ASM-PPO: Asynchronous and Scalable Multi-Agent PPO for Cooperative Charging

Yongheng Liang
Sun Yat-sen University
Guangdong, China
liangyh38@mail2.sysu.edu.cn

Hejun Wu
Sun Yat-sen University
Guangdong, China
wuhejun@mail.sysu.edu.cn

Haitao Wang
Sun Yat-sen University
Guangdong, China
wanght39@mail2.sysu.edu.cn

ABSTRACT

Wireless Rechargeable Sensor Networks (WRSNs) are especially promising in large-area monitoring tasks that are previously impossible to complete by traditional Wireless Sensor Networks (WSNs). Mobile Chargers (MCs) in WRSNs are to cooperatively charge battery drained sensor nodes high efficiently and with a guarantee of sensors survival. Considering the unpredictability and high dynamics of WRSNs during the charging process, Multi-Agent Reinforcement Learning (MARL) is an attractive alternative to schedule the cooperation among MCs. However, most existing MARL methods are based on Decentralized Partially Observable Markov Decision Processes (Dec-POMDP), a general framework to describe decentralized agents making decisions at the same time step. Nevertheless, MCs in WRSNs perform charging asynchronously since the charging time of each sensor node varies. To address the problem of asynchronous behavior, we first formulate an Asynchronous Dec-POMDP (AD-POMDP). We then propose an algorithm called Asynchronous and Scalable Multi-agent Proximal Policy Optimization (ASM-PPO) that allows asynchronous learning and decision-making in AD-POMDP based on two popular multi-agent reinforcement learning methods in Dec-POMDP. Furthermore, ASM-PPO takes advantage of the translation invariance in WRSNs to avoid the huge input space dimensions caused by centralized training. The evaluation results not only indicate that our method achieves much charging efficiency and the longer lifetime of sensor nodes, but also demonstrate that ASM-PPO has advantages in terms of stability and scalability over existing methods.

KEYWORDS

wireless rechargeable sensor network; cooperative charging; multi-agent reinforcement learning

ACM Reference Format:
Yongheng Liang, Hejun Wu, and Haitao Wang. 2022. ASM-PPO: Asynchronous and Scalable Multi-Agent PPO for Cooperative Charging. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Wireless Sensor Networks (WSNs) are a special network paradigm that uses a large number of wireless sensor nodes for sensing, data collecting and processing [1, 23, 28]. A sensor node in WSNs can only work within a short period of time, due to the limited energy supply. Consequently, large-area monitoring tasks cannot be fulfilled by traditional WSNs because of the two reasons as follows: (1) The heavy network traffic of large-scale networks need dramatically more energy in wireless communication. (2) It is impossible to change batteries manually in a large area, in which there are hundreds of sensor nodes scattered within a range of several kilometers running out of energy at every second. In comparison, Wireless Rechargeable Sensor Networks (WRSNs) are more powerful, as they are capable of sensing continuously through in-network charging. With in-network charging of WRSNs, a number of Mobile Chargers (MCs) move within a WRSN to charge sensor nodes with low batteries.

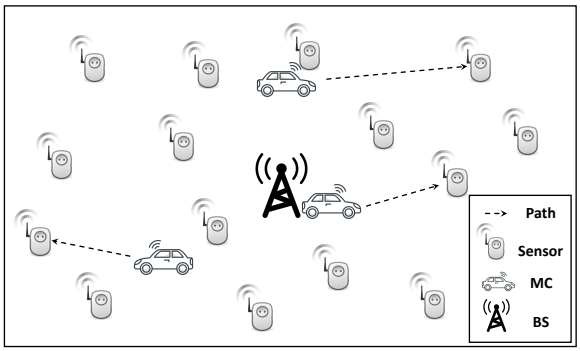

Figure 1: An example of a WRSN with three MCs

Due to the uneven workloads of different sensor nodes, MCs are desired to make runtime decisions cooperatively to maximize the efficiency of charging and to guarantee the survival of low-battery sensor nodes simultaneously. In large-scale WRSNs, multiple MCs equipped with resonant coils roam around the network and fully recharge the sensor nodes wirelessly, as shown in Figure 1. Therefore, how to schedule MCs to shorten the moving distance of MCs so that more energy can be transmitted to the working sensor nodes in WRSNs is the most important issue in cooperative charging.

In general, the scheduling methods of MC can be divided into two categories: off-line and online. (1) Off-line scheduling methods are also called static methods. They assume a constant network topology as well as a deterministic pattern of energy consumption rate so that MC can have a fixed charging sequence in each charging cycle [5, 19, 30]. (2) Online approaches require dying sensor nodes to submit charging requests to MC or the Base Station (BS), then the MC is able to rearrange the order of charging tasks according to the predefined charging strategy, and make charging decisions

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

dynamically [13, 17, 18, 27]. Unfortunately, such a manner of rule-based decision-making is either infeasible or inefficient for the long-term optimization, especially in large-scale unpredictable or highly dynamic WRSNs [6].

Multi-Agent Reinforcement Learning (MARL) has shown great potential in solving sequential decision-making problems, in which multiple autonomous agents aim to optimize the long-term return [9, 20, 24, 29]. Nevertheless, it is inappropriate to use the problem model of Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [22], a framework that can be applied to a wide range of MARL methods, for cooperative charging in WRSNs. The reason is rooted in the synchronous nature of the time step definition of the Dec-POMDP. A Dec-POMDP framework assumes synchronized action execution over agents, while MCs in WRSNs complete charging operations at different time points. Suppose we set the length of a time step to be short. Then one charging action of the MC will span multiple time steps. At these time steps, the action space of the MC is restricted to the uncompleted charging action only, and this causes training much inefficient. On the other hand, if we extend the time step to be long enough to guarantee that MCs make decisions synchronously, then those MCs with shorter action execution time will have a long idle time. Therefore, the traditional MARL algorithms based on Dec-POMDP, such as MAPDG [20], QMIX [24], and MAPPO [29], would work inefficiently in cooperative charging.

Recently, Chen *et al.* [6] proposed VarLenMARL to make MCs collaboratively learn a dynamic charging strategy. VarLenMARL is a MARL framework that supports the asynchronous decision-making process by using a Synchronized Delay-Tolerant Trajectory Collection (SDTTC) mechanism instead of the traditional trajectory data collection mechanism in MARL. The main idea of SDTTC is to use the current interactive information of the decision-making agent and the latest information of other agents as padding data to fill in the trajectory data for the training process. Unfortunately, for the decision-making agent, these padding data of other agents are not instant information at the exact same global time slot. Such padding data cause the following two bad results: (1) The decision-making agent has to take more time to find useful information from the trajectory. (2) The previous normal training process of the agent become unstable as it is interfered with by these padding data.

