Reliable and Energy-Efficient Communications in Mobile Robotic Networks by Collaborative Beamforming

MIN HE, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China
YALI CHEN, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
MIN LIU, Network Technology Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China, and Zhongguancun Laboratory, Beijing, China
XIAOKUN FAN, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China
YUCHEN ZHU, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China

For mobile robotic networks in industrial scenarios, reliable and energy-efficient communications are crucial yet challenging. Fortunately, collaborative beamforming (CB) emerges as a promising solution, which can increase the transmission gain and reduce the transmit power of robots by constructing a mobile robot-enabled virtual antenna array (MRVAA). The performance of CB is tightly related to robot positions, necessitating proper robot selection. However, robot selection may expose the network to the risk of unbalanced energy distribution, reducing network lifetime. Additionally, the mobility and variable numbers of robots require flexible and scalable robot selection algorithms. To tackle these challenges, we first formulate a multi-objective optimization problem to reduce the maximum sidelobe level (MSLL) of MRVAA while minimizing the standard deviation of the network energy distribution (SDNED) by selecting robots for CB. Then, based on distributed multi-agent learning (MARL), we propose an effective and scalable robot selection algorithm with energy considered (RoSE) to solve the problem, where difference-rewards function (DRF) and policy sharing are designed for enhancing convergence rate and policy stability. Simulation results show that the RoSE has the scalability to positions and numbers of robots. Furthermore, RoSE surpasses existing selection algorithms in network lifetime and time efficiency, while still maintaining comparable MSLL.

This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB2900102 and in part by the National Natural Science Foundation of China under Grants 62202449 and 62072436.
Authors' Contact Information: Min He, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China; e-mail: hemin20z@ict.ac.cn; Yali Chen, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China; e-mail: chenyali@ict.ac.cn; Min Liu (Corresponding author), Network Technology Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China, and Zhongguancun Laboratory, Beijing, China; e-mail: liumin@ict.ac.cn; Xiaokun Fan, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China; e-mail: fanxiaokun19@mails.ucas.ac.cn; Yuchen Zhu, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China; e-mail: mygica123@163.com.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 1550-4859/2024/09-ART112
https://doi.org/10.1145/3678011

CCS Concepts: • **Networks** → *Network services*; • **Computing methodologies** → *Distributed computing methodologies;*

Additional Key Words and Phrases: Collaborative beamforming, distributed reinforcement learning, energy efficiency, mobile robotic networks, reliable communications

ACM Reference Format:
Min He, Yali Chen, Min Liu, Xiaokun Fan, and Yuchen Zhu. 2024. Reliable and Energy-Efficient Communications in Mobile Robotic Networks by Collaborative Beamforming. *ACM Trans. Sensor Netw.* 20, 5, Article 112 (September 2024), 24 pages. https://doi.org/10.1145/3678011

1 Introduction

The Industrial Internet is revolutionizing manufacturing by enabling efficient, personalized, and intelligent processes, representing a major trend in the future industrial field [1]. As an essential component of the Industrial Internet, mobile robots including sensors, unmanned aerial vehicles, and unmanned vehicles have significantly heightened the interest of academic and industrial communities [2]. These mobile robots connected through short-range industrial wireless communications form a network that facilitates collaborative operations. [3]. Such mobile robotic networks enable a wide spectrum of industrial applications, such as intelligent transportation [4], material handling [5], and environmental monitoring [6]. In these applications, robots are deployed in large-scale areas to collect data collaboratively and transmit the data to a controller located in the edge or the cloud for further analysis and decision-making. The reliability of these data transmissions is paramount, as any failure can lead to increased manufacturing costs and potentially catastrophic economic losses [7].

However, the limited communication capability and energy reserve of robots pose great challenges to reliable data transmissions. Especially in large-scale scenarios, robots are far from the controller [6]. Long-distance communications between robots and the controller result in severe path loss, making it incapable for robots to achieve a reliable transmission rate [8–10]. Moreover, the finite battery resources of robots cannot satisfy the energy requirement of reliable communications [11, 12]. Therefore, achieving both the desired transmission quality and high energy efficiency becomes challenging yet imperative for robots.

