AFRL-AFOSR-JP-TR-2023-0052

Efficient and Fair Decentralized Task Allocation Algorithms for Autonomous Vehicles: A Machine Learning Based Approach

Qiu, Qinru
SYRACUSE UNIVERSITY
900 S CROUSE AVE STE 620
SYRACUSE, NY, 13244
USA

01/04/2023
Final Technical Report

Air Force Research Laboratory
Air Force Office of Scientific Research
Asian Office of Aerospace Research and Development
Unit 45002, APO AP 96338-5002

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
20230104	Final	START DATE 20200928	END DATE 20220927

4. TITLE AND SUBTITLE
Efficient and Fair Decentralized Task Allocation Algorithms for Autonomous Vehicles: A Machine Learning Based Approach

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
	FA2386-20-1-4062	61102F
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Qinru Qiu

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
SYRACUSE UNIVERSITY 900 S CROUSE AVE STE 620 SYRACUSE, NY 13244 USA	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AOARD UNIT 45002 APO AP 96338-5002	AFRL/AFOSR IOA	AFRL-AFOSR-JP-TR-2023-0052

12. DISTRIBUTION/AVAILABILITY STATEMENT
A Distribution Unlimited: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The project was very successful in both scientific accomplishments, and in strengthening a productive collaboration between US and Australian researchers.
One research paper has been submitted to 2023 IEEE International Conference on Robotics and Automation (ICRA) and is included as supplemental information. The group is also submitting an abstract to the 2023 workshop on Naval Applications of Machine Learning. A proposal based on their research results was submitted to NSF RINGS (Resilient and Intelligent Next-Generation Systems) program and was selected for funding.
There are also two videos about the research results at the following YouTube links
https://youtu.be/d4dRHNA5HTc
https://youtu.be/H_SYPXkBpWQ
From the PI (Qinru Qiu): "We were very excited about the performance of the BAMS based multi-agent systems. Although this project has ended, we are going to further investigate their potentials and limitations. We were recently selected by the NSF Resilient & Intelligent NextG Systems (RINGS) program and part of our proposed research is the multi-agent message passing system for cooperative sensing and navigation"

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	SAR	6

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
GEOFFREY ANDERSEN	

Efficient and Fair Decentralized Task Allocation Algorithms for Autonomous Vehicles -- Technical Report

1. Research Objectives and main contributions

The objective of this project is to improve the efficiency of the multi-agent decentralized mission coordination with an inter-agent communication infrastructure. In phase 1 of this project, we explored the enhancement of the Consensus-Based Bundle Algorithm (CBBA) for distributed task allocation with budget constraints. The limitation of the CBBA technique is that the environment must be known a priori to all agents and tasks must be clearly defined with known costs and rewards. This technique is obviously not suitable for cooperative missions in an unknown environment where agents must explore and improvise their actions together. In phase 2 of this project, we study cooperation techniques for missions in unknown environment where agents have only partial observations. The study uses multi-agent predator and prey game as a platform. The goal is for the agents to jointly locate and capture the prey. The agents have no prior knowledge of the environment or the prey's escape algorithm. They communicate with each other to obtain environment information beyond their own local observations. Based on their local understanding of the environment, the agents choose their own action, which includes where to move and whether to communicate with other agents, to maximize the team rewards. Reinforcement learning is applied to optimize the agent's policy such that the game is completed with the fewest steps.

The main contribution of our phase 2 research is the belief-map assisted multi-agent system (BAMS). A belief map represents the hidden state of the environment maintained by the agent after fusing the incoming messages. By integrating the belief map with the reinforcement learning framework and providing feedback to the belief map, we accelerate the training and improve the rewards that the system can receive. The performance of BAMS is evaluated using a cooperative predator and prey game in an environment of different levels of complexity. The BAMS provides the following benefits compared to existing multi-agent models with message passing capability.

1) The training converges 68% faster and the agents trained using BAMS model completes the game with 27.5% fewer steps.
2) It has robust performance. During the application mode, the number of agents does not have to be the same as the training environment.
3) The messages between agents are encrypted. The messages in BAMS are vectors of learned representations of the agent's belief of the environment. They do not only contain information about current but also future states of agents and environment. Each number does not have correspondence to any physical attribute of the agent or environment. Unless one has the trained model of BAMS, it is not possible to decode the information.
4) Agents reach tacit agreement during the training. From the experimental results, it seems the agents trained using BAMS understand each other's intention without explicit communication.
5) The decoded belief map provides a rough interpretation for the agent's decision. A belief map decoder is trained together with the policy network in BAMS. By comparing the belief map with the actual map, the system receives additional channel of feedback, which supervises the training process. During execution, the belief map provides a way to interpret the agent's hidden state, which can further be used to explain the agent's behavior.

2. Research Accomplishments

2.0. Implementation and training of the BAMS Model

Training of multi-agent reinforcement learning (MARL) is challenging because agents have only partial observations of the environment, and their decisions are not visible to each other. The unknown movements of other agents reduce the environment predictability and make the environment non-stationary. Message exchanges among agents can provide information beyond local observation. However, when to communicate, what to communicate and how to leverage the received messages are variables that need to be optimized. The message passing system and message format should not

be manually designed. What features are needed by the policy network of reinforcement learning are not known in prior. A manually designed message format usually does not give the best result.

Things get complicated when a trainable message passing network is used to connect agents. The messages are additional partial observations that help agents to gain insight of each other. However, the additional trainable variables in the message network significantly increases the model complexity, prolongs the training time, and escalate the chance of overfitting.

Training a complex deep neural network using reinforcement learning is time consuming because the only feedback, i.e., the reward, is delayed, sparse and indirect. We accelerate the DRL by introducing another feedback channel that helps to learn a more efficient message passing network and a more effective representation of the environment. This consequently leads to better policy and faster convergence. In our belief-map assisted multi-agent system (BAMS), each agent is supplemented with a map decoder. It transforms its hidden state into a belief map, a symbolic representation of the agent's knowledge of the global environment. This symbolic representation is simple, and its corresponding ground truth value can easily be obtained. By comparing the belief map with the actual map, the system receives additional channel of feedback, which supervises the training process. During execution, the belief map provides a way to interpret the agent's hidden state, which can further be used to explain the agent's behavior.

To help the agents coordinate better in team and retrieve information from messages more efficiently, gating and attention are also integrated into the message passing system. The attention model helps agents to differentiate important and irrelevant messages, while the gating removes the redundancy and saves communication power and bandwidth.

