multi-agent pattern formation. This is a concurrent team learning in which agents are divided into teams in the two dimensional space. It is well-suited to cooperative multi-agent systems in which a team needs to have information about others in order to make their decisions, i.e., every team learns to improve parts of the global team by sharing its experiences with others and reusing others' experiences at the same time (Fig. 1). Each agent has a limited visible visual field of depth k (shape = $[2k + 1, 2k + 1]$) in the two dimensional space and an agent can observe its teammates, the obstacles and the walls within its neighborhood area. However, the interactions between the agents are not explicitly modelled; they instead have to learn them by observing other agents' movements and reusing others' experiences which are randomly sampled from the centralized experience replay memory.

B. Distributed architecture of a team

The architecture of a team is shown in Fig. 2. Each agent concurrently and independently learns its behaviour based only on its local observations (Fig. 3). In this framework, every agent has its own neural network which is shown in Fig. 3. By using a centralized target network and replay memory, we naively mimic the team modeling framework by learning about other agents in the environment so as to make good estimates of their actions. By doing so, we alter the dynamics such

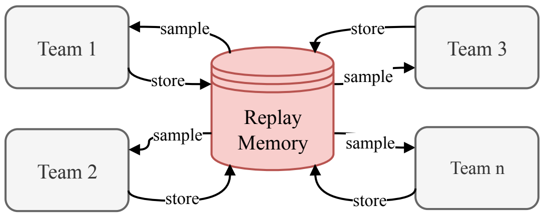


Fig. 1. General architecture with one or more homogeneous teams.

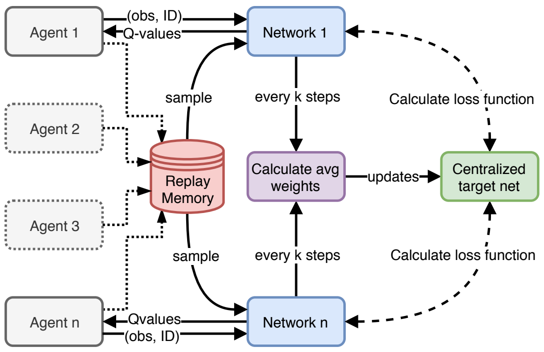


Fig. 2. Distributed model with centralized replay memory and averaged target network.

that every agent can perceive and co-adapt to the dynamics of the environment. This also helps them estimate the current policies of others and thus, achieve a better cooperation [23]. This is equivalent to providing to an agent the internal belief representation required to cooperate with its teammate and potentially avoid conflicts.

The goal positions of individual agents are not defined and must be instead collaboratively learned by agents. By doing that, agents learn to agree on their tasks or assignments in a coordinated manner. We have already seen that the only way they learn to coordinate is restricted to using only their local observation which might contains the positions of other agents and goal positions.

With our method the network continuously adapts and updates the goals depending on the environment dynamics until all agents reach their goals. An agent receives a reward of $+10$ if it finds a landmark and decide to commit to it and receives a punishment of -0.05 at every time step. The last part should potentially demotivate them to spend long time without finding goals. The key components of this framework are: *domain randomization*, *centralized experience replay memory*, and a *common target network* calculated by the average of each agent's weights.

C. Domain randomization

It is well known that transfer learning is hard with reinforcement learning policies. To avoid this trap, we use some ideas from robotics domain randomization. The goal of domain randomization is to train the agents on different environments with random properties and dynamics. In our case, the agents are trained to form themselves into randomly generated shapes. In other words, the positions of the target shapes are randomly generated and scattered all over the environment at the beginning of every episode with a variant number of targets up to 600 simulated agents during training. This will help the system to adapt to different shape during the test phase as they were trained on so many different and variable shapess. Hence, the agents will learn to generalize well to unseen pattern shapes.