In this paper, we first propose an Asynchronous Dec-POMDP (AD-POMDP) as the problem model for asynchronous decision-making scenarios such as the cooperative charging problem in WRSNs. We then propose a new approach called Asynchronous and Scalable Multi-agent Proximal Policy Optimization (ASM-PPO), which adopts the mechanisms in two popular multi-agent reinforcement learning methods in Dec-POMDP, Independent Proximal Policy Optimization (IPPO) [9] and Multi-Agent Proximal Policy Optimization (MAPPO) [29], as a solution to the problem of AD-POMDP. In ASM-PPO, agents only use their own trajectory data, without padding data, to train independently and asynchronously, similar to IPPO. We adopt a Centralized Training and Decentralized Execution (CTDE) structure to process the global information to tackle the environment non-stationary caused by independent training and to learn the collaborative strategy, as that in MAPPO. Finally, since the MCs and the sensor nodes in a WRSN are both isomorphic, the information of different sensor nodes share the same form, i.e., the input state information of MC has the translation invariance. Considering this fact, we construct a special parameter sharing component with a heterogeneous convolution kernel and a GRU (Gate Recurrent Unit) unit [7] to process the isomorphic information of different MCs. This component is able to avoid the huge dimension of the input space of the centralized value function in ASM-PPO and thus solve the stability and the scalability problems, since an extremely wide or deep model is harmful to the training stability in reinforcement learning [3, 4, 11].

The contributions of this paper are summarized as follows:

1) We formulate a new framework, AD-POMDP, to model the problem for asynchronous operations of multiple agents, such as the cooperative charging problem in WRSNs.
2) We propose an asynchronous MARL algorithm, ASM-PPO, for AD-POMDP. ASM-PPO combines the trajectory collection mechanism in IPPO with the CTDE structure in MAPPO so that all agents can infer their collaborative policy using data collected from asynchronous decision-making scenarios while maintaining the stability of ASM-PPO.
3) We design a new network component consisting of a heterogeneous convolution kernel and a GRU cell by taking advantage of the translation invariance of MCs, which improves the scalability without increasing the parameters that need to be learned in ASM-PPO.
4) We perform evaluations of our algorithm and other methods in a variety of WRSNs scenarios. The results demonstrate that ASM-PPO improves the performance of a general MARL algorithm and learns a better cooperative charging schedule strategy in WRSNs than the state-of-the-arts.

The rest of this paper is organized as follows. Section 2 reviews the related schemes and methods. Section 3 presents the WRSN model and the problem formulation. Section 4 demonstrates our AD-POMDP and ASM-PPO in detail. Section 5 validates the performance of ASM-PPO against the state-of-the-arts. Section 6 gives a conclusion of the paper and points out the future work.

2 RELATED WORK

To guarantee the lifetime of sensor nodes while minimizing the charging cost of MCs, researchers have devoted efforts in a variety of directions in the past few years. From the aspect of the charging cycle of MCs, off-line charging schemes [5, 19, 30] assume that an MC predetermines the node charging order and the moving path before departure from the BS. Off-line charging schemes are only applicable to WSNs with a relatively stable rate of energy consumption and a rather static topology. Nevertheless, realistic WRSNs usually have burst data flow or dynamic topology changes, since events occur randomly. Online charging schemes [13, 17, 18, 27] enable an MC preferentially chooses the sensor node that should be charged urgently. The charging sequence of an MC is dynamically determined according to the status of sensor nodes. Unfortunately, it is still extremely difficult for online schemes to find the long-term optimal charging strategies for WRSNs in scenarios of complex topologies and dynamic events [6].

Multi-Agent Reinforcement Learning (MARL) has shown its great potential in many real-world sequential decision-making problems, especially in making the macro operation strategies, such as

those in gaming or in economics. Traditional MARL algorithms, such as MADDPG [20], COMA [10], QMIX [24] and MAPPO [29], use Dec-POMDP [22] as the problem model. In Dec-POMDP agents determine their actions according to their own partial observations. In these MARL methods, agents observe their own observations, freeze the time step to compute the optimal decisions, and then apply their actions at the same time step. Unsurprisingly, such MARL methods struggle to solve tasks with asynchronous working agents like WRSNs. In WRSNs, operations such as charging and moving take different intervals from time to time. Therefore, it is significantly inefficient in training in these traditional WRSNs, as they assume that all agents are performing one operation within one time step.

Researchers have proposed a number of solutions to the problems in which one operation takes variable periods of time. For instance, some methods redefine a time step to be long enough to guarantee that all tasks are completed [21], or extend a Dec-POMDP to be a hierarchical framework with macro-actions [2]. Such methods are still not well-suited for cooperative charging problems. Lengthen each time step makes an MC idling for a long period of time, if this MC only needs a rather short time period in task execution. Moreover, using the hierarchical framework for Dec-POMDP makes the problem much more complicated, because different actions such as charging and waiting do not have hierarchical characteristics.

So far, there have been two major structures proposed for MARL to Dec-POMDP. The first is Decentralizing Training and Decentralizing Execution (DTDE). DTDE decomposes a MARL problem into multiple decentralized and single-agent problems. DTDE allows an agent to regard other agents as a part of the environment and thus the agent can learn its policy only through its own observation independently [9, 12, 25]. While easy to handle the scalability problem caused by the increase of the number of agents, DTDE suffers from theoretical limitations including a non-stationary environment and sensitivity to partial observability [26]. The other structure is CTDE. CTDE improves upon DTDE by using a centralized critic to process the global state during centralized training. In addition, CTDE uses several decentralized actors to process local observations[10, 20, 29].

Although CTDE can solve the problem of partial observability of MARL, there is still a huge input space dimension for the centralized critic, especially when there are a large number of agents. For MARL, an extremely wide or deep network is harmful to training stability [3, 4, 11]. Thus, how to make the model small enough to improve the stability and training speed, while ensuring that the model capacity is sufficient to learn the complex functions is a critical issue in MARL.

3 PROBLEM FORMULATION

In this section, we present the WRSN network model in detail and give the statement of the objective problem for cooperative charging.

3.1 Network Model

In this paper, we study the problem of cooperative charging in a two-dimensional L-meter-square WRSN. The WRSN is configured with M homogeneous MCs in addition to N homogeneous stationary sensor nodes and a base station (BS) located in the center of the area. The major notations for WRSN are listed in Table 1.