Details of the BAMS are shown in Figure 1. For the *i*th agent, the model consists of the following four major components:

- Observation Encoder$E_i()$: The observation encoder extracts key features from agent's local observation, which will later be combined with received messages and be used to update the hidden states.
- Message Attention Module $A_i()$: The attention module assigns weights to different messages to filter relevant information based on the hidden state of the local agent.
- Map Decoder $D_i()$: The decoder reconstructs a belief map of the environment at symbolic level based on the hidden state of the local agent. The belief map represents agent's local knowledge of the global environment. It will be compared with the ground truth to provide additional feedback to assist the training of the observation encoder, the hidden state generator, and the message attention module.
- Policy Network $p_i()$: The policy network is an actor-critic model that selects the best action for the local agent to maximize the overall system utility. In BAMS, the action consists of two parts, a discrete movement action and a binary communication action. The former decides how agent moves to complete the game and the latter decides whether the agent should broadcast its local information to connected neighbors.
- Hidden State Generator $lstm_i()$: The hidden state generator is a Long short-term memory (LSTM) that fuses feature from local observation and received messages into a vector. This vector represents the agent's knowledge about the current state of the global environment.

All five components in the BAMS are trained together. The training of BAMS is to help the agents to learn when to communicate with others and how to efficiently utilize the received messages and local observation to make the best decision to maximize the system reward and at the same time reconstruct the belief map of the environment. The loss function is calculated as the weighted sum of three components, value loss, action loss and map loss. The value and action losses are the loss of the critic network and actor network respectively. And the map loss is calculated as $Map\ Loss = MSE\ (\mathbb{G}_i^t - b_i^t)$, where $\mathbb{G}_i^t$ denotes the ground truth map for agent *i* at time *t*, and b_i^t is the predicted map given by map decoder $D_i()$.The entropy regularization coefficient for value loss and map loss is 0.01. For more detailed information of the BAMS model please refer to our paper [3] submitted to ICRA attached to this report.

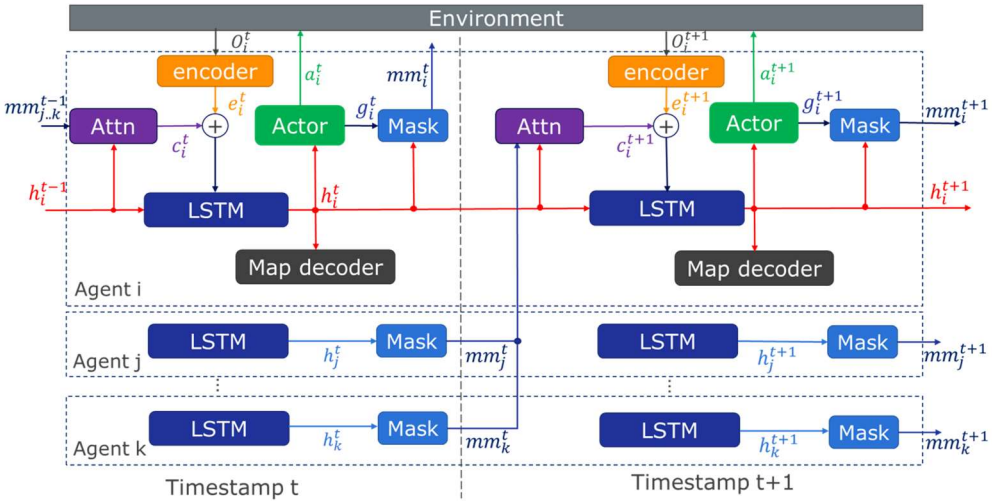

Figure 1 Architecture of BAMS model

2.1. Setup experiments to evaluate BAMS in a predator-prey game

We utilized the classic grid-based predator-prey environment for our experiment and evaluation. In this task, there are *N* predators (agents) with limited vision *v* to explore the *mxm* sized grid environment with a single prey. The value of *N* varies from 3 to 10, and *m* varies from 7 to 20 to represent games with different complexity. The environments are further divided into 2 categories, with obstacles and without obstacles. For environments with obstacles, *K* obstacles are randomly placed, with *K* varies from 10 to 30. This corresponding to 7% to 20% of the total environment space.

The agents have limited observation range. An agent can only observe things in the 3x3 area centered by itself. Agent can take 5 different actions (up, down, left, right, and stay) at every timestep. They also make a binary decision every timestep on whether to broadcast their own hidden state. All agents (predators) have a maximum step limitation, which varies with the environment size.

We compared BAMS with 2 reference models based on previous work. They are: Modified TarMAC (M-TarMAC) and IC3Net. All three models leverage a message passing system among agents and their agent action spaces are the same. The differences of the three models are summarized in the following table:

	Message passing	Gated message	Attention to the message	Decoded belief map
IC3Net [1]	Yes	Yes	No	No
M-TarMAC [2]	Yes	Yes	Yes	No
BAMS	Yes	Yes	Yes	Yes

The main difference between BAMS and other two models is the addition of the map decoder and the consideration of the map loss in the training. The additional channel of feedback during training will make sure that after combining the received message with the local observation and the local hidden state, the agent is able to reconstructor a map that correctly reflects the current environment. This step helps agents to reach consensus about message encoding and interpretation, and consequently lead to a more effective communication network.

In addition to M-TarMAC and IC3Net, we introduced a heuristic rule-based algorithm as the baseline. The heuristic algorithm will guide the agents to explore the map from left to right, and top to bottom. Whenever the agent finish exploring a row, they will move to the next row beyond their observation range. When they reach the margin of map, they will turn around and explore in the opposite direction until they reach the step limit or observed the prey. Once one agent observed the prey, it will send out the location to all other agents. And all other agents will take the shortest path to catch the prey.

2.2. Performance evaluation of BAMS with static prey

We first test the performance of BAMS in the environments where prey does not move. With the static prey, to ensure cooperation among agents, the game will not complete until all agents reached the prey. Table I compares the BAMS with three reference algorithms when environment size varies from 7x7 to 20x20. It shows that our approach takes fewer steps in average to complete the gate compared with the references. When the environment size is 7×7 with 3 agents, the improvement of BAMS is 34.05% compared with M-TarMAC and IC3Net. As the environment becomes more complicated, the performance of IC3Net and M-TarMAC degrades because of slow and difficult convergence in the training. The BAMS gives 21.07% and 34.62% improvements compared to IC3Net and M-TarMAC respectively.

TABLE I. AVERAGE STEPS TAKEN AND COMMUNICATION RATE FOR TESTING UNDER SIMPLE ENVIRONMENTS

	N=3, *m*=7, max steps = 20		*N*=5 *m*=12, max steps = 40		*N*=10, *m*=20, max steps = 80	
	Avg steps	Comm rate	Avg steps	Comm rate	Avg steps	Comm rate
Heuristic	14.56	-	33.24	-	68.9	-
IC3Net	12.48	0.6	32.9	0.39	73.82	0.6
M-TarMAC	12.39	0.32	29.8	0.04	71.76	0.35
BAMS	**8.17**	1	**21.64**	0.27	**56.46**	0.05

As seen from Figure 2, the BAMS has much faster convergence than the other two works. This indicates that, with the help of additional feedback from the belief map, the relationship between the hidden state, action, and reward is more consistent, hence it can be learned with fewer iterations. In other words, the supervised loss that we obtained by comparing the belief map with the ground truth map helps the model to learn a better representation of the global environment and helps to tune the message passing network to be more effective. We need to point out again, the feedback is only needed during the training. During the execution, no ground truth map is available, the only purpose of the belief map is to provide an interpretation of the agent's decision. The map decoder $D_i()$ is not required for the agent to play the game.