D. Centralized experience replay memory and common target network

Each agent of the team has its own main network to behave autonomously. It stores and updates its representation of the environment by randomly sampling from the replay memory. As a consequence, each agent independently computes its patrolling plan by taking advantage of other agents knowledge without exchanging coordination messages. It is important to combine this approach with a good sampling strategy because the existence of multiple agents accessing and updating a centralized memory may result in earlier memories of some agents to be overwritten.

The centralized target network is updated by the average weights of individual agents' networks. In other words, the target network $\boldsymbol{\theta}^-$ is a single network that combines the individual agents' networks (θ) of the same team by taking the average of their weights.

$$\boldsymbol{\theta}^- = \frac{1}{n}\sum_{i=1}^{n}\theta^i. \qquad (9)$$

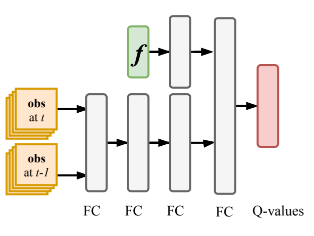
Equation 7b becomes

$$y_t = R_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta); \boldsymbol{\theta}^-). \qquad (10)$$

By doing so, we also ensure that an agent can react to the previous actions and rewards of others by taking into account their dynamics. As a result, this somehow provides some sort of communication to cope with the local view. Moreover, this framework seems to provide the same characteristics of a team with implicit communication and global view. This is still better than most actor-critic methods such as MADDPG [9] in which the central critic limits the ability of agents to generalize their learned behaviours to different environments or even the same environment with different number of agents or action space. The agents are homogeneous and anonymous, so they cannot be distinguished by their appearance. Besides, our framework works in a completely distributed mode and agents have no preference for their goal destinations.

E. Observation and network architecture of an agent

With this approach, agents rely on themselves to make decisions instead of a centralized leader. However, without centralized coordination, our framework works in a complete distributed mode, using only the local environmental information provided by the agents sensors. Agents do not have a global knowledge of their environment and should achieve good results with smooth motions in a relatively short time.



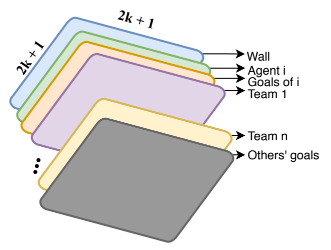


Fig. 3. Left: Network architecture. Right: spatial observation of an agent.

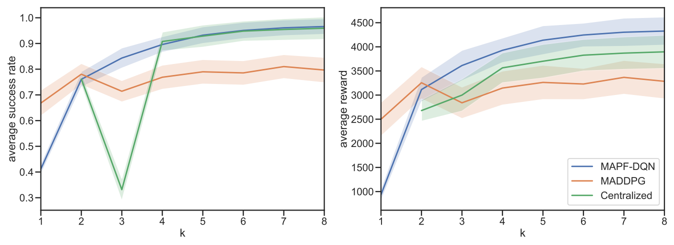


Fig. 4. Left: average success rate. Right: average reward.

The observation of an agent (Fig. 3) is encoded as a multi-channel image in which every channel represents a different feature from its view as in [24]–[26].

Every channel is a binary matrix with 1 indicating the presence of an object and 0 otherwise. The first channel represents positions of the wall and obstacle. The second one represent the position of the agent itself, and the third channel is the position of the goals in that local view. If we have more than one team, we add a new channel for each different team. Finally, the last channel contains all other information such as the goal positions of all other teams. This representation could be pushed further by using ideas from neural-symbolic computing [27] that provides the ability to learn from experiences and at the same time the ability to reason from what has been learned.

The neural network architecture of each agent is shown in Fig. 3. It consists of fully connected layers only. The inputs are the spatial observations of agents (Fig. 3) at t and $t-1$ and a feature vector $\boldsymbol{f}$ from the environment containing the last action, last reward, relative position, number of goal points in their local views. This is very similar to the state representation in [24] with the only difference being that we reused the history to make it conform to our dec-POMDP model.