Table 1: Major Notations for WRSN

Notation	*Description*
L	Side length of the area
N	Number of sensor nodes
M	Number of MCs
s_k	Sensor node with ID k
m_j	MC with ID j
E_s	Maximum battery capacity of sensor nodes
ξE_s	Minimum battery capacity of sensor nodes
E_c	Battery capacity of MCs
$E_{res}^{[t_i^k]}$	Residual energy of s_k at t_i
$E_{res}^{[t_i^j]}$	Residual energy of m_j at t_i
h_s	Energy threshold for sensor nodes
h_c	Energy threshold for MCs
Loc_i	Location of sensor node or MC with ID i
ρ	Energy transfer efficiency
t_i	i-th time slot of global system clock
T_i^j	i-th time step of m_j's local clock
T_{-i}^j	Number of time slots to finish m_j's action
η	Average charging utility
$E_{cha}^{[T_i^j]}$	Charging cost of m_j in time step T_i^j
$E_{mov}^{[T_i^j]}$	Moving cost of m_j in time step T_i^j
θ	Request miss rate
$\mathcal{N}_{miss}$	Number of missed requests
$\mathcal{N}_{req}$	Number of total requests

In a WRSN, all sensor nodes are randomly placed in a field to be monitored. Each sensor node is equipped with a rechargeable Lithium battery with different limited capacities. The maximum battery capacity of these sensor nodes is E_s, and the minimum ξE_s. As events may occur unpredictably around any sensor node and cause the subsequent burst data flow from the sensor node, the energy consumption rate of sensor nodes is dynamically changing. To model random events, we assume that the occurrence of an event follows the Poisson Distribution with parameter λ. The duration of an event follows the Exponential Distribution with parameter μ. When the remaining energy of a sensor node falls below a pre-defined threshold h_s, the sensor node initiates a charging request that carries its ID, location, residual energy, current energy consumption rate, and a timestamp. This charging request finally reaches the global request pool in the BS, through multi-hop forwarding. The sensor node can only be recharged by one MC at a time. If the sensor node cannot be recharged in time, it will go to sleep and not be able to provide any services in WRSN.

MCs in a WRSN are deployed to the location point same as that of the BS initially. Once a charging request arrives at the BS, the BS broadcasts it to all of the MCs. Then an MC that has a battery capacity E_c chooses to serve the charging request according to its own strategy. The MC moves to the location close enough to the selected sensor node in a straight line, and fully recharges the node

wirelessly. We assume that all MCs have the same energy transfer efficiency ρ. ρ is the ratio of the energy received by the sensor node to the energy transmitted by the MC. When MC completes the charging task, it continues to choose to serve the next request, or stay where it is now to wait for serving another charging request. When the residual energy of an MC falls below the threshold h_c, it will move back to the BS for energy provisioning to be prepared for the next charging cycle.

3.2 Problem Formulation

We define a global system clock and the MC's local clock in a WRSN. The global system clock represents the notion of the world time in a WRSN. The global system clock is used for global time and the duration of the i-th time slot t_i in the global system clock is short enough. Note that, the unit for this global clock is a time slot, and the time unit for an agent is a time step. The length of a time step is variable and the length of a time slot is constant, as the time slot is a global concept. In an MC, T_i^j is the i-th time step in terms of the local clock of MC m_j. All of the MCs share the same global system clock and each MC has its own private local clock. The local clock of two MCs can be different because MCs are performing charging operations asynchronously. Within a local time step, m_j can finish an action, e.g., a charging task that m_j moves to the node location and fully recharges the node, or a waiting task in which m_j keeps idling. Consequently, as the length of executing each operation is different, the duration of each local time step T_i^j is variable.

Given the network model, global and local time clock as described above, the cooperative charging problem is to schedule multiple MCs to serve different charging tasks simultaneously. The objective of this problem is jointly maximizing the average charging utility of MCs and minimizing the number of dead sensor nodes.

Firstly, we define the first optimization objective, namely the average charging utility η of these M MCs, as Equation (1):

$$\eta = \frac{1}{M}\sum_{j=1}^{M}\frac{\rho\sum_{T_i^j=1}^{T^j}E_{cha}^{[T_i^j]}}{\sum_{T_i^j=1}^{T^j}E_{cha}^{[T_i^j]}+E_{mov}^{[T_i^j]}} \tag{1}$$

subject to:

$$0 \le \rho \le 1,$$

$$0 \le \sum_{T_i^j=1}^{T^j} E_{cha}^{[T_i^j]} \le E_c,$$

$$0 \le \sum_{T_i^j=1}^{T^j} E_{cha}^{[T_i^j]} + E_{mov}^{[T_i^j]} \le E_c.$$

where T^j is the total number of time steps for MC m_j using its own local clock, $E_{cha}^{[T_i^j]}$ the total amount of energy provided by m_j to a sensor node in the time step T_i^j, while $E_{mov}^{[T_i^j]}$ is the energy spent in moving of m_j in the time step T_i^j. Apparently, a lower $E_{mov}^{[T_i^j]}$ is better.

To minimize the number of dead sensor nodes, we turn to minimize the missing rate of charging requests θ. As shown in Equation (2), θ is the ratio between the number of unserved charging requests, $\mathcal{N}_{miss}$, and the total number of charging requests $\mathcal{N}_{req}$, which is the number of sensor nodes deaths caused by not being recharged in time, to the number of the total charging requests $\mathcal{N}_{req}$.

$$\theta = \frac{\mathcal{N}_{miss}}{\mathcal{N}_{req}} \tag{2}$$

subject to:

$$0 \le \mathcal{N}_{miss} \le \mathcal{N}_{req}$$

4 ASM-PPO

4.1 AD-POMDP

To solve the problem of cooperative charging, MCs in a WRSN should collectively accomplish charging tasks asynchronously. However, since the current Dec-POMDP framework requires synchronous decision-making, it is inappropriate to model such a cooperative charging problem as a Dec-POMDP. This section presents AD-POMDP as an improved problem model.

AD-POMDP is defined as a tuple $\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \mathcal{O}, \mathcal{A}, \Omega, U, P, R, \gamma \rangle$, where $\mathcal{N} \equiv \{1, 2, ..., M\}$ is a finite set of M agents, $\mathcal{S}$ the global state space of all agents. $\mathcal{O} \equiv \{\mathcal{O}^1, ..., \mathcal{O}^M\}$ and $\mathcal{A} \equiv \{\mathcal{A}^1, ..., \mathcal{A}^M\}$ are the observation space and the action space, respectively. Among them, $\mathcal{O}^j$ is the observation space of agent j and $\mathcal{A}^j$ the action space of agent j.