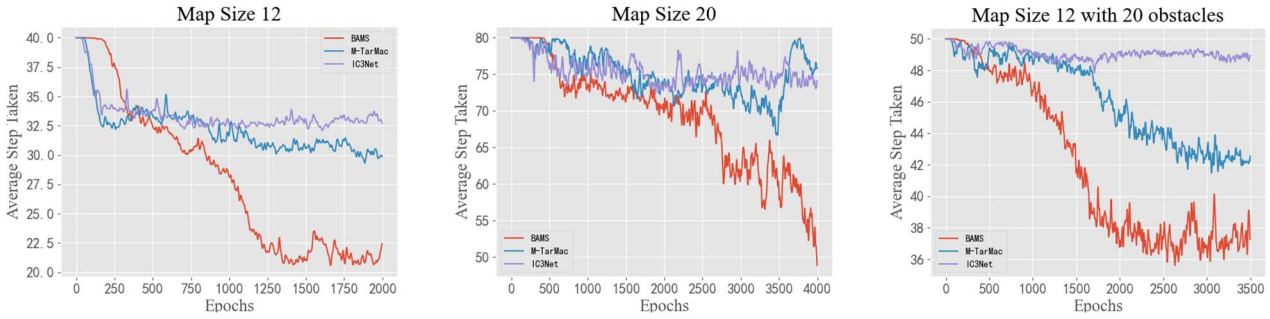

Figure 2 Convergence Speed Comparison (a)Simple_12x12 (b)Simple_20x20 (c)Complex_12x12

We further tested the model using testing environments with 10, 20 and 30 obstacles. We found that, even though the network is trained with 20 obstacles, it can handle different environments. The performances of the three deep learning models in complex environment are shown in Table II. In average BAMS reduces the number of steps by 23.6% and 16.5% compared to IC3Net and M-TarMAC, respectively.

TABLE II. AVERAGE STEPS TAKEN AND COMMUNICATION RATE FOR TESTING UNDER COMPLEX ENVIRONMENTS

No. of obstacles	10		20		30	
	Avg steps	Comm rate	Avg steps	Comm rate	Avg steps	Comm rate
IC3Net	45.39	0.53	48.56	0.54	49.37	0.57
M- TarMAC	39.43	0.062	44.78	0.073	46.92	0.076
BAMS	**31.8**	0.056	**36.51**	0.065	**41.42**	0.054

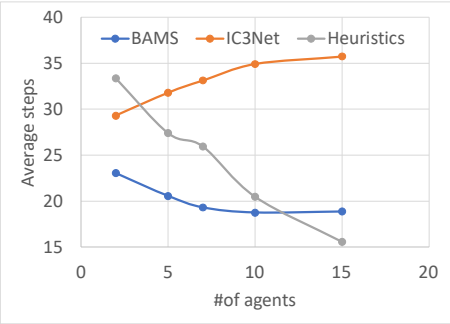

Figure 3 BAMS trained using 5 agents can handle environment where agent population ranges from 2 to 15.

To test the robustness of the trained policy, we train the BAMS and IC3Net model in an environment with 5 agents and test them in different environments with agent number varying from 2 to 15. As shown in Figure 3, the average number of steps that BAMS takes to complete the game reduces with the number of agent increases as we expected. However, this is not the case for IC3Net. For IC3Net, when the number of agents increases, due to the increased amount of message, the agents have difficulty in getting an accurate view of the environment. Therefore, the number of steps to complete the game increases.

For more experimental results please refer to the attached paper. An animation of the game can be found at (https://youtu.be/d4dRHNA5HTc). It shows the trajectory of agents and the decode belief map.

2.3. Performance evaluation of BAMS with moving prey

We further tested the performance of BAMS in an environment where prey is capable of escaping. Both the agents and the prey have the same observation range. Once the prey observes an agent or multiple agents, it will move to a nearby location that is the farthest away from the agent(s). To make the game more challenging, in addition to the original 3×3 observation, we also created a test scenario where both the agents and prey have 5×5 observation range. In a moving prey scenario, it usually requires multiple agents to round up the prey in order to capture it. A heuristic algorithm is also developed, where the agents first scan the map in a zig-zag manner from top to bottom and then from left to right. If the prey is found, they will chase the prey to the corner to capture it.

TABLE III. MOVING PREY GAME WITH 3×3 OBSERVATION RANGE

Pre-train	Observation size	Success rate	Average steps	Communication rate	Avg. steps of Heuristic alg.
No	3×3	0.31	32.47	0.94	33.78
Yes	3×3	0.85	15.13	0.07	33.78
Yes	5 ×5	0.92	17.42	0	29.54

The average performance of our agents is summarized in TABLE III. Rows 1 and 2 in the table correspond to environments where agents have 3x3 observation range. Row 1 gives the performance of agents using BAMS model trained directly with moving prey. Only in 31% of the testing cases the agents can successfully capture the prey and the agents actively communicate (94% of the time) with each other. Row 2 gives the performance of agents using BAMS model pretrained with fixed prey and then fine turned with moving prey. The success rate improves from 31% to 85%. The results show that moving prey created a dynamic environment, which is hard for agents to learn to understand each other. Pretrain the agents in a simple environment with fixed prey is more effective. In this environment, the average number of steps to complete the game using the heuristic algorithm is 33.78 steps, which is 4% and 123% more than the agents without and with pretraining.

When the vision size increases to 5x5, with pre-training, the success rate to capture the prey increases to 92% and the average steps to complete the game also increases to 17.42 steps. Overall, the performance is 69.6% better than the heuristic algorithm. We also found that, with moving prey, the agents tend to cooperate without communication. It is our hypothesis that the side channel information, such as the elapsed time,

A video that showcases the intelligence of the trained agent can be found at https://youtu.be/H_SYPXkBpWQ.

3. Dissemination of Research Results

One research paper [3] has been submitted to 2023 IEEE International Conference on Robotics and Automation (ICRA). We are also submitting an abstract to the 2023 workshop on Naval Applications of Machine Learning. A proposal based on our research results was submitted to NSF RINGS (Resilient and Intelligent Next-Generation Systems) program and was selected for funding.

References

[1] Singh, A., Jain, T., and Sukhbaatar, S. (2019). “Learning when to communicate at scale in multiagent cooperative and competitive tasks.” In ICLR, 2019.

[2] Abhishek Das, Th´eophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. (2019). “TarMAC: Targeted multi-agent communication.” In ICML, pages 1538–1546, 2019.

[3] C. Luo, Q. Huang, F. Kong, A. B. Wu, H. Li, and Q. Qiu, “Enhancing Multi-agent Message Passing for Cooperative Games Using Belief Map Assisted Training,” submitted to IEEE International Conference on Robotics and Automation (ICRA).