Fortunately, **collaborative beamforming** (**CB**) has been demonstrated as a promising technology to promote data transmission quality between transmitters and receivers, as well as the energy efficiency of transmitters [13]. In CB, several nodes individually equipped with an antenna construct a **virtual antenna array** (**VAA**). By appropriately setting the initial phase of the transmitting signal of each node, the VAA can generate a directional beam toward target receivers [14]. Theoretically, CB can improve received gain and reduce transmit power [15]. Accordingly, the data transmission quality is enhanced, and the energy consumption of data transmissions is decreased, thereby improving the reliability of communications and the energy efficiency of transmitters.

In recent years, CB has been extensively applied in **wireless sensor networks** (**WSNs**) and the research on CB optimization has attracted increasing attention [15]. The CB optimization is usually performed by reducing the **maximum sidelobe level** (**MSLL**). Considering the relationship between the MSLL and node positions, many existing studies are devoted to node selection to optimize the MSLL [16–18]. However, selecting nodes for CB may lead to premature energy depletion of some nodes, which causes a short network lifetime [19]. Although some works aim at minimizing the total energy consumption to prolong the network lifetime, uniform network energy distribution which is critical for sustainability is ignored [2, 13]. In addition, the studied scenarios in the above works are static. In mobile robotic networks, positions and numbers of robots are constantly changing, which requires frequent robot selection for CB and poses challenges to node selection algorithms with regard to time efficiency and scalability. To the best of our knowledge, none of the previous works has considered balancing the network energy distribution when selecting robots for CB in the mobile robotic network.

In this article, we study reliable and energy-efficient communications based on CB in mobile robotic networks. Then, we formulate a multi-objective optimization problem to minimize the MSLL of the **mobile robot-enabled VAA** (**MRVAA**) and the **standard deviation of the network energy distribution** (**SDNED**) by selecting robots for CB. However, this problem is challenging because of dynamic positions and varying numbers of robots. Traditional optimization approaches are intractable. Thus, we reformulate the optimization problem as a **Markov decision process** (**MDP**) and propose a distributed **multi-agent reinforcement learning** (**MARL**)-based algorithm with **difference rewards** (**DRs**) [20] and policy sharing. Our main contributions in this article are summarized as follows:

— To address long-distance communication challenges in the mobile robotic network, we establish an MRVAA utilizing CB to enhance the data transmission quality and energy efficiency of robots.
— We formulate a multi-objective optimization problem that aims to simultaneously minimize the MSLL of the MRVAA and the SDNED by optimizing the robot selection for CB. Constraints include the movement and energy of robots. To the best of our knowledge, this is the first work that optimizes CB while considering balancing the network energy distribution to prolong the network lifetime.
— We propose a distributed MARL-based robot selection algorithm with energy considered named RoSE, which is scalable to robot positions and numbers. Specifically, a **difference-rewards function** (**DRF**) is devised to evaluate the contribution of each robot to obtain better policies. Furthermore, robots share policies to adapt to unstable environments caused by distributed learning and accelerate training convergence. To the best of our knowledge, this is the first application of MARL for CB optimization.
— The simulation results indicate that the proposed RoSE algorithm achieves the best system performance in considered algorithms. Moreover, RoSE markedly reduces the SDNED with low time overhead in dynamic scenarios, which justifies the superiority of the proposed algorithm in terms of prolonging the network lifetime and time efficiency. Compared with the benchmark algorithm with the best system performance, RoSE has a similar MSLL, but it can reduce the SDNED by 2.70% and the time consumption by 66.57%.

The remainder of this article is organized as follows. In Section 2, we review related works of the CB optimization by node selection. In Section 3, the system model of a CB-based mobile robotic network is presented. In Section 4, we formulate a multi-objective optimization problem to optimize the CB performance and the network lifetime. In Section 5, we propose a distributed MARL-based algorithm to solve this problem. Extensive simulations and performance evaluation are given in Section 6. Finally, we conclude this article in Section 7.

2 Related Work

In this article, we focus on optimizing the MSLL of CB and the network lifetime based on node selection. Other CB optimization approaches such as mobile node placement and coefficient perturbation are beyond the scope of this article, and please refer to [15, 21] and [22]. The node selection algorithms adopted in related works can be broadly classified into *regular topology-based algorithms* and *heuristics-based algorithms*.

2.1 Regular Topology-based Algorithms

The basic idea of regular topology-based algorithms is optimizing the CB performance by selecting nodes to form the antenna array with special geometric shapes, such as the line, circle, and sector.