As agents in AD-POMDP make decisions asynchronously. In AD-PODMP, at time slot t_i, there is a set of available agents $\mathcal{N}'$, $\mathcal{N}' \subseteq \mathcal{N}$. Only the agents in $\mathcal{N}'$ need to make decisions at time slot t_i. $\mathcal{N}'$ is obtained from function $U : \mathcal{S} \to N'$, given the global state $\mathcal{S}$. Corredpondingly, the observation vector $\boldsymbol{o}$ of agents $\mathcal{N}'$ can be obtained from function $\Omega : \mathcal{S} \times \mathcal{N}' \to \mathcal{O}^{\mathcal{N}'}$. Agent j in $\mathcal{N}'$ selects an action $a_j \in \mathcal{A}^j$ according to its own observation, form a joint action vector $\boldsymbol{u} \in \mathcal{A}^{\mathcal{N}'}$. The joint action execution of these agents results in a transition to the next global state $s' \in \mathcal{S}$ through transition function $P : \mathcal{S} \times \mathcal{A}^{\mathcal{N}'} \to \mathcal{S}$. $\boldsymbol{r}$ is a reward vector from the reward function $R : \mathcal{S} \times \mathcal{A}^{\mathcal{N}'} \times \mathcal{S} \to \mathbb{R}^{|\mathcal{N}'|}$. Finally, $\gamma \in (0, 1]$ is the discount factor.

Now we present the key elements of the AD-POMDP for the cooperative charging problem in WRSNs. Note that our proposed algorithm uses the CTDE framework. When conducting centralized training in a simulation environment, we can easily obtain the information of all MCs at the same time slot. Meanwhile, in decentralized execution, each MC obtains its local observation at different time slots according to its own local clock and performs its action asynchronously.

Observation Space: We denote the local observation of MC m_j at global system clock time slot t_i as $o_j^{[t_i]}$. $o_j^{[t_i]}$ includes the information of the MC itself and the information of the charging request from sensor nodes. The information of an MC is defined as a tuple $\left\langle T_{-i}^j, Loc_j, E_{res}^{[t_i^j]} \right\rangle$, where T_{-i}^j denotes the number of global system clock slots to finish the current charging task of MC m_j, Loc_j is the location of m_j, and $E_{res}^{[t_i^j]}$ is the residual energy of m_j at time slot t_i. For the decision-making agent, T_{-i}^j will be zero. In order to make the information form of different MC consistent, and to make better use of MCs' translation invariance, we still

keep these variable in the local observation. The information of a sensor node obtained from its charging request is defined as a tuple $\left\langle Loc_k, E_{res}^{[t_i^k]}, p_s^k \right\rangle$. In this tuple, Loc_k is the location of sensor node s_k, $E_{res}^{[t_i^k]}$ the residual energy of s_k at time slot t_i, and p_s^k the energy consumption rate when the sensor node sends this request. Therefore, for the decision-making agent m_j, its local observation includes its own information $\left\langle T_{-i}^j, Loc_j, E_{res}^{[t_i^j]} \right\rangle$, and N_{near} tuples $\left\langle Loc_k, E_{res}^{[t_i^k]}, p_s^k \right\rangle$ obtained from the charging requests sent by the N_{near} nearest nodes from m_j. In addition, if there are not so many charging requests in the current request pool, the remaining node information will be filled with zeros.

State Space: We denote the global state of m_j at time slot t_i as $s_j^{[t_i]}$. $s_j^{[t_i]}$ is the concatenation of the local observations from all agents at t_i. Suppose at time slot t_i, m_j is to make a decision and another agent $m_{j'}$ is on its way to sensor node s_k. For global state $s_j^{[t_i]}$, T_{-i}^j in $o_j^{[t_i]}$ is zero and $T_{-i}^{j'}$ in $o_{j'}^{[t_i]}$ is the number of time steps $m_{j'}$ needs to complete the current charging task. When $m_{j'}$ becomes inactive due to low remaining energy before m_j gose back to the BS, the observation of $m_{j'}$ is replaced by a specific zero vector. Such a specific zero vector is also called *Death Masking* [29]. Note that the form of the local observation $o_j^{[t_i]}$ of each agent in the global state $s_j^{[t_i]}$ is consistent. Therefore, it is convenient to use the heterogeneous convolution kernel to avoid the huge input space dimension of a centralized critic and to improve the scalability of our algorithm.

Action Space: We denote the action of m_j at its own local time step T_i^j as $a^{[T_i^j]}$. Since each agent chooses N_{near} charging requests sent by the nearest nodes from the agent itself, the agent should select one of these requests and complete it, or wait for the next better charging request. Therefore, the dimension of the action space of agent m_j is $N_{near}+1$. Suppose there are only $n < N_{near}$ charging requests in the request pool, the remaining $N_{near} - n$ actions will all be invalid actions.

Reward Function: The local reward of m_j at its own local time step T_{i+1}^j is $r^{[T_{i+1}^j]}$. According to the two optimization goals for cooperative charging mentioned above, we need to increase the amount of energy transmitted into the network while reducing the moving distance of MCs and the number of dead sensor nodes in the network. To include these objectives, we define $r^{[T_{i+1}^j]}$ as follows:

$$r^{[T_{i+1}^j]} = \frac{E_{cha}^{[T_i^j]}}{E_{cha}^{[T_i^j]} + E_{mov}^{[T_i^j]}} - \alpha \frac{\mathcal{N}_{miss}^{[T_i^j]}}{\mathcal{N}_{miss}^{[T_i^j]} + \mathcal{N}_{req}^{[T_i^j]}} \tag{3}$$

where α is used to adjust the two ratios in the reward function. As shown in Equation (3), the two ratios in $r^{[T_{i+1}^j]}$ reflect: (1) the relation between the charging energy of m_j and the moving distance, and (2) the relation between the total number of dead nodes and the total number of requests received. This reward function also reflects the collisions in which multiple chargers move towards the same sensor node. As such collisions significantly reduce the charging utility, MCs will get a lower reward if collision arises.