Enhancing Multi-agent Message Passing for Cooperative Games Using Belief Map Assisted Training

Chen Luo*, Qinwei Huang*, Fanxin Kong, *Member, IEEE*, Alex B. Wu, Helen Li, *Fellow, IEEE*, and Qinru Qiu, *Senior Member, IEEE*

***Abstract*—Multi-agent cooperative games carried out by robots or autonomous vehicles have wide applications in civil, military, and scientific expeditions. Compared to centralized control, distributed decision making, where each agent chooses its own action to maximize the overall system rewards, is sometimes more favorable due to its low complexity and high resilience. A message passing system among agents is an effective way to help them obtain information beyond the observation of their local area and make global optimal decisions. When to communicate, what to communicate and how to leverage the received messages are variables that can significantly impact the agents' ability to make good decisions and should be optimized. Existing works impose no rule in the message passing system and will learn it together with the agent's actions in a reinforcement learning (RL) setting. Due to the nature of RL and the large search space, the training converges slowly. In this work, we present a belief-map assisted multi-agent system (BAMS). A belief map, which represents the hidden state of the environment maintained by the agent after fusing the incoming messages, is integrated into the model. By providing feedback to the belief map, we accelerate the training and improve the rewards that the system can receive. The performance of BAMS is evaluated using a cooperative predator and prey game with maps of different complexity and compared to previous multi-agent models with message passing capability. The simulation result shows that the training of BAMS converges 68% faster and the agents trained using BAMS model completes the game with 27.5% fewer steps.**

Keywords— multi-agent system, multi-agent reinforcement learning (MARL), message passing, partial observation.

I. Introduction

A multi-agent cooperative game features multiple autonomous systems collaborating with each other on a task to jointly maximize the overall utility of the system. From rescue missions where multiple robots are dispatched to search for a missing person, to military applications where multiple UAVs collaborate in surveying of a wide area, to scientific expeditions where rovers jointly explore an unknown terrain, multi-agent cooperative game can be used to model many different applications. When the number of agents increases, centralized monitoring, controlling and searching [1][2] for the optimal behavior of all agents will not be feasible due to the exponential growth of the complexity. It will also increase the vulnerability of the system due to potential single-point failure [3][4]. Distributed control and optimization, where each agent makes its own decision based on local information, is sometimes more desirable. However, its obvious limitation is that the agents only have partial observations of their local area, and hence cannot make global optimal decisions.

Message exchanges among agents can provide global information and help the agents to move out of local optimum. However, when to communicate, what to communicate and how to leverage the received messages are variables that need to be optimized. In a real-world environment, communication usually takes place within the resource constraints. Constant communication will result in a large number of messages, which consumes not only communication energy and bandwidth but also processing power. The messages sending in consecutive cycles by the same agent, or by agents close to each other are likely to be similar. Sending redundant messages again and again is a waste of energy and bandwidth. Frequently communicating every bit of observed information to other agents is not only wasteful but may also undermine the receiver's decision-making ability, as it cannot effectively distinguish valuable information from irrelevant information. The message passing system and message format should not be manually designed. Many multi-agent games are optimized using reinforcement learning (RL) such as actor-critic model. What features are needed by the policy network to select action are not known in prior. A manually designed message format usually does not give the best result. Finally, to save communication energy and to improve the communication security, the high-dimensional observation must be encoded into a low-dimensional message that can only be decoded by agents. Similar to [16][23], in this work, we train the message passing network together with the policy network. The message generation and encoding network evolves with the policy network during the training process.

Deep reinforcement learning (DRL) has received close attentions and extensive studies because of its outstanding performance in control and optimization of autonomous systems [1][6][7]. In the past decade, single-agent reinforcement learning has achieved remarkable success [7] [8]. The research focus has now shifted to the multi-agent reinforcement learning (MARL) to solve problems in challenging domains such as multi-player simulated games [9][10]. In those applications, agents have partial observations of the environment, based on the observation, they select their own actions. Training of MARL is challenging because agents' decisions are not visible to each other. The unknown movements of other agents reduce the environment predictability and make the environment non-stationary.

* Indicates equal contributions.

Chen Luo, Qinwei Huang, Fanxin Kong and Qinru Qiu are with the Department of Electrical Engineering & Computer Science, Syracuse University, Syracuse, NY 13244 USA (e-mail: {cluo05, qhuang18, fkong03, qiqiu}@ syr.edu).

Alex B. Wu is with Fayetteville-Manlius High School. (e-mail: abwu2016@gmail.com).

Helen Li is with the Department of Electrical & Computer Engineering, Duke University, Durham, NC 27708. (e-mail: hai.li@duke.edu).

Things get more complicated when a trainable message passing network is used to connect agents. The messages are additional partial observations that help agents to gain insight of each other. Previous research [11] proved that all agents could share their own information and aim for a common goal through internal communication. However, the additional trainable variables in the message network significantly increases the model complexity, prolongs the training time, and escalate the chance of overfitting.

Training a complex deep neural network using reinforcement learning is time consuming because the only feedback, i.e., the reward, is delayed, sparse and indirect. In this work, we accelerate the DRL by introducing another feedback channel that helps to learn a more efficient message passing network and a more effective representation of the environment. This consequently leads to better policy and faster convergence. In our belief-map assisted multi-agent system (BAMS), each agent is supplemented with a map decoder. It transforms its hidden state into a belief map, a symbolic representation of the agent's knowledge of the global environment. This symbolic representation is simple, and its corresponding ground truth value can easily be obtained. By comparing the belief map with the actual map, the system receives additional channel of feedback, which supervises the training process. During execution, the belief map provides a way to interpret the agent's hidden state, which can further be used to explain the agent's behavior.

To help the agents coordinate better in team and retrieve information from messages more efficiently, gating and attention are also integrated into the message passing system. The attention model helps agents to differentiate important and irrelevant messages, while the gating removes the redundancy and saves communication power and bandwidth.

We evaluated the performance of the BAMS model using a multi-agent predator-prey game with and without obstacles. Centralized training and distributed execution are adopted. The experimental results show that, compared to existing models, the BAMS is more suitable for large-scale environments with complicated landscapes.

The key contributions of this paper are summarized as follows:

- We proposed a belief-map assisted training mechanism that supplements reinforcement learning with supervised information to accelerate the training convergence.
- Proposed a belief-map decoder to reconstruct a symbolic map from the environment embedding to provide additional feedback during the training. The map transforms the hidden state of agents into a human-readable format, which significantly improves the interpretability of the agent's decision-making process.
- Adopt message gating and attention in the message passing system to improve the communication efficiency. The gating is implemented at the sender side to effectively cut the transmission power and communication bandwidth.
- Simulation results show that agents with the enhancements can be trained for effective operation in large and complex environments. The training time reduces by 68% and the overall performance improves by 27.5%.

The rest of the paper is organized as follows. Section II introduces some previous works related to communication in a multi-agent reinforcement learning system. Section III gives the detail of our proposed method including the believe map decoder and attention model. The experimental results are given in Section IV followed by the conclusions in Section V.