V. Results

A. Experimental settings

The following results are average of 10 experimental runs with different random seeds by using our proposed framework. We assume that there is no noise in agents' observations and that the target shape is static during every episode. In all experiments, the target network is updated after every $10,000$ steps which also means that the average target network is calculated at the same time. We set the learning rate to 0.0001, $\gamma = 0.9$, batch size of 128 samples, and used ε-greedy as the exploration strategies. The values of ε are linearly decayed from 0.5 to 0.01 during the first 5000 of the 10000 training episodes. The maximum samples that the shared experience replay memory can have at a time is $1,000,000$.

We use a centralized system as a baseline, in which a team's strategy is computed by a central agent and subsequently communicated to all teammates. We also compare our method against a discrete action space version of MADDPG [9]. To improve the effectiveness of our method and its ability to generalize on completely random and unseen environment, the agents were trained on a large 2-dimensional grid graph (150×150) with utmost 700 randomly generated landmarks at the beginning of each episode.

B. Results with one team

First, we trained our model with only one team of utmost 700 agents during training and a varying number of agents during testing. Figure 4 shows the average reward and completion rate of agents with different visual field depths. We do not provide the result of the centralized method when $k = 1$ as that would generate an observation range of 3×3 which would be too small for a convolutional layer with a 3×3 kernel. With a smaller view range $(k = 1)$, MADDPG achieved the highest reward, followed by the centralized network while agents using our method struggle to learn how to organize themselves into the target shapes. As the view range increases $(k \geq 2)$, the proposed method steadily improves while the performance of MADDPG beomes less stable. The instability of MADDPG is due to the exchange of observation during training. Moreover, the centralized framework is not stable either for a smaller view ranges. This is probably caused by the limited scope of the information from their local observations. In summary, with a large enough view range, agents can solve their tasks by using either our proposed method or a centralized system with a lead agent.

While our method has a similar average success rate as the centralized method for $k \geq 4$, it also achieved the highest rewards among all of the tested methods. This means that agents using MAPF-DQN take less time to learn acceptable strategies. Surprisingly, we observe in all methods, agents do prefer to finish most of their tasks during the first hundred steps before slowly trying to complete and improve the global shape (Fig. 9). In addition, MADDPG agents trained on randomly generated patterns cannot generalize well on unseen and structured patterns when the number of agents is large as in Fig 9.

C. Results with many teams

To evaluate the robustness of our method, we train it with a different number of homogeneous teams. The advantage of a multi-team system is that we could eventually use MAPF-DQN with heterogeneous teams with different action spaces, velocities, and learning techniques. The number of agents per team is equal to the number of landmarks divided by the

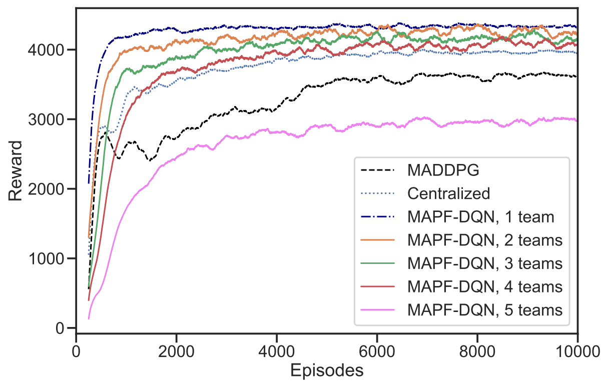


Fig. 5. Multi-team average reward.