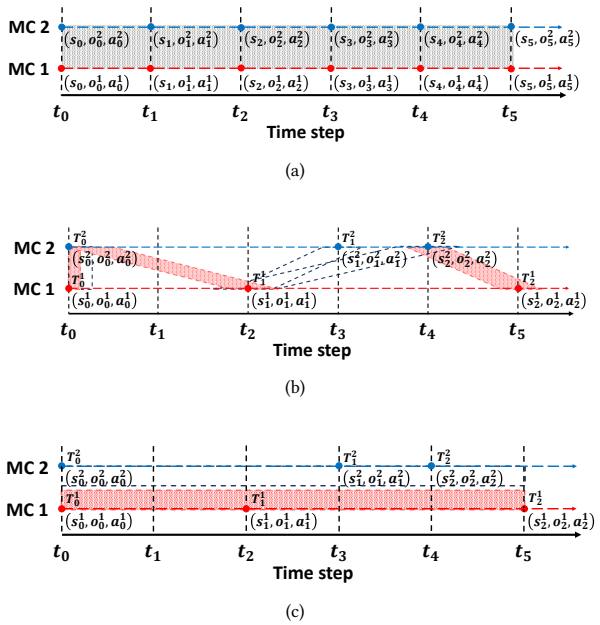

Figure 2: Comparison of the trajectory collection mechanisms under Dec-POMDP, SDTTC, and our proposed AD-POMDP. (a) Dec-POMDP collects all the joint information at the same time slot as a trajectory. (b) In SDTTC MC 1 collects the data covered by shadow and puts them together into one trajectory. (c) In AD-POMDP MC 1 only collect its own trajectory, namely the part covered by shadow.

4.2 Trajectory Collection Mechanism

To train the MARL algorithm, we first need to collect the trajectory data composed of the interaction information of all agents, such as global states, local observations, actions, rewards, etc. However, the original trajectory collection mechanism of Dec-POMDP, which requires collecting the joint actions and joint observations of all agents at the same time step, is no longer applicable to AD-POMDP. Therefore, for AD-POMDP, we only collect the trajectory of agent m_j based on its own local clock T_i^j asynchronously. For M agents in each episode, we collect M trajectories which will be used for ASM-PPO training together.

To illustrate the difference in the trajectory collection mechanism of Dec-POMDP, Synchronized Delay-Tolerant Trajectory Collection (SDTTC) mechanism [6], and AD-POMDP, we use the following example setup. Suppose s_i^j is the global state of agent j at local time step T_i^j, o_i^j the observation of agent j at T_i^j, and a_i^j the action of agent j at T_i^j. For brevity, we do not consider other data collected, such as rewards.

Dec-POMDP requires the collection of interactive information of all agents at the same time step, as shown in Figure 2(a), the

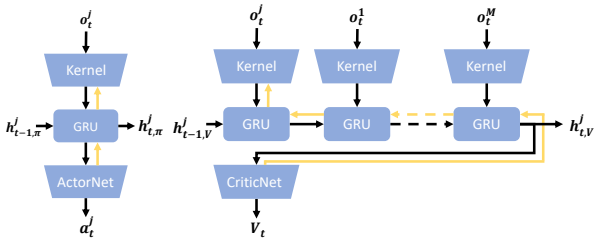

Figure 3: ASM-PPO architecture. The dark arrow indicates the data flow of forwarding propagation, and the light arrow indicates the data flow of gradient backpropagation.

trajectory collected is:

$$[(s_0, o_0^1, o_0^2, a_0^1, a_0^2), (s_1, o_1^1, o_1^2, a_1^1, a_1^2), (s_2, o_2^1, o_2^2, a_2^1, a_2^2), ...]$$

Since the joint interaction information of all agents cannot be collected in asynchronous decision-making scenarios, SDTTC requires each agent to collect a trajectory, and to use the latest data of the other agents as padding data, as shown in Figure 2(b). The trajectories of MC 1 and MC 2 are:

$$[(s_0^1, o_0^1, a_0^1, s_0^2, o_0^2, a_0^2), (s_1^1, o_1^1, a_1^1, s_0^2, o_0^2, a_0^2), (s_2^1, o_2^1, a_2^1, s_2^2, o_2^2, a_2^2), ...]$$
$$[(s_0^2, o_0^2, a_0^2, s_0^1, o_0^1, a_0^1), (s_1^2, o_1^2, a_1^2, s_1^1, o_1^1, a_1^1), (s_2^2, o_2^2, a_2^2, s_1^1, o_1^1, a_1^1), ...]$$

We can see that in the trajectory of MC 1 collected by SDTTC, the padding data of MC 2 is incomplete due to the missing data of MC 2 in T_1^2, which will interfere with the training process. AD-POMDP requires each agent to collect only its own interaction information without padding data, as shown in Figure 2(c). Correspondingly, the trajectories of MC 1 and MC 2 are as follows:

$$[(s_0^1, o_0^1, a_0^1), (s_1^1, o_1^1, a_1^1), (s_2^1, o_2^1, a_2^1), ...]$$
$$[(s_0^2, o_0^2, a_0^2), (s_1^2, o_1^2, a_1^2), (s_2^2, o_2^2, a_2^2), ...]$$

Therefore, the trajectory collection mechanism in AD-POMDP enables agents to collect trajectories asynchronously. Furthermore, this mechanism is relatively more flexible as well as stable, and reduces the storage of redundant padding data.

4.3 Architecture

Our ASM-PPO is designed on the basis of AD-POMDP. This algorithm follows the CTDE framework with parameter sharing, where all agents share the same policy $\pi_\phi(a|o)$ and centralized value function $V_\varphi(s)$. In the actor network and the critic network of our algorithm, there is a heterogeneous convolution kernel and a GRU unit, as illustrated in Figure 3. We adopt the learning mechanism of MAPPO in designing ASM-PPO because MAPPO uses stochastic policy and is more suitable for task scenarios with large randomness like cooperative charging in WRSNs.

In a WRSN, the forms of the local observations of each agent are consistent. Therefore, for the centralized critic network in ASM-PPO, we can easily use a parameter-sharing heterogeneous convolution kernel to process the local observations of different agents in the global state. In ASM-PPO, the heterogeneous convolution kernel is instantiated by MLP (Multi-Layer Perceptron), and is used to extract the current features of different agents, or to check the

Algorithm 1 ASM-PPO

Input: A set of M agents m_j with their shared policy $\pi_\phi(a|o)$
Output: A trained collaborative policy $\pi_\phi(a|o)$