II. Motivations and Previous Works

We consider a fully cooperative multi-agent game as a *decentralized partially observable Markov Decision Process* (*DEC-POMDP*) [12]. DEC-POMDP is defined as a tuple $\langle N, S, P, \boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{A}}, \mathrm{Z}, \gamma \rangle$, where N denotes the number of agents; S is a finite state space; $P(s'|s, \boldsymbol{a}): S \times \boldsymbol{\mathcal{A}} \times S \rightarrow [0,1]$ stands for the state transition function; $\boldsymbol{\mathcal{A}} = [\boldsymbol{A_1} \ldots \boldsymbol{A_N}]$ is a finite set of actions, where $\boldsymbol{A_i}$ represents the set of local actions $\boldsymbol{a_i}$ that agent $\boldsymbol{i}$ can take; $\boldsymbol{\mathcal{O}} = [\boldsymbol{O_1} \ldots \boldsymbol{O_N}]$ is a finite set of observations controlled by observation function $\mathrm{Z}: S \times \boldsymbol{\mathcal{A}} \rightarrow \boldsymbol{\mathcal{O}}$; $\boldsymbol{\mathcal{R}}: S \times \boldsymbol{\mathcal{A}} \rightarrow \mathbb{R}$ is the reward function; and $\gamma \in [0,1]$ is the discount factor.

According to the DEC-POMDP model, each agent i takes an action a_i based on its local observation o_i. When all agents applied their actions $[a_0, a_1, \ldots, a_N]$ to the environment, the environment moves to a new state s' and returns a joint reward r. The MARL trains policies $\pi_i(a_i|o_i): \boldsymbol{O_i} \rightarrow \boldsymbol{A_i}, \forall i$, that maximizes the expected discounted reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r^t]$.

Sharing the observation improves the performance and helps agents to learn a better action policy. Efficient communication allows agents to have more information about the global environment and reduces the negative impact of partial observations. Previous works model a multi-agent communication system as a message passing graph neural network [13][14]. Each node in the graph represents an agent and each edge models a communication pathway equipped with message encoding and decoding. Different graph topologies have been studied [15]. Recent works focus on improving the efficiency and reducing the cost of the communication using gated message passing [18], attention [17], and event/memory driven processing [19][20][21].

As the first work on learnable communication, [22] designed a message passing network where message generation only depends on the agent's local observation, local action and received messages. The message encoder is a MLP trained together with the policy network using reinforcement learning. CommNet [25] embeds a centralized communication channel into the network. It improves [22] by maintaining a local hidden state in each agent using a recurrent neural network (RNN). The hidden state is determined by the sequence of local observations and received messages and will be sent to other agents as the communication message. When multiple messages are received, the agent consolidates them by calculating the average. Then as the extension of CommNet, the IC3Net used the long short-term memory (LSTM) to generate hidden states. It introduces a gating mechanism, which is a binary action, to dynamically block or unblock the

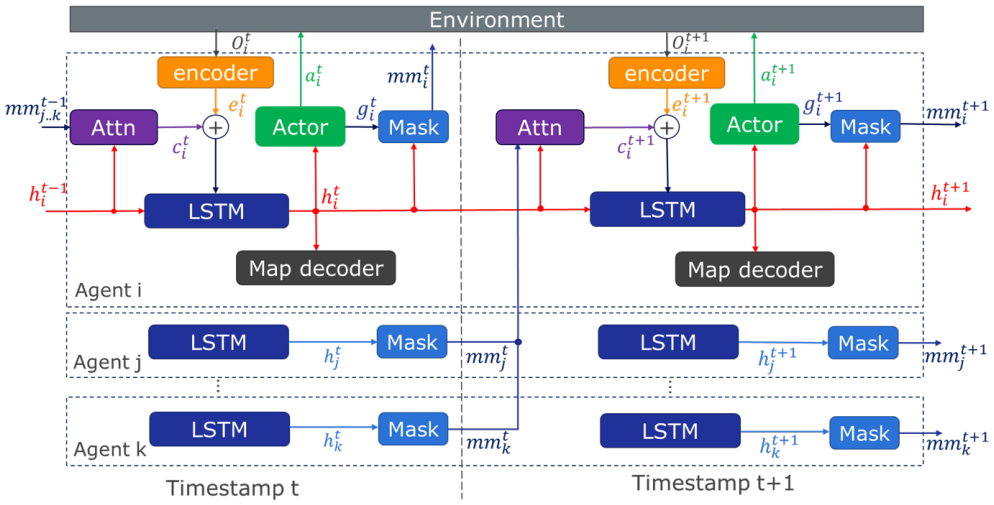

Figure 1 Architecture of BAMS model

message transmission. The policy of communication gating is optimized using reinforcement learning.

Gated-ACML [18], TarMAC[23] and ATOC [26] adopt the gating mechanism and improve the inter-agent communication using an attention model. The attention model prioritizes received messages so that agents can choose useful features. While the Gated-ACML applies gating to prune the message to save the communication bandwidth, ATOC applies attention to determine whom the agent should talk to, and dynamically changes network structure accordingly. TarMAC uses gating and attention to improve the success rate and reward of the game. However, as we will discuss later, their gating is applied at the receiver side, therefore, it does not reduce communication cost and bandwidth. Furthermore, there will be unintentional message leakage under certain conditions.

In this work, we improve the gating mechanism and implement it at the sender's side without information leakage. We also proposed a belief-map assisted training which significantly improves the training speed and quality for large and complex games.

III. Proposed Method

In this section, we present the structure and training of belief-map assisted multi-agent system (BAMS). Details of the BAMS are shown in Figure 1. For the *i*th agent, the model consists of the following four major components:

- Observation Encoder $E_i()$: The observation encoder extracts key features from agent's local observation, which will later be combined with received messages and be used to update the hidden states.
- Message Attention Module $A_i()$: The attention module assigns weights to different messages to filter relevant information based on the hidden state of the local agent.
- Map Decoder $D_i()$: The decoder reconstructs a belief map of the environment at symbolic level based on the hidden state of the local agent. The belief map represents agent's local knowledge of the global environment. It will be compared with the ground truth to provide additional feedback to assist the training of the observation encoder, the hidden state generator, and the message attention module.
- Policy Network $p_i()$: The policy network is an actor-critic model that selects the best action for the local agent to maximize the overall system utility. In BAMS, the action consists of two parts, a discrete movement action and a binary communication action. The former decides how agent moves to complete the game and the latter decides whether the agent should broadcast its local information to connected neighbors.
- Hidden State Generator $lstm_i()$: The hidden state generator is a Long short-term memory (LSTM) that fuses feature from local observation and received messages into a vector. This vector represents the agent's knowledge about the current state of the global environment.

The training of BAMS is to help the agents to learn when to communicate with others and how to efficiently utilize the received messages and local observation to make the best decision to maximize the system reward.