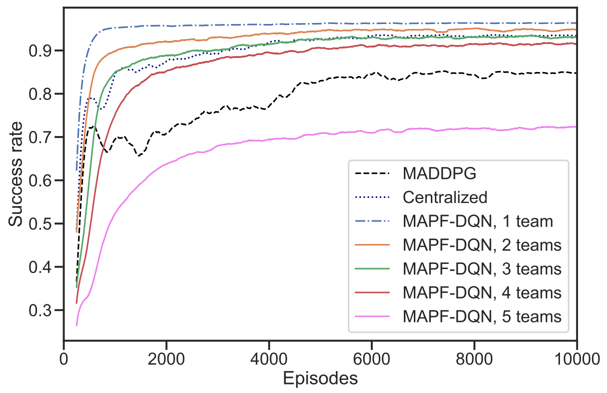


Fig. 6. Multi-team average success rate.

number of teams. Figures 6 and 5 show that our method with 1 to 4 teams do also outperform all other methods during training even though its performance keeps decreasing as the number of teams increases. Surprisingly, we observe a sudden performance drop when the number of teams is 5, which becomes even worse than MADDPG [9].

Figure 7 shows the distribution of rewards for our framework in environment with different number of teams. Our method is more confident when we just have one group as the dimension of the observation is smaller and contains less noise as can be seen in the reward density estimation. The confidence is slightly shrinking as the number of teams increases and the distribution of the rewards starts varying in a larger range. Finally, our method becomes less confident about its strategies as the number of teams becomes larger. It also shows that our method works better with smaller number of teams (up to 4) and does not scale for teams of more than four teams. Therefore, the behaviours become less predictable for a team of five teams. Though, this is not surprising because we know the dimension of the observation of an agent increases as the number of teams increases. This make the dynamics of the environment noisy and thus, very hard to predict. In addition, Figure 8 shows that our method is robust up to 4 teams, from which it becomes less stable and confident about its

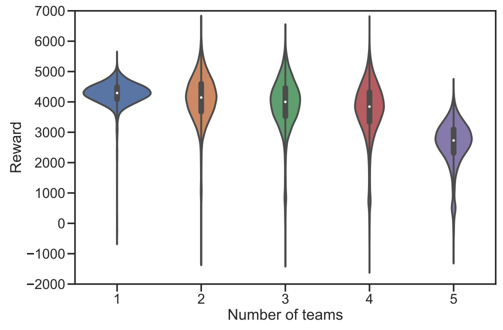


Fig. 7. Distribution of rewards.

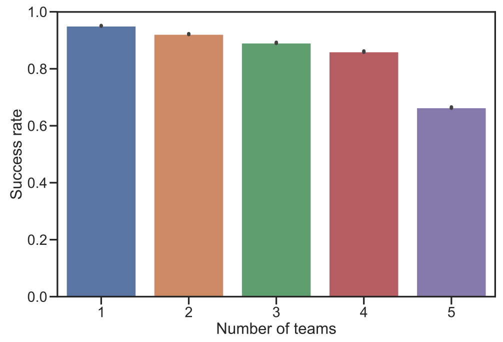


Fig. 8. Multi-team average success rate for $k = 6$.

predictions.

D. Generalization

After evaluating the behaviours of our proposed method with different team sizes with different number of agents, and different observation range without any fine-tuning, our results suggest that the proposed framework achieves zero-shot generalization on all environments independently of the depth of view of agents. Figure 9 shows 550 agents trying to organize themselves into an X shape in a 150×150 grid-graph. At the end of this episode, our proposed method achieves the highest completion rate of 94%, followed by the centralized architecture with 87% and finally MADDPG [9] with 73%. Agents tend to finish the large chunk of their tasks during the first few steps. However, agents with different techniques have different behaviors of representing the shape. For example, with our method, agents tend to start from the center of the shape and progressively explore the rest of the environment (Fig. 9), while with MADDPG [9], the agents, do not really have any preferences, they start exploring the environment in its entirety.

This difference can justify why MADDPG [9] agents do not achieve a good result at the end in large-scale environment. The other reason is that MADDPG [9] hardly scale in large environment because agents have to share their information, i.e., the more agents you have, the higher the complexity of the observation of an agent is. In contrast, our agents hardly explore the whole environment by starting from the center, which often a good behaviour, but you could easily see the limitations when you have a disconnected or discontinue shape. However, we still think that the behaviour of MADDPG [9] could be well-suited for small-scale environment with less agents and simple target shapes.

