Importance-Aware Message Exchange and Prediction for Multi-Agent Reinforcement Learning

Xiufeng Huang, Sheng Zhou
Beijing National Research Center for Information Science and Technology,
Department of Electronic Engineering, Tsinghua University, Beijing, China
Email: huangxf18@mails.tsinghua.edu.cn, sheng.zhou@tsinghua.edu.cn

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001408

***Abstract*—The cooperation among intelligent agents is the key to build smarter real-world intelligent systems. Recent researches have shown that wireless communication plays a vital role in multi-agent cooperation. However, limited wireless resources become the bottleneck of large-scale multi-agent cooperation. Therefore, we focus on how to improve the performance of multi-agent reinforcement learning under the constraint of communication resources. To share more valuable information among agents under resource constraint, we formulate the message importance and design a decentralized scheduling policy with query mechanism, so that agents can effectively exchange messages according to their message importance. We further design a message prediction mechanism to compensate for those messages that are not scheduled for transmission in the current round. Finally, we evaluate the performance of the proposed schemes in a traffic junction environment, where only a fraction of agents can broadcast their messages in each round due to limited wireless resources. Results show that the importance-aware message exchange can extract valuable information to guarantee the system performance even when less than half of agents can share their states. By exploiting message prediction, the system can further save 40% of communication resources while guaranteeing the system performance.**

I. Introduction

Recently, reinforcement learning has made great progress combined with deep learning and provided many intelligent services, such as vehicular network [1], robotics control [2] and game playing [3]. To enable these intelligent services in real-world environments, one key is the cooperation among intelligent agents and thus many researches [4] [5] have focused on multi-agent reinforcement learning (MARL). Compared with single-agent reinforcement learning, MARL is faced with partial observability and the loss of stationarity of the environment, and thus independent processing of every agent without cooperation can hardly perform well [6]. Therefore, communication becomes a significant part of MARL systems. With communication, agents can share the information of their observations and decisions [7] [8]. However, with the explosive development of intelligent devices [9], the wireless resources for intelligent systems inevitably become scarce and thus communication becomes the bottleneck of large-scale MARL systems.

In this paper, we focus on how to improve the performance of MARL systems with limited communication resources. Under the constraint of bandwidth, only part of agents can send their messages to other agents. Therefore, the first problem is how to schedule agents to exchange their messages. The idea is to select the most important messages to share and thus the problem turns to how to evaluate the importance of messages. Recent researches obtain message importance with value function [10], self-attention mechanism [11], attention model [12] or neural-network-generated weight [13]. From our perspective, the message importance depends not only on the messages themselves, but also on the needs of other agents who receive them. Therefore, we design a query mechanism to enable agents to express what they can provide and what they need. With the message importance evaluated by the query mechanism, the agents can select important messages for transmission to save communication resources and perform weighted averaging for the message aggregation. In our work, we define the message importance as their influence on the MARL model output, i.e., more change of the model output after receiving updated messages means larger importance of the messages. Furthermore, we design a decentralized scheduling policy to avoid access collisions.

Although above agent scheduling can select important messages to transmit, there is still information loss because some agents are unable to update their messages. Therefore, we introduce message prediction into the MARL system. It is inspired by the fact that the agents with less message importance have higher probability to be stable in terms of states and decisions. Therefore we can predict the messages of these agents according to their history and the knowledge of environments that is encoded in the reinforcement learning model. As a result, the performance of MARL system becomes robust with respect to the communication resources.

To summarize, we design decentralized importance-aware message exchange and message prediction schemes to optimize the MARL system with limited communication resources. In particular, our contributions include:

- Design decentralized agent scheduling policy with query mechanism, which can avoid the collision during medium access. The proposed query can help select important messages for exchange and perform weighted averaging for message aggregation.
- Propose a definition of message importance as the influence on the model output and derive a simplified formulation of message importance. The proposed message importance is more friendly for calculation and does not

need online training, which makes it suitable for dynamic environments.
- Design message prediction block for the agents that are not scheduled to share messages. With message prediction, the agents can benefit from history messages and the knowledge of environment.