A. Hidden state generation and policy network

At every time step, each BAMS agent collects the observation from their local sensor. The local observation of agent i at time t is denoted as o_i^t . The representation of o_i^t is usually manually designed and is application specific. It directly affects the complexity and the feature extraction ability of the observation encoder $E_i()$. Based on the DEC-POMDP, the agent selects actions based on the hidden state. The hidden state is maintained by a LSTM using local observations and the received messages as the following:

$$h_i^{t+1}, s_i^{t+1} = lstm_i(E_i(o_i^t), c_i^t, h_i^t, s_i^t), \qquad (1)$$

where h_i^t, s_i^t are hidden state and cell state at time t of agent i, and c_i^t is the aggregated feature extracted from the received messages using the attention model: $c_i^t = F_i(A_i(\{m_j^t, j \neq i\})$, $F_i()$ is a linear layer to transform the aggregated message to the communication tensor.

The policy network $p_i()$ is a typical actor-critic model, with an actor network $\Theta_i(h_i^t)$ and a critic network $V_i(h_i^t)$. The $\Theta_i(h_i^t)$ is a one-layer fully connected network. Its input is h_i^t and the output has two components a_i^t and g_i^t,

$$a_i^t, g_i^t = \Theta_i(h_i^t). \quad (2)$$

Vector a_i^t gives the probabilities for the game actions, i.e., the movement that the agent is allowed to take to complete the game. $g_i^t \in [0,1]$ gives the probability for the binary communication action, i.e., block or pass. In each step, the action with the highest probability will be executed.

B. *Message passing model*

Agents send messages to their connected neighbors. Following TarMAC and IC3Net, we use the hidden state as the communication message. The hidden state contains all information that an agent needs to decide local actions. However, not all information is useful to its neighbors. Furthermore, some of the information may overlap with previous messages from the same agent or messages sent by a nearby agent. To improve the efficiency of the communication network, the senders need to reduce the number of redundant messages and the receivers must be able to extract useful information pertinent to its own decision making. We propose to solve these two problems using message gating on the sender side and attention model on the receiver side.

Unlike TarMAC, where the message gating is implemented at the receiver side within the attention model, our gating is implemented at the sender side so that gating can effectively save the transmission power and communication bandwidth. The outgoing message mm_j^t of agent j is calculated as the product of h_j^t and the binary gate action g_j^t.

$$mm_j^t = h_j^t \times g_j^t. \quad (3)$$

Given received messages mm_j^t $(j \neq i)$, agent i aggregates the messages to select the most relevant content using an attention model. The attention model is trained to maximize the reward from the game and minimize the loss of the belief-map construction. Considering the communication delay, agent i uses gated message mm_j^{t-1} send by agent j in previous time step as the input of the key and value networks to generate k_j^t and v_j^t for time t. The query q_i^t of the attention model is generated based on the agent's local hidden state at current time step (h_i^t).

$$k_j^t = key\left(mm_j^{t-1}\right) \quad (4)$$

$$v_j^t = value\left(mm_j^{t-1}\right) \quad (5)$$

$$q_i^t = query(h_i^t) \quad (6)$$

$$\alpha_i^t = softmax\left[\frac{{q_i^t}^T k_1^t}{\sqrt{d_k}} \dots \frac{{q_i^t}^T k_j^t}{\sqrt{d_k}} \dots \frac{{q_i^t}^T k_1^t}{\sqrt{d_k}}\right] \quad (7)$$

$$c_i^t = \sum_{j=1}^{N} \alpha_i^t v_j^t \quad (8)$$

where $key()$, $value()$ *and* $query()$ are networks with one fully connected linear layer, d_k is the number of dimensions of hidden state. c_i^t is the aggregated feature vector that will be used to update the hidden state. Figure 2 shows the structure of the attention model.

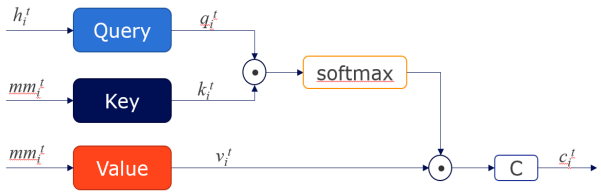

Figure 2 Architecture of the attention model

C. *Believe map generator*

Instead of solely relying on the reward from the environment, additional channels of feedback information could be added to speed up the training. In this work we assist the RL training using a decoded belief map with the multi-agent cooperative predator-prey game in mind. As the aggregation of past observations and incoming messages, the hidden state of an agent represents its knowledge of the environment. It helps the agent to make decisions considering global environment to get more rewards from the game. The more accurate this knowledge is, the better decision the agent will make. However, the agent's hidden state is a feature vector not interpretable. The basic idea of BAMS is to decode the hidden state into a symbolic map that is human interpretable so that the ground truth version of the map can be constructed. By comparing the decoded map with ground truth map, we provide additional feedback to assist the training of the whole system.

The map decoder $D_i(h_i^t)$ can be considered as an inverse process of the observation encoder $E_i(o_i^t)$. Both the observations and decoded maps are $m \times m$ graded planes, where S is the size of the environment. The status of each grid location is coded as a size M vector, where M is the number of possible states of the grid. For example, for the predator-prey game, a grid can have 4 possible states, observed (or unobserved), occupied by a predator, occupied by a prey, occupied by an obstacle. These 4 states are not necessarily exclusive to each other; hence each grid is encoded as a multi-hot vector with size M. Overall, both maps have dimension $M \times m \times m$. The observation map contains only information from local agent, while the belief map should combine the information from all agents.

In this work, we consider centralized training and distributed execution. During training, the central controller generates ground truth map for each agent by keep tracking the movement of all agents. A grid is observed if it has been observed by any agent. For any grid that has not been observed, its other status will also be false.

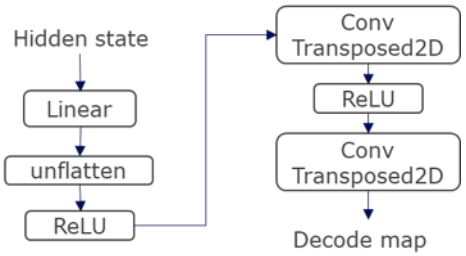

Figure 3 Architecture of belief map model

The map decoder generates the believe map b_i^t for agent i at timestamp t. As shown in Figure 3, the model has a linear layer and two transposed convolutional layers with ReLU activation. Let $\mathbb{G}_i^t$ denote the ground truth map for agent i at time t, Mean Squared Error (MSE) is used to calculate the map loss:

$$Map\ Loss\ =\ MSE\ (\mathbb{G}_i^t\ -\ b_i^t) \tag{9}$$

$$b_i^t\ =\ decoder(h_i^t) \tag{10}$$

The map loss can be obtained every time step. Minimizing the map loss can help all agents converge to an effective communication protocol and efficient message processing. All five components in the BAMS are trained together.

IV. Experiments

We utilized the classic grid-based predator-prey environment [13] for our experiment and evaluation. In this task, there are N predators (agents) with limited vision v to explore the $m*m$ sized grid environment and chase a fixed location prey. The value of N varies from 3 to 10, and m varies from 7 to 20 to represent games with different complexity. The environments are further divided into 2 categories, with obstacles and without obstacles. For environments with obstacles, K obstacles are randomly placed, with K varies from 10 to 30. This corresponding to 7% to 20% of the total environment space.