Figure 10 shows n agents trying to organize themselves into different shapes in a 150×150 grid-graph by using MAPF-DQN. These shapes consist of an "O" shape, a dolphin, a house, and a mandala. As in Fig 9, the agents finish the bulk of the tasks during the very first steps of an episode and then subsequently try to improve their coverage. Even though all teams have utmost 700 agents during training, we can see that they can more or less generalize well in environment with much larger number of agents during the test phase. We can also see that the generalization is somewhat independent to the number of agents but highly tied to the pattern themselves. For example, the dolphin requires only 864 agents and mandala 1669, but our method has a better result with the later.

VI. Conclusion

We showed that agents using our method can organize themselves into a complex 2-dimensional pattern even though they were trained on random patterns. Our results showed that the proposed framework achieved zero-shot generalization on unseen environments without retraining the agents independently of the depth of their views. We finally showed that our framework can generate complex strategies when the team is divided into independent homogeneous groups. However, our method does not scale when we have a large number of groups or teams. Finally, it would be interesting to investigate our method for multi-pattern formations in which agents are expected to achieve smooth transitions between given patterns. Also, it would be interesting to investigate our framework with heterogeneous instead of homogeneous groups and further investigate the robustness of this framework.

Acknowledgment

This research is partly supported by JSPS KAKENHI Grants No. 17KT0044 and 20H04245.

References

[1] I. Suzuki and M. Yamashita, "Distributed anonymous mobile robots: Formation of geometric patterns," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1347–1363, 1999.

[2] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[3] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 73–80.

[4] Q. Gu and Z. Deng, "Formation sketching: an approach to stylize groups in crowd simulation," in *Proceedings of Graphics Interface 2011*. Canadian Human-Computer Communications Society, 2011, pp. 1–8.

[5] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple mobile robots," *The International Journal of Robotics Research*, vol. 31, no. 6, pp. 753–773, 2012.

[6] Y. Dieudonné, F. Petit, and V. Villain, "Leader election problem versus pattern formation problem," in *International Symposium on Distributed Computing*. Springer, 2010, pp. 267–281.

[7] Y. Yamauchi, T. Uehara, and M. Yamashita, "Pattern formation problem for synchronous mobile robots in the three dimensional euclidean space," *arXiv preprint arXiv:1509.09207*, 2015.

[8] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

[9] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[10] M. Hüttenrauch, S. Adrian, G. Neumann *et al.*, "Deep reinforcement learning for swarm systems," *Journal of Machine Learning Research*, vol. 20, no. 54, pp. 1–31, 2019.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[12] A. Smola, A. Gretton, L. Song, and B. Schölkopf, "A hilbert space embedding for distributions," in *International Conference on Algorithmic Learning Theory*. Springer, 2007, pp. 13–31.

[13] J. Lin, A. S. Morse, and B. D. Anderson, "The multi-agent rendezvous problem. part 2: The asynchronous case," *SIAM Journal on Control and Optimization*, vol. 46, no. 6, pp. 2120–2147, 2007.

[14] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *Departmental Papers (ESE)*, p. 29, 2003.

[15] B. Ranjbar-Sahraei, F. Shabaninia, A. Nemati, and S.-D. Stan, "A novel robust decentralized adaptive fuzzy control for swarm formation of multiagent systems," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3124–3134, 2012.

[16] Z. Zhou, W. Zhang, J. Ding, H. Huang, D. M. Stipanović, and C. J. Tomlin, "Cooperative pursuit with voronoi partitions," *Automatica*, vol. 72, pp. 64–72, 2016.

[17] H. Xu, H. Guan, A. Liang, and X. Yan, "A multi-robot pattern formation algorithm based on distributed swarm intelligence," in *2010 Second International Conference on Computer Engineering and Applications*, vol. 1. IEEE, 2010, pp. 71–75.