The rest of this paper is organized as follows. In Section II we introduce the system model of the MARL system, including the reinforcement learning model and the message exchange scheme. In Section III we introduce the designed query mechanism. In Section IV we propose the definition of message importance and the derivation of its simplified formulation, with which we design a decentralized scheduling policy. Section V describes the algorithm of message prediction block. Experiments results are provided in Section VI. The paper is concluded in Section VII.

II. System Model

In the MARL system under consideration, there are N agents coordinated to perform a task, which is described as a tuple $\langle N, \boldsymbol{S}, \boldsymbol{A}, r, P, \boldsymbol{O}, \boldsymbol{M}, \boldsymbol{Q}, N_{\mathrm{c}} \rangle$. The element symbols are explained as follows.

$\boldsymbol{S}$ is the state space of the whole environment. $\boldsymbol{A}$ is the action space of all agents. r is the reward of the reinforcement learning task. $P : \boldsymbol{S} \times \boldsymbol{A} \to \boldsymbol{S}$ is the state transition function of the environment. $\boldsymbol{O} = \{\boldsymbol{O}_i\}_{i=1,\dots,N}$ is the space of all agents' observations. For coordination, the agents generate messages according to their observations with the learning model and share the messages through wireless channels. $\boldsymbol{M} = \{\boldsymbol{M}_i\}_{i=1,\dots,N}$ is the space of messages. $\boldsymbol{Q} = \{\boldsymbol{Q}_i\}_{i=1,\dots,N}$ is the space of queries, which will be introduced in Sec. III in detail. Messages and queries are both real-valued vectors generated by neural networks.

The system is time-slotted and every agent takes one action in every time slot. For communication, a time slot is divided into two phases, the query phase and the message phase. In the query phase, the agents share their queries. Because the query has much smaller size, i.e., the number of elements in the vector, than the message, we assume that the resource blocks in the query phase are sufficient for the transmission of all agents' queries. Agents can periodically take the resource blocks to broadcast queries. Next, agents share messages in the message phase. Due to the limitation of wireless resources, the resource blocks in the message phase can be allocated to at most N_{c} agents. We propose a decentralized scheduling policy to select N_{c} agents to take the resource blocks and avoid collision in Sec. IV.

The reinforcement learning model that generates actions, messages and queries is shown in Fig. 1, similar to [8]. Different from [8], we propose to add a message block and a query block, which will be introduced in Sec. III. Take the j-th agent for example shown in Fig. 1, in every time slot, agent j obtains its observation $\boldsymbol{o}_j$ and generates its hidden state $\boldsymbol{h}_j$ with a neural network H. Next, agent j generates message $\boldsymbol{m}_j$ and query $\boldsymbol{q}_j$. All agents share their queries in the query phase. Upon receiving the queries, the agents calculate the message importance of

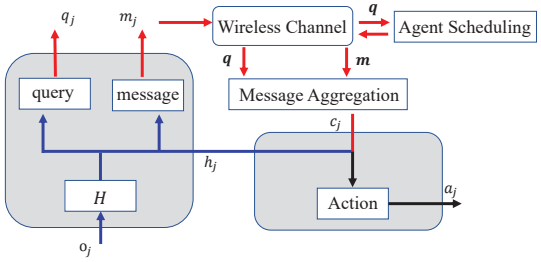

Fig. 1. The MARL model.

every agent, i.e., how these messages can impact all other agents. In the message phase, N_{c} agents that have the largest message importance take the resources blocks in order of their message importance. After receiving updated messages, agent j aggregates these messages by weighted averaging to obtain $\boldsymbol{c}_j$. Finally, agent j generates its action with the hidden unit $\boldsymbol{h}_j$ and aggregated message $\boldsymbol{c}_j$ via the action block. The reinforcement learning model of every agent, including hidden layer H, query block, message block and action block, which are all fully connected layers, is trained with asynchronous advantage actor-critic (A3C) [14]. The system is implemented by centralized training and decentralized execution. The parameters of the learning model are shared among all agents and the training maximizes individual reward, instead of global reward, like IC3Net [15].