Initialize parameters ϕ for policy $\pi_\phi(a|o)$ and parameters φ for critic $V_\varphi(s)$
for $episode = 1 \rightarrow episode_{max}$ **do**
  Set data buffer $D = \{\}$
  Initialize $\tau^1, ..., \tau^M$ empty lists
  Initialize $h_{0,\pi}^1, ..., h_{0,\pi}^M$ actor RNN states
  Initialize $h_{0,V}^1, ..., h_{0,V}^M$ critic RNN states
  for $t = 1 \rightarrow T$ **do**
    for all available agents a **do**
      Obtain $\tau_2^a = (s_t^a, o_t^a, r_t^a)$
      Complete a step $\tau_{t-1}^a \leftarrow \tau_1^a \cup \tau_2^a$
      $\tau^a \leftarrow \tau^a \cup \tau_{t-1}^a$
      $p_t^a, h_{t,\pi}^a = \pi(o_t^a, h_{t-1}^a; \phi)$
      $u_t^a \sim p_t^a$
      $v_t^a, h_{t,V}^a = V(s_t^a, h_{t-1,V}^a; \varphi)$
      Obtain $\tau_1^a = (s_t^a, o_t^a, h_{t,\pi}^a, h_{t,V}^a, u_t^a)$
    end for
    Execute actions $\boldsymbol{u_t}$
  end for
  for all agents a **do**
    $D \leftarrow D \cup \tau^a$
  end for
  Compute rewards-to-go $\hat{R}_t$ in D
  Compute advantage estimates $\hat{A}_t$ in D
  Adam update ϕ on π_ϕ like PPO
  Adam update φ on V_φ like PPO
end for

"busy" degree of these MCs. We use a GRU cell to process the output of different agents from the heterogeneous convolution kernel, which can improve the scalability without increasing the parameters that need to be learned in ASM-PPO when the number of agents increases. Additionally, to overcome the partial observability in AD-POMDP and to fully utilize the GRU, the hidden state in the GRU of the critic and the actor networks are also forwarded throughout an episode. Compared with LSTM (Long Short-Term Memory) [14], GRU has fewer parameters, faster training speed, and requires fewer data in training. Hence GRU is more suitable for reinforcement learning. Eventually, the output of a GRU cell will be fed into the rest of the critic network.

In the actor network, the local observations of m_j transport directly through the heterogeneous convolution kernel, the GRU cell, and the rest of the actor network, since there is no need to process other agents' information. Moreover, there are only a few charging requests in the request pool at the beginning of a charging cycle or when WRSN is not so busy. This leads to a large number of unavailable actions in the action space at these time steps. These illegal actions slow down the training speed of the model, if we treat all these actions as waiting actions. To avoid this problem, ASM-PPO uses action mask [16] to mask out the invalid actions in both the forward and backward pass, by replacing the weight of the unavailable action with a large enough negative number. In

Table 2: Network Parameters

Parameter	*Value*
Side length of the area L	1000m
Number of sensor nodes N	100 to 500
Number of MCs M	3 to 7
Maximum battery capacity of sensor nodes E_s	100J
Minimum battery capacity of sensor nodes ξE_s	0.6×100J
Battery capacity of MCs E_c	10kJ
Energy threshold for sensor nodes h_s	0.3
Energy threshold for MCs h_c	0.3
Energy transfer efficiency ρ	0.9
Charging power of MC p_c	2W
Moving power of MC p_c	5W
Moving speed of MC p_c	5m/s
Poisson Distribution λ	1
Exponential Distribution μ	1
Action window sizse N_{near}	5
Reward coefficient α	1

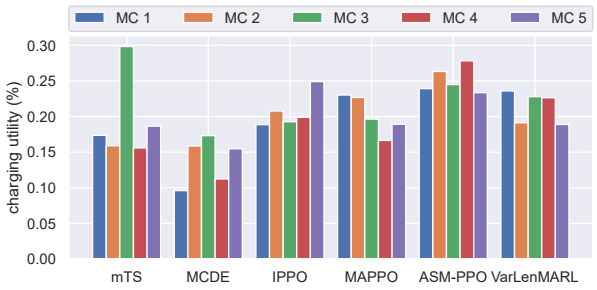

Figure 4: Comparison of the performance of MCs in different methods in terms of charging utility.

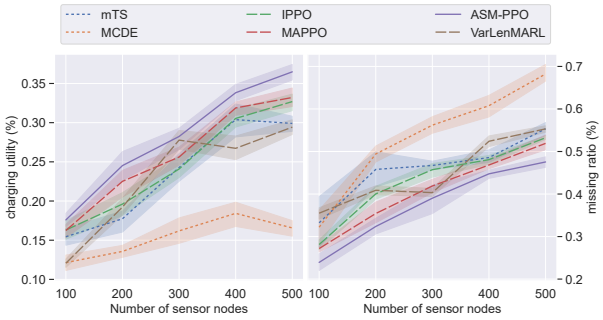

Figure 5: Comparison of all algorithms in terms of the average charging utility and the request miss rate under different sensor nodes.

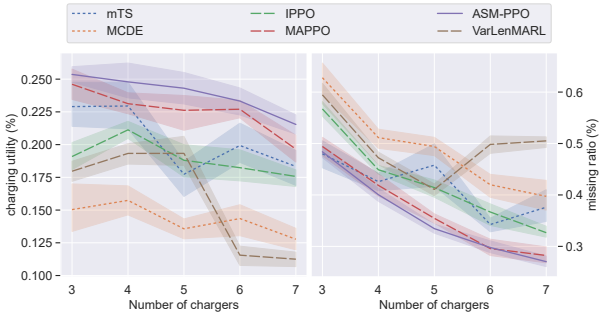

Figure 6: Comparison of all algorithms in terms of the average charging utility and the request miss rate under different MCs

this way, when we use the Softmax function to calculate the logits of each action, the logits of these unavailable actions will be close enough to zero.

In the training process, we update the actor network $\pi_\phi(a|o)$ and the critic network $V_\varphi(s)$ just like the original MAPPO. For $V_\varphi(s)$ in ASM-PPO, the gradient will only flow through the kernel which processes the observation of the decision-making agent, as shown by the light arrow in Figure 3. The pseudocode for ASM-PPO is shown in Algorithm 1.

5 EVALUATION

5.1 Configurations

In the evaluation of algorithms, we implemented a WRSN using existing source codes [8]. The configurations of the WRSN have listed in Table 2. On such a WRSN, we compare ASM-PPO with mTS [17], MCDE [27], IPPO [9], MAPPO with VarlenMARL framework [6], and MAPPO with AD-POMDP. MAPPO with AD-POMDP is an ablation experiment to test the structure consisting of a heterogeneous convolution kernel and GRU cell in ASM-PPO.