[18] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes," *Mathematics of operations research*, vol. 27, no. 4, pp. 819–840, 2002.

[19] E. A. O. Diallo and T. Sugawara, "Learning coordination in adversarial multi-agent DQN with dec-pomdps," 2018, workshop on Reinforcement Learning under Partial Observability, 32nd Neurips, 2018.

[20] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[23] E. A. O. Diallo and T. Sugawara, "Multi-agent pattern formation with deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[24] L. Zheng, J. Yang, H. Cai, M. Zhou, W. Zhang, J. Wang, and Y. Yu, "Magent: A many-agent reinforcement learning platform for artificial collective intelligence," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[25] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 2085–2087.

[26] E. A. O. Diallo and T. Sugawara, "Coordination in adversarial multi-agent with deep reinforcement learning under partial observability," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2019, pp. 198–205.

[27] A. d. Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran, "Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning," *arXiv preprint arXiv:1905.06088*, 2019.

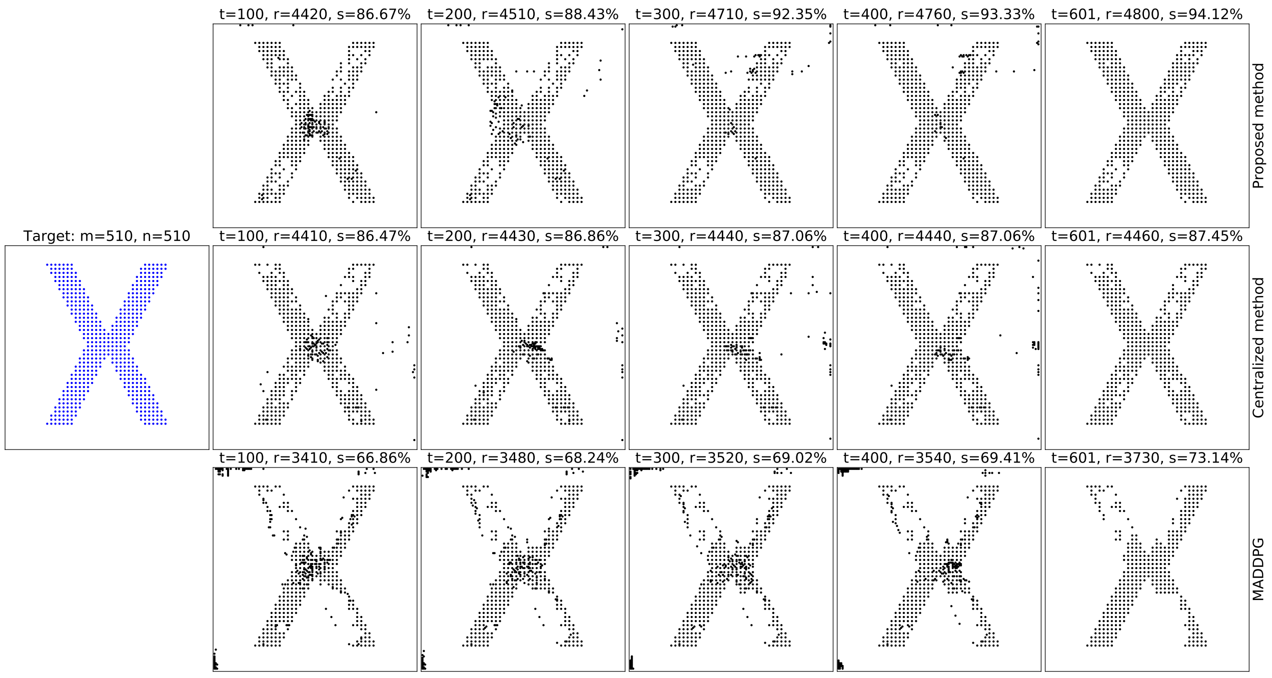


Fig. 9. 550 agents try to organize themselves into an X shape in a 150×150 grid-graph.