In [16], the authors took a **virtual linear antenna array** (**VLA**) as a reference and selected nodes to minimize the MSLL of CB in the WSN. Similarly, the authors in [23] adopted **particle swarm optimization** (**PSO**) for node selection to approximate a linear antenna array, aiming at reducing the MSLL of CB. In [24], the authors selected the nodes to form a uniform circular array with the objective of minimizing the MSLL. Considering the concentric circular antenna array as the reference, the authors in [25] proposed a hybrid optimization approach to minimize the MSLL and energy consumption of the CB-based WSN by jointly optimizing node selection and the transmit power of the selected nodes. The authors in [26] proposed a potential game theory-based algorithm to jointly optimize node selection and the transmit power of the selected nodes. The goals were minimizing the MSLL of sensor-enabled VAA and balancing the energy consumption of the selected nodes. In [27], the authors proposed a hybrid least square PSO algorithm. Nodes were selected to shape a circular antenna array for obtaining a beam pattern with the minimum **sidelobe level** (**SLL**) and controllable size of the first null beam width. To minimize the MSLL, the authors in [17] selected nodes to construct a concentric circular ring array. With the goal of attaining a beam pattern with a narrow mainlobe and low SLLs, the authors in [28] randomly selected nodes from a few fixed sector regions.

2.2 Heuristics-based Algorithms

This category of node selection approaches adopts heuristic algorithms to optimize the MSLL by acquiring the optimal node combination for CB. The authors in [13] modified the traditional cuckoo search algorithm and proposed a discrete cuckoo search algorithm to minimize the MSLL and improve the energy efficiency of sensor-enabled VAA with energy consumption constraints. In [18], the authors studied a CB-based WSN, where multiple BSs were deployed in neighboring areas. A greedy algorithm was proposed to optimize nodes for CB for minimizing SLLs at unintended BSs with the complexity constraint. Similarly, the authors in [29] proposed a central scheduling algorithm to control node selection and the order of communication with the BSs or **access points** (**APs**), which aims at achieving a beam pattern with low SLLs toward the unintended BSs or APs. Considering a CB-based WSN with multiple receivers, the authors in [30] proposed a node selection algorithm based on the decentralized cross-entropy optimization to minimize SLLs at unintended receivers.

According to the above-related works, we have the following observations:

- Different from the static scenarios involved in the above works, we focus on a dynamic scenario where positions and numbers of robots are variable.
- The above works neglect the network energy distribution when selecting nodes. However, balanced network energy distribution is crucial for network lifetime, especially for robots with limited energy reserve in mobile robotic networks.
- Regular topology-based algorithms mentioned above cannot guarantee the effectiveness of CB optimization under any node distribution. Heuristics-based algorithms are more likely to obtain the optimal solution, but their scalability is poor. When used in mobile scenarios, heuristics-based algorithms require a complete rerun once there are changes in node positions or numbers, resulting in low time efficiency. Thus, we are devoted to designing a node selection algorithm with high scalability and time efficiency.

3 System Models

As shown in Figure 1, we consider a mobile robotic network in intelligent manufacturing scenarios for real-time monitoring of industrial conditions. It consists of a cluster of robots (e.g., automated guided vehicles) represented by the set $\mathcal{K} = \{1, 2, \ldots, K\}$. These robots are dispatched to inspect