III. Query Mechanism

We exploit the model designed in [8], which uses hidden state of deep reinforcement learning model as shared messages, as our basic architecture. In our work, we add a message block and a query block, which are fully connected layers, to improve the performance of the MARL system as shown in Fig. 1.

Firstly, the hidden state $\boldsymbol{h}_j$ does not only encode the information needed for other agents, but also encodes the information needed for making its own decisions. Therefore, sending hidden units will cost excessive communication resources. We therefore add the message block to extract the information needed for sharing among agents and save communication resources.

Next, we focus on the aggregation of messages. In [8], the aggregated message $\boldsymbol{c}_j$ used as model input is the average of all agents' messages, i.e.,

$$\boldsymbol{c}_j = \frac{1}{N-1} \sum_{i \neq j} \boldsymbol{m}_i, \tag{1}$$

where the messages from different agents have the same weight. However, the importance of received messages from agents should be different, depending on their positions, targets or other information that is significant for the reward of the MARL system. Accordingly, we propose the query mechanism for the aggregation of messages, where a block is added to generate query in parallel with the message block. A query represents the needs from the agent who sends it out. If properly trained, larger dot product of the queries from two agents means strong mutual importance, and thus larger weights should be adopted in their

message aggregation processes. Therefore, we propose new message aggregation method along with the query mechanism,

$$\boldsymbol{c}_j = \frac{1}{N-1}\sum_{i\neq j} \boldsymbol{q}_j^{\mathrm{T}} \boldsymbol{q}_i \boldsymbol{m}_i, \tag{2}$$

where $\boldsymbol{q}_i$ is the query vector generated by the i-th agent.

In the proposed MARL system, the designed query has much smaller size than the message, which ensures the system to benefit from broadcasting queries of all agents without too much extra communication overhead. Besides providing weights for message aggregation, broadcasting queries before message sharing can also help schedule agents to transmit their messages according to the information encoded in the queries. The details of how to select important messages are described in the next section.

IV. Decentralized Scheduling Policy

A. Message Importance Measure

One of the key issues of MARL system is the communication bottleneck. In practice, there can be a large number of agents broadcasting their messages in a wireless channel, which may not be feasible in time due to the limited resource blocks. Therefore, it is vital to reduce the communication cost of the system. In our work, we schedule a subset of agents to broadcast their messages in every time slot. Specifically, in one time slot, only N_{c} agents can be allocated with resource blocks to broadcast their message over the wireless channel. The problem is which N_{c} out of all agents should be selected. To solve this problem, we first need to define the importance of message.

If properly trained, the change of the reinforcement learning model output (such as the Q-value, the probability of selecting actions) resulted by the shared messages can improve the performance of the whole system. Therefore, the messages affecting more on the model output should have larger importance. For example, upon receiving message $\boldsymbol{m}_i$, the model output of the j-th agent is $f(\boldsymbol{m}_i; \boldsymbol{\theta}_j, \boldsymbol{o}_j, \boldsymbol{m}^{\mathrm{c}})$, where $\boldsymbol{\theta}_j$ is its model parameters, $\boldsymbol{o}_j$ is its observation and $\boldsymbol{m}^{\mathrm{c}}$ includes the messages of other agents. If agent j does not receive message $\boldsymbol{m}_i$, the message used for model input can be said $\boldsymbol{m}_i'$ (e.g. the history message from agent i) and the model output is $f(\boldsymbol{m}_i'; \boldsymbol{\theta}_j, \boldsymbol{o}_j, \boldsymbol{m}^{\mathrm{c}})$. Then we can obtain the message importance of $\boldsymbol{m}_i$ for agent j as

$$I_j(\boldsymbol{m}_i) = \left\| f(\boldsymbol{m}_i; \boldsymbol{\theta}_j, \boldsymbol{o}_j, \mathbf{m}^{\mathrm{c}}) - f(\boldsymbol{m}_i'; \boldsymbol{\theta}_j, \boldsymbol{o}_j, \mathbf{m}^{\mathrm{c}}) \right\|. \tag{3}$$