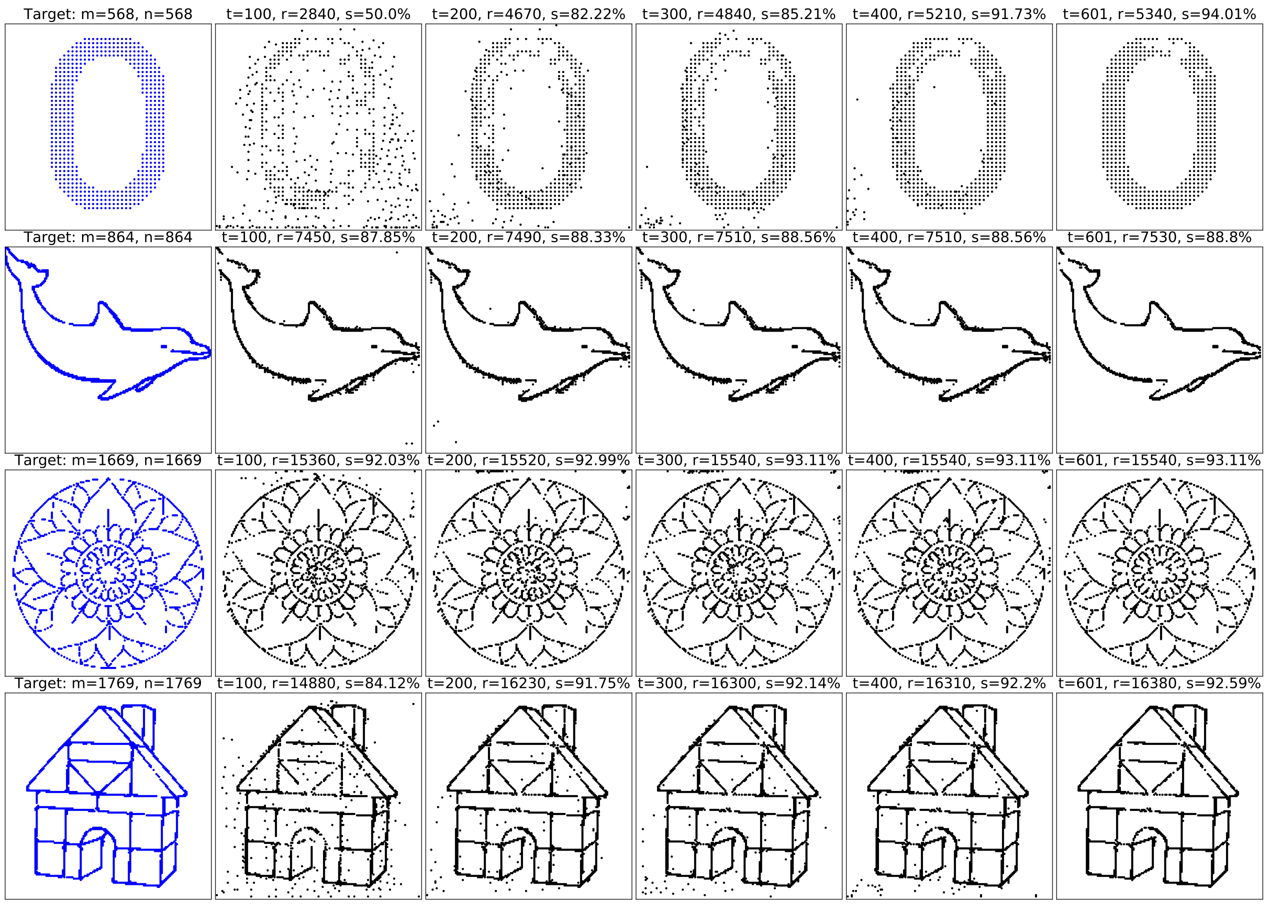


Fig. 10. n agents trying to organize themselves into different shapes in a 150×150 grid-graph.