A. Experiment setting

We trained our network with a batched synchronous multi-agent Actor-Critic model, using RMSprop with a learning rate of 0.001 and $\alpha = 0.97$. The loss function is the weighted sum of three components, value loss, action loss and map loss. The entropy regularization coefficient ε for value loss and map loss is 0.01. The hidden state size for LSTM is 64. For the attention model, the key (k_j^t) and query (q_j^t) have dimension 16 and the dimension of value (v_j^t) is 64. All agents use the same BAMS model. the parameters for the BAMS model are shared among all agents.

The agents have limited observation range. An agent can only observe things in the 3x3 area centered by itself. Agent can take 5 different actions (up, down, left, right, and stay) at every timestamp. All agents (predators) have a maximum step limitation, which varies with the environment size. Before an agent reaches the prey, it will receive a penalty $r_{searching} = -0.05$ in each time step. To avoid this penalty, the agent must learn to reach the prey as early as possible. Once an agent arrives at the prey, it will stay on the prey and receives no more penalty. If the agents reach the limit of maximum number of steps, they will stop and mark this case as failure.

We select the best model during training based on the average steps taken for agents to reach the target. The best model will be used for testing. Environment setting for testing is the same as training in terms of the number of predators and prey, the action space, and the map size.

We compared BAMS with 2 reference models based on previous work. They are: Modified TarMAC (M-TarMAC) and IC3Net. As we mentioned earlier, the original TarMAC gates the messages at the receiver side after the attention layer. Furthermore, when all senders decided to gate their outgoing messages, the receiver will generate an aggregated message by paying equal attention to the received messages. In other words, there is a message leak that let the receiver to still get information even though all senders choose to gate their message. In this work we fixed the message leak and move the gating to the sender side for a fair comparison. Compared to BAMS, M-TarMAC has similar gating and attention mechanism, however, it does not have the additional channel of feedback from the belief maps. IC3Net adopts the same gating mechanism however, the receiver generates the aggregated message simply by averaging the input messages without attention. Neither does it have the additional feedback channel as BAMS has.

TABLE I. AVERAGE STEPS TAKEN AND COMMUNICATION RATE FOR TESTING UNDER SIMPLE ENVIRONMENTS

	N=3, m=7, max steps = 20		N=5 m=12, max steps = 40		N=10, m=20, max steps = 80	
	Avg steps	Comm rate	Avg steps	Comm rate	Avg steps	Comm rate
Heuristic	14.56	-	33.24	-	68.9	-
IC3Net	12.48	0.6	32.9	0.39	73.82	0.6
M-TarMAC	12.39	0.32	29.8	0.04	71.76	0.35
BAMS	**8.17**	1	**21.64**	0.27	**56.46**	0.05

Moreover, we introduced a heuristic rule-based algorithm as the baseline. The heuristic algorithm will guide the agents to explore the map from left to right, and top to bottom. Whenever the agent finish exploring a row, they will move to the next row beyond their observation range. When they reach the margin of map, they will turn around and explore in the opposite direction until they reach the step limit or observed the prey. Once one agent observed the prey, it will send out the location to all other agents. And all other agents will take the shortest path to catch the prey.

B. Experiments Results for simple environment

The simple environment does not have any obstacles. Predators can move towards any direction within the map. We utilize the average steps taken to represent the overall performance of the models. Under the same circumstance, lower average steps taken represents better performance. TABLE I compares the BAMS with three reference algorithms using games with different size. It shows that our approach takes fewer steps in average to complete the gate compared with the references. When the environment size is 7×7 with 3 agents, the improvement of BAMS is 34.05% compared with M-TarMAC and IC3Net. As the environment becomes more complicated, the performance of IC3Net and M-TarMAC degrades because of slow and difficult convergence in the training. The BAMS gives 21.07% and 34.62% improvements compared to IC3Net and M-TarMAC respectively.

As seen from Figure 4, our approach has much faster convergence than the other two works. This indicates that, with the help of additional feedback from the belief map, the relationship between the hidden state, action, and reward is more consistent, hence it can be learned with fewer iterations. In other words, the supervised loss that we obtained by comparing the belief map with the ground truth map helps the model to learn a better representation of the global environment and helps to tune the message passing network to

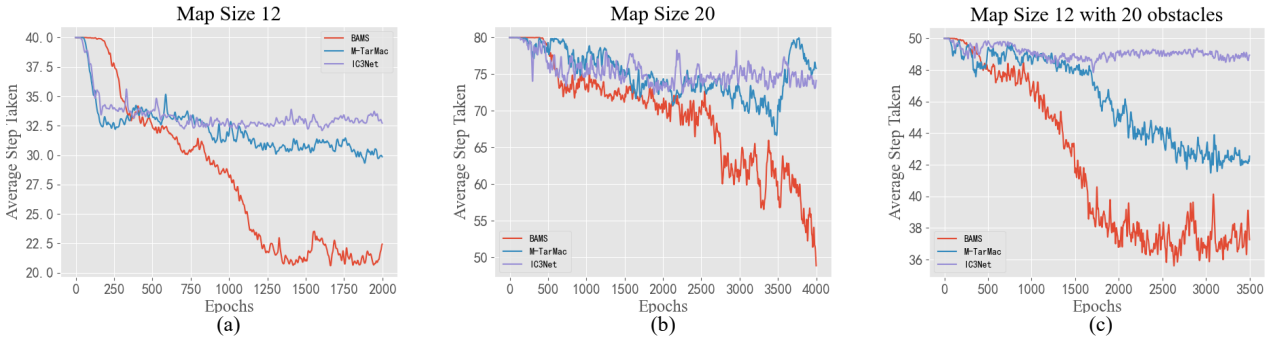

Figure 4 Average Step Taken Comparison (a)Simple_12x12 (b)Simple_20x20 (c)Complex_12x12

be more effective. We need to point out again, the feedback is only needed during the training. During the execution, no ground truth map is available, the only purpose of the belief map is to provide an interpretation of the agent's decision. The map decoder $D_i()$ is not required for the agent to play the game.

C. *Experiments Results for complex environment*

Next, we tested our approach under a complex environment with obstacles. Each grid in the environment is encoded as multi-hot vector of size 4, which represents whether the grid is occupied by a predator, a prey or an obstacle, and if the grid has been observed. We fixed the environment size to be 12×12. Each randomly generated training environment has 20 randomly placed obstacles.

Figure 4(c) compares the convergence speed BAMS, IC3Net and M-TarMAC. We can see that BAMS again has the fastest convergence speed compared to other models. Furthermore, in average, BAMS completes the game with 3 steps fewer than the other two models. Compared to Figure 4(a), the performance different between M-TarMAC and IC3Net also increases. This means effective message passing network becomes more important under complicated environment. under complex environment.

We also observed that, when the environment gets more complexed, the performance of those models oscillates more significantly. This can be seen in Figure 4(b) and (c) when the environment size is 20 or when obstacles are included. This is because, for randomly generated large (complicated) environment, the complexity of the game varies significantly. Things such as the initial location of the agents and the way the obstacles are distributed will affects the number of steps needed to complete the game.