In a MARL system that continuously exchanges messages, the message distance, i.e., $\|\boldsymbol{m}_i - \boldsymbol{m}_i'\|$, is small. Therefore the message importance can be approximated by

$$I_j(\boldsymbol{m}_i) \approx \left\| \frac{\partial f}{\partial \boldsymbol{c}}\Big|_{\boldsymbol{c}=\boldsymbol{c}_j} \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{m}}\Big|_{\boldsymbol{m}=\boldsymbol{m}_i} (\boldsymbol{m}_i - \boldsymbol{m}_i') \right\| \tag{4}$$

$$= \left\| \frac{\partial f}{\partial \boldsymbol{c}}\Big|_{\boldsymbol{c}=\boldsymbol{c}_j} \boldsymbol{q}_j^{\mathrm{T}} \boldsymbol{q}_i (\boldsymbol{m}_i - \boldsymbol{m}_i') \right\|, \tag{5}$$

where the calculation of $I_j(\boldsymbol{m}_i)$ needs the information from agent j, i.e., $\frac{\partial f}{\partial \boldsymbol{c}}|_{\boldsymbol{c}=\boldsymbol{c}_j}$, which requires extra communication resources for sharing among agents. To estimate $I_j(\boldsymbol{m}_i)$ without sharing gradient $\frac{\partial f}{\partial \boldsymbol{c}}|_{\boldsymbol{c}=\boldsymbol{c}_j}$, we treat it as a random vector, assuming the uniform distribution of the direction and identical distribution of the magnitude, i.e.,

$$\begin{aligned} &\mathbb{P}\left(\left\| \frac{\partial f}{\partial \boldsymbol{c}}\Big|_{\boldsymbol{c}=\boldsymbol{c}_a} \right\| = g; \boldsymbol{c}_a \right) \\ =&\mathbb{P}\left(\left\| \frac{\partial f}{\partial \boldsymbol{c}}\Big|_{\boldsymbol{c}=\boldsymbol{c}_b} \right\| = g; \boldsymbol{c}_b \right), \forall g \in \mathbb{R}^*, \boldsymbol{c}_a, \boldsymbol{c}_b \in \mathbf{M}. \end{aligned} \tag{6}$$

Then, we obtain the message importance by taking expectation

$$I_j(\boldsymbol{m}_i) \approx \mathbb{E}_{\frac{\partial f}{\partial \boldsymbol{c}}|_{\boldsymbol{c}=\boldsymbol{c}_j}} \left[\left\| \frac{\partial f}{\partial \boldsymbol{c}}\Big|_{\boldsymbol{c}=\boldsymbol{c}_j} \boldsymbol{q}_j^{\mathrm{T}} \boldsymbol{q}_i (\boldsymbol{m}_i - \boldsymbol{m}_i') \right\| \right] \tag{7}$$

$$= \frac{\int_0^{2\pi} |\cos\theta| \mathrm{d}\theta}{2\pi} \int_0^{\infty} g p(g) \|\boldsymbol{q}_j^{\mathrm{T}} \boldsymbol{q}_i (\boldsymbol{m}_i - \boldsymbol{m}_i')\| \mathrm{d}g \tag{8}$$

$$= C \|\boldsymbol{q}_j^{\mathrm{T}} \boldsymbol{q}_i (\boldsymbol{m}_i - \boldsymbol{m}_i')\|, \tag{9}$$

where $p(g) = \mathbb{P}\left(\left\| \frac{\partial f}{\partial \boldsymbol{c}}\Big|_{\boldsymbol{c}=\boldsymbol{c}_j} \right\| = g; \boldsymbol{c}_j \right)$ and $C = \frac{\int_0^{2\pi} |\cos\theta| \mathrm{d}\theta}{2\pi} \int_0^{\infty} g p(g) \mathrm{d}g$. Note that C is a constant independent on j and i. Therefore, to compare the importance of messages, we can simplify $I_j(\boldsymbol{m}_i)$ into

$$I_j(\boldsymbol{m}_i) = \|\boldsymbol{q}_j^{\mathrm{T}} \boldsymbol{q}_i (\boldsymbol{m}_i - \boldsymbol{m}_i')\|. \tag{10}$$

Eq. (10) shows an approximate method for calculating the message importance without extra information sharing of other agents and reveals that the importance depends on the change of messages as well as the weights of message aggregation. In Section VI we show that with the simplified message importance the system can achieve the same performance as exploiting the original message importance in (5).