For the network structure, we take the WRSN with 5 MCs as an example. The actor and critic network of IPPO both consist of three hidden layers of 32, 64, 64 units. In MAPPO with the VarlenMARL framework and with AD-POMDP, the actor network consists of one hidden layer of 32 units and five hidden layers of 64 units. And critic network consists of one hidden layer of 120 units and four hidden layers of 64 units. For ASM-PPO, the actor and critic network both consist of two hidden layers of 32 and 64 units, one GRU layer whose hidden state dim is 64, and two hidden layers of 64 units. All of these networks utilize ReLU (Rectified Linear Units) non-linearity and Hardswish non-linearity [15] for the last layer.

The performance of each method in WRSN was measured with average charging utility η, as defined in Equation (1), and request missing rate θ, as defined in Equation (2), in one episode. In this paper, one episode refers to a single charging cycle that MCs start from the BS to perform charging tasks until all MCs have returned to the BS due to low remainder energy. For a fair comparison, the evaluation results are averaged over five different random seeds. Furthermore, we make all MARL methods use the same hyperparameters as those of ASM-PPO. To improve the generalization ability of the scheduling algorithm, the positions of all sensor nodes are randomly determined before each episode in the training process. We also use the relative distance and direction between the decision-making MC and another MC or sensor node, namely the

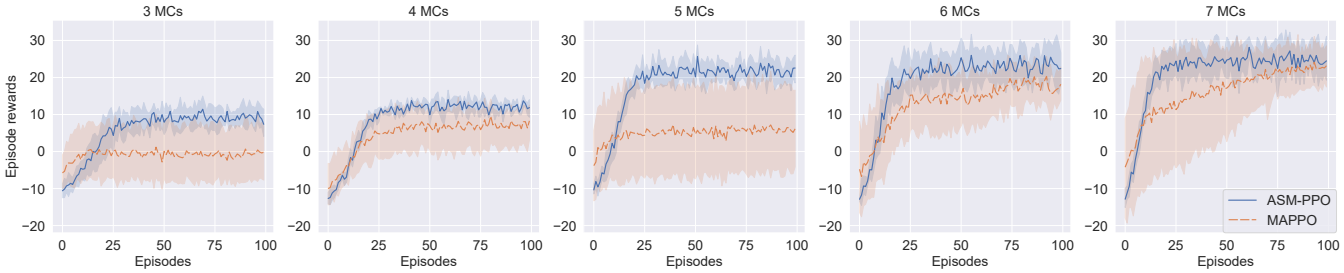

Figure 7: Ablation experiments demonstrating the effect of the heterogeneous convolution kernel and GRU unit in ASM-PPO.

relative position, to replace the absolute location coordinate of that MC or sensor node.

5.2 Evaluation Results

We first study the performance of each MC in WRSN with 200 sensor nodes and 5 MCs. The results are depicted in Figure 4. Obviously, in ASM-PPO, the MCs can transmit more energy into the network, while keeping a relatively shorter moving distance. Meanwhile, MCs in ASM-PPO perform much better in terms of load balancing than the traditional online methods such as mTS and MCDE. Especially in mTS, MC 3 achieves a much higher charging utility than all other MCs. This is because the MARL method can learn the long-term optimal cooperative charging strategy and enable all MCs to get a higher reward, while the MCs in traditional rule-based methods are difficult to handle the large-scale and highly dynamic WRSNs.

We then carry out the evaluations in WRSNs with various settings to test the generation ability of these methods. Figure 5 show the experimental results in WRSNs with 5 MCs and 100 to 500 sensor nodes. ASM-PPO can stably achieve 6% higher charging utility and 8% lower request miss rate in comparison with existing frameworks. In comparison, VarLenMARL has poor stability, and the performance of VarLenMARL changes significantly. It can also be seen from the figure that as the number of sensor nodes increases, the decrease in the distance between sensor nodes improves the average charging utility. However, the large set of sensor nodes also increases the burden on the MCs, making the request miss rate increase at the same time.

In Figure 6, we fix 200 sensor nodes and change the number of MCs from 3 to 7. It can be seen from the figure that ASM-PPO is still superior to other methods, and there is an improvement of 5% in terms of the charging utility and 9% in terms of the request missing rate. Compared with the high scalability of ASM-PPO, when the number of MCs is greater than 5, the performance of VarLenMARL drops sharply. The main reason is that the increase in the number of MCs increases the size of the padding data in the collected trajectories and this interferes with the stability of the algorithm. Interestingly, the increase in the number of MCs enables them to transmit more energy into the network, but the charging utility decreases, as well. This is because the average number of sensor nodes that each MC needs to serve is reduced. This is equivalent to the scenario that the sensor nodes become sparse for MCs. Consequently, the charging utility becomes lower.

To further evaluate ASM-PPO, we compare the learning curve of MAPPO and ASM-PPO in a WRSN with 200 sensor nodes and 3 to 7 MCs in Figure 7. It can be found that in scenarios of different numbers of agents, ASM-PPO can all converge quickly. The convergence process of ASM-PPO is also quite stable, since the learning curve of our ASM-PPO does not change much under different random seeds. This demonstrates the importance of the heterogeneous convolution kernel and GRU unit for algorithm training. In comparison, when the number of agents increases, MAPPO may converge to a sub-optimal strategy prematurely, or the convergence of MAPPO slows down. These results show that a too large critic network will indeed affect the stability of training. In conclusion, the structure consisting of a heterogeneous convolution kernel and GRU cell is beneficial to the training stability and scalability of the algorithm.

6 CONCLUSION

This paper studies the cooperative charging problem in large-scale highly dynamic WRSNs. We first propose AD-POMDP as the problem formulation for asynchronous decision-making scenarios. With AD-POMDP, we propose ASM-PPO to learn the dynamic and adaptive charging strategy for MCs. ASM-PPO allows agents to behave asynchronously. ASM-PPO is equipped with a special component consisting of a heterogeneous convolution kernel and GRU cell to address the stability and scalability issues. The evaluation results demonstrate that ASM-PPO is much competitive to traditional as well as MARL scheduling methods in terms of average charging utility and request miss rate. Furthermore, the ablation experiments between MAPPO with AD-POMDP and ASM-PPO show that the special component we proposed can well improve the stability and scalability of MARL. Additionally, although we test these algorithms on a simplified WRSN model, MCs can still learn to cooperate in the complex environment with obstacles, water, etc., in the training process of MARL. We aim to further improve the scalability of ASM-PPO with GNN, and apply this algorithm to more generic asynchronous decision-making scenarios.