We further tested the model using testing environments with 10, 20 and 30 obstacles. We found that, even though the network is trained with 20 obstacles, it can handle different environments. The performance of the three deep learning models in complex environment is shown in TABLE II. In average BAMS reduces the number of steps by 23.6% and 16.5% compared to IC3Net and M-TarMAC, respectively.

TABLE II. AVERAGE STEPS TAKEN AND COMMUNICATION RATE FOR TESTING UNDER COMPLEX ENVIRONMENTS

No. of obstacles	10		20		30	
	Avg steps	Comm rate	Avg steps	Comm rate	Avg steps	Comm rate
IC3Net	45.39	0.53	48.56	0.54	49.37	0.57
M-TarMAC	39.43	0.062	44.78	0.073	46.92	0.076
BAMS	**31.8**	0.056	**36.51**	0.065	**41.42**	0.054

TABLE I and TABLE II also compared the communication rate among 3 deep learning models. This is the percentage of time when a message is not gated. The data show that BAMS completes the game much faster without significantly increasing the communication cost. We also observed that when the map is relatively small and simple, BAMS agents tend to communicate more frequently. This perhaps is because part of their goals is to reconstruct the map. When agents accrued enough information about the environment, they will share it with others. We will investigate more about this behavior in our future works.

V. CONCLUSION

We proposed a novel belief map assisted training to improve the convergence and efficiency of multi-agent cooperative game with distributed decision. Attention-based inter-agent communication is adopted to overcome the partial observation. The agent will learn when to gate the message to save bandwidth and avoid interfering each other with irrelevant (or redundant) information. We compared our approach with IC3Net and TarMAC in simple and complex predator-prey environments. The experimental results show that our attention-based belief map can help the agent learn a better representation of the environment hidden state and effective message processing, which will lead to wiser decision. We also show that the belief map assisted training improves convergence speed and reduces average steps needed to complete the game.

Our future work is to find better structures of the network and investigate the communication behavior of the agents. We will also extend the BAMS model to other multi-agent applications with asynchronous interface, limited bandwidth, etc.

ACKNOWLEDGMENT

This work is partially supported by the National Science Foundation under grant CNS-2148253 and Air Force Office of Scientific Research under grant FA2386-20-1-4062.

REFERENCES

[1] Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., & Mordatch, I. (2017). "Multi-agent actor-critic for mixed cooperative-competitive environments." Advances in neural information processing systems, 30.

[2] Huang, R., Chu, X., Zhang, J., & Hu, Y. H. (2015). "Energy-efficient monitoring in software defined wireless sensor networks using reinforcement learning: A prototype." International Journal of Distributed Sensor Networks, 11(10), 360428.

[3] Lynch, G. S. (2009). "Single point of failure: The 10 essential laws of supply chain risk management." John Wiley and Sons.

[4] Moradi, M. (2016). "A centralized reinforcement learning method for multi-agent job scheduling in Grid." In 2016 6th International Conference on Computer and Knowledge Engineering (ICCKE) (pp. 171-176). IEEE.

[5] Isele D, Rahimi R, Cosgun A, et al. (2018). "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning." In IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 2034-2039.

[6] Guillaume Lample and Devendra Singh Chaplot. (2017). "Playing fps games with deep reinforcement learning." In AAAI Conference on Artificial Intelligence (AAAI), 2017

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. (2015). "Human-level control through deep reinforcement learning." Nature, 518:529–533, 2015.

[8] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. (2016). "Mastering the game of go with deep neural networks and tree search." Nature, 529(7587):484–489, January 2016.

[9] Vinyals O, Babuschkin I, Czarnecki W M, et al. (2019). "Grandmaster level in StarCraft II using multi-agent reinforcement learning." Nature, 2019, 575(7782): 350-354.

[10] Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., ... & Zhang, S. (2019). "Dota 2 with large scale deep reinforcement learning." arXiv preprint arXiv:1912.06680.

[11] D. Szer and F. Charpillet, (2004). "Improving coordination with communication in multi-agent reinforcement learning," in Proc. 16th IEEE Int. Conf. Tools Artif. Intell., Nov. 2004, pp. 436–440

[12] Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. (2002). "The complexity of decentralized control of markov decision processes." Mathematics of operations research 27(4):819–840.

[13] Liu, Y., Wang, W., Hu, Y., Hao, J., Chen, X., & Gao, Y. (2020). "Multi-agent game abstraction via graph attention neural network." In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 05, pp. 7211-7218).

[14] Li, Q., Gama, F., Ribeiro, A., & Prorok, A. (2020). "Graph neural networks for decentralized multi-robot path planning." In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 11785-11792). IEEE.

[15] Sheng, Junjie, et al. (2022). "Learning structured communication for multi-agent reinforcement learning." Autonomous Agents and Multi-Agent Systems 36.2 (2022): 1-31.

[16] Singh, A., Jain, T., and Sukhbaatar, S. (2019). "Learning when to communicate at scale in multiagent cooperative and competitive tasks." In ICLR, 2019.

[17] M. Geng, K. Xu, X. Zhou, B. Ding, H. Wang, and L. Zhang, (2019). "Learning to cooperate via an attention-based communication neural network in decentralized multi-robot exploration," Entropy, vol. 21, no. 3, p. 294, Mar. 2019.

[18] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, and Y. Ni. (2020). "Learning agent communication under limited bandwidth by message pruning," in Proc. AAAI Conf. Artif. Intell. (AAAI), 2020, vol. 34, no. 4, pp. 5142–5149.

[19] E. Pesce and G. Montana. (2020). "Improving coordination in small-scale multiagent deep reinforcement learning through memory-driven communication," Mach. Learn., vol. 109, pp. 1727–1747, Jan. 2020.

[20] Hu, Guangzheng, et al. (2021). "Event-Triggered Communication Network With Limited-Bandwidth Constraint for Multi-Agent Reinforcement Learning." IEEE Transactions on Neural Networks and Learning Systems (2021).

[21] D. Simões, N. Lau, and L. P. Reis. (2020). "Multi agent deep learning with cooperative communication," J. Artif. Intell. Soft Comput. Res., vol. 10, no. 3, pp. 189–207, Jul. 2020.

[22] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. (2016). "Learning to communicate with deep multi-agent reinforcement learning." In Advances in Neural Information Processing Systems (NIPS), 2016.

[23] Abhishek Das, Th´eophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. (2019). "TarMAC: Targeted multi-agent communication." In ICML, pages 1538–1546, 2019.

[24] Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. (2019). "Learning to schedule communication in multi-agent reinforcement learning." In ICLR, 2019.

[25] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. (2016). "Learning multiagent communication with backpropagation." In Advances in Neural Information Processing Systems (NIPS), 2016.

[26] J. Jiang and Z. Lu. (2018). "Learning attentional communication for multi-agent cooperation," In Advances in Neural Information Processing Systems (NIPS), 2018, pp. 7254–7264.