Note that the message importance in (10) depends on how we choose the alternative input $\boldsymbol{m}_i'$. For example, if we exploit zero vector $\boldsymbol{m}_i' = \mathbf{0}$, the message importance will be

$$I_j(\boldsymbol{m}_i) = \|\boldsymbol{q}_j^{\mathrm{T}} \boldsymbol{q}_i \boldsymbol{m}_i\|, \tag{11}$$

namely the influence-based message importance. Another choice is using history message $\boldsymbol{m}_i^{\mathrm{o}}$ (of the last time slot when agent i is scheduled) as model input $\boldsymbol{m}_i'$, which obtains the message importance

$$I_j(\boldsymbol{m}_i) = \|\boldsymbol{q}_j^{\mathrm{T}} \boldsymbol{q}_i (\boldsymbol{m}_i - \boldsymbol{m}_i^{\mathrm{o}})\|, \tag{12}$$

namely the difference-based message importance. In Sec. V we will use predicted message to replace $\boldsymbol{m}_i'$ to achieve better performance.

B. Decentralized Scheduling

Fig. 2 shows the workflow of agent scheduling for message exchange. After generating messages and queries, the agents broadcast their queries to other agents. If the agents can calculate message importance of all agents as explained in what follows, the system can perform decentralized scheduling. Eq. (10) shows that the calculation of message importance

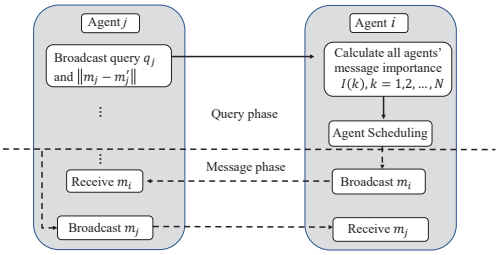

Fig. 2. The workflow of decentralized scheduling.

needs not only the queries, but also the message distance $\|\boldsymbol{m}_i - \boldsymbol{m}'_i\|, i = 1, 2, \cdots, N$. Therefore, in the query phase, the message distance, only a floating point number, is integrated into the packet of query for broadcasting. The message importance of an agent is the summation of its impact on all other agents. Taking agent i as example, its message importance is

$$I(i) = \frac{1}{N-1} \sum_{j \neq i} I_j(\boldsymbol{m}_i). \tag{13}$$

After calculating the message importance of all agents, N_{c} agents with the highest message importance take the resource blocks according to the order of message importance in the message phase.

V. Message Prediction

Due to the resource limitations, some agents can not be scheduled to share their messages. To perform MARL without these updated messages, which are the input of the learning model, one can exploit history messages or zero vectors as the model input, but this brings performance degradation. To reduce this performance degradation, we propose message prediction mechanism to provide better model input.

In MARL systems, the messages from agents encode the information of their states, observations and decisions. With this information, an agent can predict these agents' actions and following states, with which it can further predict these agents' new messages. Note that the prediction is performed for the messages with relatively small importance otherwise they should be scheduled. If an agent has small message importance, its state and decision will have high probability to be stable. This is the insight why the message prediction can perform well for these agents.

To predict messages, we introduce function $F_{\mathrm{p}}(\cdot)$, a neural network, into the MARL model. For the training of $F_{\mathrm{p}}(\cdot)$, we use the history messages as network input and the newest messages as the groundtruth of the network output. The loss function used for training is L_2-norm loss. An agent stores every agent's message $\boldsymbol{m}' = \{\boldsymbol{m}'_1, \boldsymbol{m}'_2, \cdots, \boldsymbol{m}'_N\}$, which are broadcasted or predicted, and updates them in every time slot as shown in Alg. 1

After message update (including message broadcast and message prediction) in Alg. 1, the updated $\boldsymbol{m}'$ can be used for message aggregation, reducing the performance degradation resulted by missing message sharing.