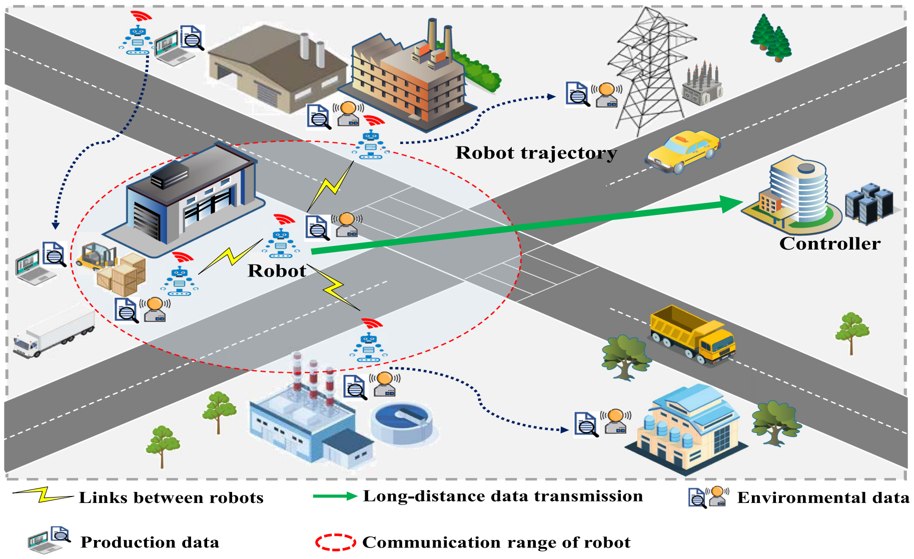

Fig. 1. System model of the mobile robotic network.

Table 1. Summary of Key Notations

Notations	Description
$\mathcal{K}, k$	Set and index of mobile robots
$\hat{k}$	Index of the mobile robot demanding data transmissions
$\mathcal{I}, i$	Set and index of time slots
$\tau_i, \tau_i^{mov}, \tau_i^{cb}$	Duration of each time slot, movement and data transmission duration of robots in each time slot
v_i^k, ζ_i^k	Movement velocity and direction of robot k at time slot i
$(r_i^k, \phi_i^k, \theta_i^k)$	Position of robot k at time slot i
ϕ_i^{ML}, ϕ_i^{SL}	Directions of MSLL and sidelobe at time slot i
$\phi_i^{FN1}, \phi_i^{FN2}$	Angles of two first nulls around the mainlobe at time slot i
p_i^k	Motion power consumption of robot k at time slot i
$\Delta E_{i,m}^k, \Delta E_{i,d}^k$	Motion and data transmission energy consumption of robot k at time slot i
E_i^k	Residual energy of robot k at time slot i
E_i	Network energy distribution at time slot i
f_i^1	MSLL of the MRVAA at time slot i
f_i^2	SDNED of the mobile robotic network at time slot i

factory facilities to collect essential data on the environment and production, such as air quality, pollution level, hazardous material leaks, and equipment status. The collected data is subsequently transmitted to a remote controller for further analysis to guide environment adjustments, maintenance scheduling and emergency responses, ensuring production safety and efficiency. Specifically, each robot is equipped with an omnidirectional antenna with the communication range of R and can obtain its position via GPS, anchor nodes, seed nodes, and so on [31]. Additionally, the carrier frequency, time, and initial phase of all robots are synchronized [30]. For ease of reference, Table 1. summarizes the key notations used in this article.

In the studied time horizon, we discretize the timeline of the system into I time slots denoted as $\mathcal{I} = \{1, 2, \ldots, I\}$, as shown in Figure 2. Within the time slot $i \in \mathcal{I}$ with length τ_i, all robots collect data and transmit the data by CB. Concretely, robots first move to collect data with time

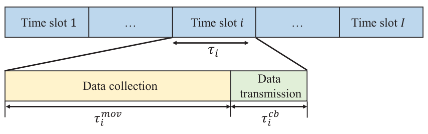

Fig. 2. Timeline of the system.

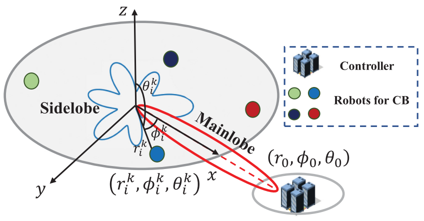

Fig. 3. Sketch map of the CB-based mobile robotic network.

duration τ_i^{mov}. In the subsequent time duration τ_i^{cb}, there exists a robot $\hat{k} \in \mathcal{K}$ which demands transmitting the data to the controller. Then, the mobile robotic network selects suitable robots and the selected robots transmit the data collaboratively. To effectively perform beamforming and achieve reliable communications with the controller, we suppose that all robots are static during robot selections and data transmissions [32].