7 ACKNOWLEDGMENT

This paper was mainly supported by the Science and Technology Program of Guangzhou, China (No. 202002020045). This work was also partially supported by the Professorial and Doctoral Initiating Project of Huizhou University (No.2017JB005). We thank Mr. Yuxin Chen and Dr. Guoming Lai for their contributions to this work.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. 2002. Wireless sensor networks: a survey. *Computer Networks* 38, 4 (2002), 393–422. https://doi.org/10.1016/S1389-1286(01)00302-4

[2] Christopher Amato, George Dimitri Konidaris, and Leslie Pack Kaelbling. 2014. Planning with macro-actions in decentralized POMDPs. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, Ana L. C. Bazzan, Michael N. Huhns, Alessio Lomuscio, and Paul Scerri (Eds.). IFAAMAS/ACM, 1273–1280. http://dl.acm.org/citation.cfm?id=2617451

[3] Marcin Andrychowicz, Anton Raichuk, Piotr Stanczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. 2020. What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study. *CoRR* abs/2006.05990 (2020). arXiv:2006.05990 https://arxiv.org/abs/2006.05990

[4] Johan Bjorck, Carla P. Gomes, and Kilian Q. Weinberger. 2021. Towards Deeper Deep Reinforcement Learning. *CoRR* abs/2106.01151 (2021). arXiv:2106.01151 https://arxiv.org/abs/2106.01151

[5] Vrajesh Kumar Chawra and Govind P. Gupta. 2021. Hybrid meta-heuristic techniques based efficient charging scheduling scheme for multiple Mobile wireless chargers based wireless rechargeable sensor networks. *Peer Peer Netw. Appl.* 14, 3 (2021), 1303–1315. https://doi.org/10.1007/s12083-020-01052-8

[6] Yuxin Chen, Hejun Wu, Yongheng Liang, and Guoming Lai. 2021. VarLenMARL: A Framework of Variable-Length Time-Step Multi-Agent Reinforcement Learning for Cooperative Charging in Sensor Networks. In *18th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON 2021, Rome, Italy, July 6-9, 2021*. IEEE, 1–9. https://doi.org/10.1109/SECON52354.2021.9491594

[7] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). ACL, 1724–1734. https://doi.org/10.3115/v1/d14-1179

[8] Darolt. 2018. "Wsn simulator". https://github.com/darolt/wsn

[9] Christian Schröder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. 2020. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? *CoRR* abs/2011.09533 (2020). arXiv:2011.09533 https://arxiv.org/abs/2011.09533

[10] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 2974–2982. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17193

[11] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay D. Shet. 2014. Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6082

[12] Jayesh K. Gupta, Maxim Egorov, and Mykel J. Kochenderfer. 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In *Autonomous Agents and Multiagent Systems - AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 10642)*, Gita Sukthankar and Juan A. Rodríguez-Aguilar (Eds.). Springer, 66–83. https://doi.org/10.1007/978-3-319-71682-4_5

[13] Liang He, Peng Cheng, Yu Gu, Jianping Pan, Ting Zhu, and Cong Liu. 2014. Mobile-to-mobile energy replenishment in mission-critical robotic sensor networks. In *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*. IEEE, 1195–1203. https://doi.org/10.1109/INFOCOM.2014.6848051

[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[15] Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc V. Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. 2019. Searching for MobileNetV3. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 1314–1324. https://doi.org/10.1109/ICCV.2019.00140

[16] Shengyi Huang and Santiago Ontañón. 2020. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. *CoRR* abs/2006.14171 (2020). arXiv:2006.14171 https://arxiv.org/abs/2006.14171

[17] Chi Lin, Zhiyuan Wang, Jing Deng, Lei Wang, Jiankang Ren, and Guowei Wu. 2018. mTS: Temporal-and Spatial-Collaborative Charging for Wireless Rechargeable Sensor Networks with Multiple Vehicles. In *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*. IEEE, 99–107. https://doi.org/10.1109/INFOCOM.2018.8486402

[18] Chi Lin, Shuang Wei, Jing Deng, Mohammad S. Obaidat, Houbing Song, Lei Wang, and Guowei Wu. 2018. GTCCS: A Game Theoretical Collaborative Charging Scheduling for On-Demand Charging Architecture. *IEEE Trans. Veh. Technol.* 67, 12 (2018), 12124–12136. https://doi.org/10.1109/TVT.2018.2872890

[19] Tang Liu, Baijun Wu, Hongyi Wu, and Jian Peng. 2017. Low-Cost Collaborative Mobile Charging for Large-Scale Wireless Sensor Networks. *IEEE Trans. Mob. Comput.* 16, 8 (2017), 2213–2227. https://doi.org/10.1109/TMC.2016.2616309

[20] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 6379–6390. https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html

[21] F Oliehoek and Arnoud Visser. 2006. A hierarchical model for decentralized fighting of large scale urban fires. *ACM Transactions on Multimedia Computing, Communications, and Applications - TOMCCAP*.

[22] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer. https://doi.org/10.1007/978-3-319-28929-8

[23] Bushra Rashid and Mubashir Husain Rehmani. 2016. Applications of wireless sensor networks for urban areas: A survey. *J. Netw. Comput. Appl.* 60 (2016), 192–219. https://doi.org/10.1016/j.jnca.2015.09.008

[24] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 4292–4301. http://proceedings.mlr.press/v80/rashid18a.html

[25] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2015. Multiagent Cooperation and Competition with Deep Reinforcement Learning. *CoRR* abs/1511.08779 (2015). arXiv:1511.08779 http://arxiv.org/abs/1511.08779

[26] Ming Tan. 1993. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *In Proceedings of the Tenth International Conference on Machine Learning*. Morgan Kaufmann, 330–337.

[27] Kun Wang, Lei Wang, Chi Lin, Mohammad S. Obaidat, and Muhammad Alam. 2020. Prolonging lifetime for wireless rechargeable sensor networks through sleeping and charging scheduling. *Int. J. Commun. Syst.* 33, 8 (2020). https://doi.org/10.1002/dac.4355

[28] Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. 2006. Deploying a Wireless Sensor Network on an Active Volcano. *IEEE Internet Comput.* 10, 2 (2006), 18–25. https://doi.org/10.1109/MIC.2006.26

[29] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre M. Bayen, and Yi Wu. 2021. The Surprising Effectiveness of MAPPO in Cooperative, Multi-Agent Games. *CoRR* abs/2103.01955 (2021). arXiv:2103.01955 https://arxiv.org/abs/2103.01955

[30] Sheng Zhang, Jie Wu, and Sanglu Lu. 2015. Collaborative Mobile Charging. *IEEE Trans. Computers* 64, 3 (2015), 654–667. https://doi.org/10.1109/TC.2013.2297926