Algorithm 1 Message update with prediction

Require:
Newest messages $\boldsymbol{m} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \cdots, \boldsymbol{m}_N\}$;
History messages $\boldsymbol{m}' = \{\boldsymbol{m}'_1, \boldsymbol{m}'_2, \cdots, \boldsymbol{m}'_N\}$;
Message prediction function F_{p};
Ensure:
Updated history messages $\boldsymbol{m}' = \{\boldsymbol{m}'_1, \boldsymbol{m}'_2, \cdots, \boldsymbol{m}'_N\}$;
for $i = 1$ to N **do**
  if Agent i broadcast its newest message $\boldsymbol{m}_i$ **then**
    $\boldsymbol{m}'_i = \boldsymbol{m}_i$
  else
    $\boldsymbol{m}'_i = F_{\mathrm{p}}(\boldsymbol{m}'_i)$
  end if
end for

Note that the message prediction function does not need extra communication resources. Then why is it able to provide more information for the MARL model input? Firstly, $F_{\mathrm{p}}(\cdot)$ has the knowledge of state transition and the agents' policy of making decisions, which is learned from the environment and thus adds extra information into the predicted message. Secondly, the message prediction is performed in every time slot until a new message is received. Therefore, the prediction also utilizes the information of the number of time slots that have passed since the last message broadcast.

VI. Experiment

A. Experiment Setup

We use a traffic junction environment similar to that of [8]. There is a 4-way junction on a 14×14 grid shown in Fig. 3. In each time slot and from each of the four directions, the arrival rate of new vehicle is $p = 0.1$. Every vehicle is randomly assigned one of three possible routes to exit the junction. To focus on the performance of the proposed communication schemes, the vehicle can only observe the state of the cell of its location, i.e., the vehicle has limited sensing capability.

The actions of a vehicle in any time slot include: taking one cell forward on its route or staying at its current location. Every vehicle occupies one cell and a collision occurs when more than one vehicle exists in one cell. There are at most $N_{\max} = 10$ vehicles in the junction, which means no vehicle will enter the junction if the number of vehicles reaches $N_{\max}$.

In every time slot, vehicles broadcast their messages via a wireless channel. Due to the limited resources blocks of the wireless channel, only N_{c} vehicles can broadcast messages in one time slot.

The state used for model input of a vehicle is a one-hot binary vector set $\{n, l, w\}$, which are its ID, location and assigned route respectively. The reward of collision is $r_{\mathrm{coll}} = -10$ and the reward due to vehicle staying is $\tau r_{\mathrm{time}} = -0.01\tau$, where τ is the number of time slots passed since the vehicle enters the junction. The total reward of the system at time slot t is

$$r(t) = C(t) r_{\mathrm{coll}} + \sum_{i=1}^{N(t)} \tau_i r_{\mathrm{time}}. \tag{14}$$

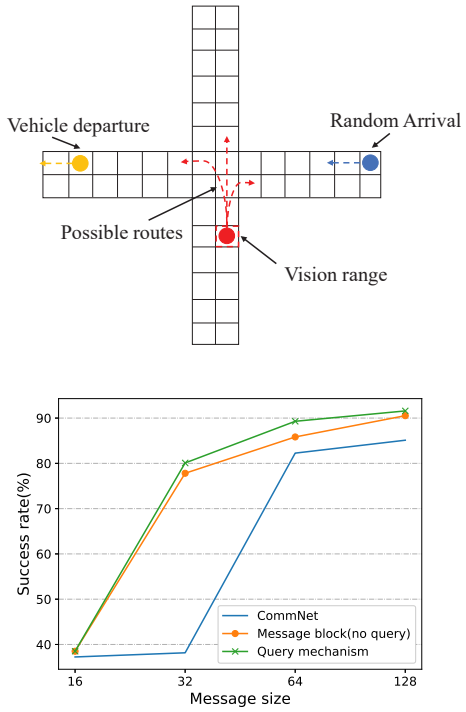

Fig. 4. Performance of CommNet, CommNet + message block, and CommNet + message block + query mechanism with different message sizes.

where $C(t)$ is the number of collisions at time t and $N(t)$ is the number of vehicles in the junction at time t.

In every simulation episode, the MARL system runs 40 time slots, where success means no collision occurs in the whole episode. The metric of system performance is success rate, i.e., the ratio of successful episodes to total episodes.

B. Simulation Results

Firstly, we evaluate the performance improvement brought by message block and query mechanism. We consider the case without constraint of communication resources, i.e., $N_c = N_{max} = 10$. The size of query is 16. Fig. 4 shows the performance of CommNet [8], CommNet + message block, and CommNet + message block + query mechanism with different message sizes. It is shown that the performance of CommNet degrades heavily when the message size is decreasing, especially when the message size is reduced to 32. However, with our proposed message block, the performance has been significantly improved. This is because, with the same message size, the sharing messages processed by message block can encode more information for cooperation compared with hidden states (which spend part of bits on encoding the information used for the decision making of themselves). The query mechanism also brings improvement because the weighted aggregation helps the agents filter more important information for decision making. When the message size reaches 64, the performance improvement due to increasing message size becomes marginal. Therefore, we choose 64 as the message size in the following experiments. The size of query is still 16.

Next, we evaluate the performance of the proposed query mechanism with our message importance measure. In Fig. 5 we compare the performance of random scheduling, IC3Net [15] and our proposed message-importance-based scheduling, including influence-based message importance (the alternative input is zero vector as shown in (11)) as well as difference-based message importance (the alternative input is history message as shown in (12)). Considering IC3Net does not need to send query, we set the message size for IC3Net to 128. To show how the query mechanism works under resource limitations, the constraints of resource blocks in Fig. 5 include $N_c = 3$, 5 and 8. For comparison, the performance of $N_c = 10$, i.e., without constraint of communication resources, is also shown. The success rate of random scheduling shows that the resource limitation results in significant performance degradation. Our proposed query mechanism brings significant improvement over random scheduling especially when the number of scheduled agents is less than 5, while having more than 5% improvement compared with IC3Net. In addition, we can find that the importance-aware message exchange can achieve the performance of random scheduling with only 60% of the total communication resources. Difference-based message importance performs better than influence-based message importance but the gain is marginal.

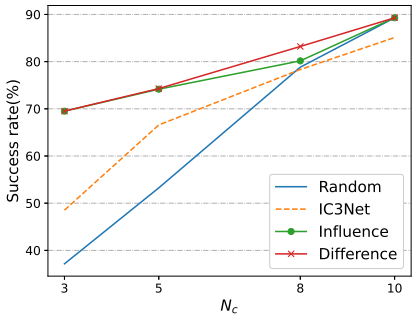

Fig. 5. Performance of query mechanism with different constraints of communication resources.

In the simplification of message importance in Sec. IV, we make some assumptions on the gradients, including uniform distribution of direction and independent distribution of magnitude, to get the approximate message importance. To show whether the assumptions are reasonable, we compare the performance of the derived approximate message importance and the original one using the actual gradients (as shown in (5)) in Fig. 6. The results show that the scheme with approximated message importance obtains almost the same performance as the one with no approximations.

Finally, we evaluate the performance of message prediction. For comparison, we choose a common method that exploits history message (the last shared message) as model input, i.e., the importance-aware message exchange with difference-based message importance. Fig. 7 shows the performance under different N_c. The system performance degrades much more slowly as the number of scheduled agents becomes smaller.

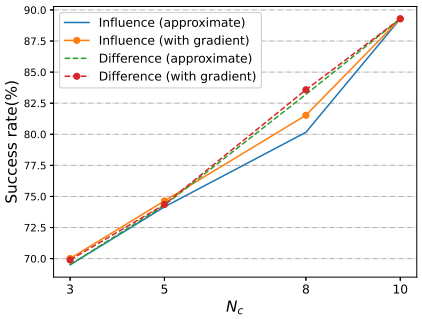

Fig. 6. Performance of approximate message importance and the importance using gradients.

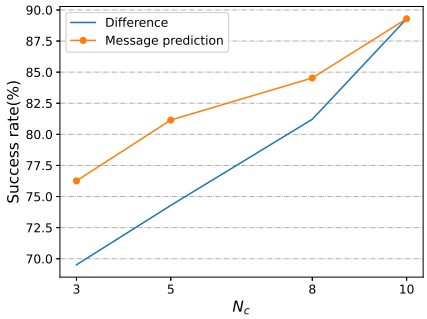

Fig. 7. Performance of message prediction with different constraints of communication resources.

The key of message prediction is providing more information for model input when the agents can not receive new messages. Therefore, with stringent resource limitations, the message prediction can still provide valuable information and obtain substantial performance gain. With message prediction, the system can obtain the same performance as the one without message prediction, while costing only about 60% of the total communication resources. In summary, our proposed schemes consisting of the importance-aware message exchange and prediction, can save about 60% communication resources compared with the one exploiting random scheduling and no message prediction.

VII. Conclusion

In this paper, we study the MARL system with limited communication resources by importance-aware message exchange and message prediction. The importance-aware message exchange filters important information needed for cooperation to save communication resources with decentralized scheduling policy. The message prediction extracts more information from the knowledge of the environment and history messages to help the reinforcement learning perform better.

We evaluate the performance of the proposed schemes under different constraints of communication resources. The results reveal the significant performance degradation as the communication resources become limited. Nevertheless, with the proposed communication schemes, the system can save about 60% of communication resources and achieve the same performance as compared with the one exploiting random scheduling and no message prediction, which means that the proposed scheme can slow down the performance degradation due to the resource limitations.

VIII. Acknowledgements

This work is sponsored in part by the National Key R&D Program of China No. 2020YFB1806605, by the Nature Science Foundation of China (No. 62022049, No. 61871254), and Hitachi Ltd.

References

[1] I. Althamary, C. Huang, and P. Lin, "A survey on multi-agent reinforcement learning methods for vehicular networks," in *15th International Wireless Communications & Mobile Computing Conference (IWCMC 2019)*, 2019, pp. 1154–1159.

[2] 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," *4th International Conference on Representation Learning (ICLR 2016)*, 2016.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*, 2017, pp. 6382–6393.

[5] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, 2018, pp. 6846–6859.

[6] M. Tan, "Multi-agent reinforcement learning: independent versus cooperative agents," in *Proceedings of the 10th International Conference on Machine Learning (ICML 1993)*, 1993, pp. 330–337.

[7] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS 2016)*, 2016, pp. 2145–2153.

[8] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," 2016, pp. 2252–2260.

[9] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, Dec 2016.

[10] G. Ding, T. Huang, and Z. Lu, "Learning individually inferred communication for multi-agent cooperation," in *Advances in Neural Information Processing Systems 33th (NeurIPS2020)*, vol. 33, 2020, pp. 22 069–22 079.

[11] S. Li, Y. Zhou, R. E. Allen, and M. J. Kochenderfer, "Learning emergent discrete message communication for cooperative reinforcement learning," in *37th Conference on Uncertainty in Artificial Intelligence (UAI 2021),early access*, 2021.

[12] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," *Advances in Neural Information Processing Systems 31th (NeurIPS2018)*, vol. 31, pp. 7254–7264, 2018.

[13] D. Kim, S. Moon, D. Hostallero, W. J. Kang, T. Lee, K. Son, and Y. Yi, "Learning to schedule communication in multi-agent reinforcement learning," in *7th International Conference on Representation Learning (ICLR 2019)*. International Conference on Representation Learning, 2019.

[14] V. Mnih, A. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 33th International Conference on Machine Learning (ICML 2016)*, 2016, pp. 1928–1937.

[15] A. Singh, T. Jain, and S. Sukhbaatar, "Learning when to communicate at scale in multiagent cooperative and competitive tasks," in *6th International Conference on Learning Representations (ICLR2018)*, 2018.