Figure 3 illustrates the geometrical configuration of the CB-based robotic network. For analytical convenience, we use the polar coordinates to represent the positions of robots and the controller. Then, the position of robot k, $\forall k \in \mathcal{K}$, at time slot i is denoted by $(r_i^k, \phi_i^k, \theta_i^k)$, where r_i^k is the Euclidean distance and $\phi_i^k \in [0, 2\pi)$ is the azimuth angle. θ_i^k is the elevation angle and equals $\frac{\pi}{2}$ because robots are deployed on the ground. Meanwhile, the position of the controller is denoted by (r_0, ϕ_0, θ_0). For generality, we set $\phi_0 = 0$ and $\theta_0 = \frac{\pi}{2}$ [13].

3.1 Robot Mobility Model

In the cartesian coordinates, the position of robot k at time slot i is expressed as (x_i^k, y_i^k), where $x_i^k = r_i^k \sin\theta_i^k \cos\phi_i^k$ and $y_i^k = r_i^k \sin\theta_i^k \sin\phi_i^k$. Note that the movement of robots is restricted to a rectangular area with side length X_{max} and Y_{max}. In time slot i, robot k moves with the radial direction $\zeta_i^k \in [0, 2\pi)$ and the velocity $v_i^k \in [0, V_{max}]$. Given τ_i^{mov}, the position of robot k at time slot i is updated as [33]

$$x_i^k = x_{i-1}^k + v_i^k \tau_i^{mov} \cos\zeta_i^k, \tag{1a}$$

$$y_i^k = y_{i-1}^k + v_i^k \tau_i^{mov} \sin\zeta_i^k. \tag{1b}$$

The corresponding trajectory of robot k during the task duration I can be characterized by the set $\{(x_i^k, y_i^k)\}_{i=1}^I$.

3.2 CB Model

Since all robots and the controller are in the same plane, we just consider the beam pattern in a plane [34]. To collaboratively transmit the data, robot $\hat{k}$ needs to share it with other robots. However, data sharing with robots outside the communication range requires multi-hop communications, which leads to high communication overhead and delay. Therefore, only neighbor robots within the communication range of robot $\hat{k}$ are considered to participate in CB. Robot $\hat{k}$ and the set of neighbor robots at time slot i are denoted by $N_i^{\hat{k}}$.

Assume that $C_i^{\hat{k}} \subseteq N_i^{\hat{k}}$ denotes the set of n robots for CB at time slot i, the **array factor** (**AF**) for characterizing the beam pattern of the MRVAA is given by [14, 29]

$$AF(\phi, i) = \frac{1}{n} \sum_{k \in C_i^{\hat{k}}} e^{j\frac{2\pi}{\lambda} r_i^k [\cos\phi_i^k - \cos(\phi - \phi_i^k)]}, \tag{2}$$

where λ is the wavelength of the radio frequency carrier. Obviously, the value of $|AF|$ is maximized when $\phi = 0$, that is, the azimuth angle of the mainlobe is $\phi_i^{ML} = 0$ which points to the controller as shown in Figure 3. Furthermore, the MSLL at time slot i is given by

$$f_i^1 = \max_{\phi_i^{SL}} |AF(\phi_i^{SL}, i)|, \tag{3}$$

where $\phi_i^{SL} \in (\phi_i^{FN1}, \phi_i^{FN2})$ is the direction of sidelobe, and ϕ_i^{FN1} and ϕ_i^{FN2} are angles of the two first nulls around the mainlobe, respectively.

3.3 Network Lifetime

All robots begin executing the task with fully charged batteries and maintain the same initial energy level, denoted as E_0. It is assumed that robots cannot charge or replace batteries during the task. To ensure that robots with low energy levels can exit the network timely and move to charging stations for power replenishment, we set an energy threshold pE_0, where $p \in [0, 1]$ is the percentage coefficient.

The energy consumption of the robot during the task is mainly caused by the motion and data transmissions. Our analysis begins with the model of the motion energy consumption for robots. At time slot i, the motion power consumption of the robot k with velocity v_i^k is given by [35]

$$P_i^k = \begin{cases} p_1 v_i^k + p_2, & 0 < v_i^k \leq V_{max}, \quad (4a) \\ 0, & v_i^k = 0, \quad (4b) \end{cases}$$

where p_1 and p_2 are constant motion parameters. Correspondingly, the motion energy consumption in the time slot i is $\Delta E_{i,m}^k = P_i^k \tau_i^{mov}$.

For the energy consumption of data transmissions, it is supposed that robots in $C_i^{\hat{k}}$ transmit l_i bits data to the controller with the same transmit power in time slot i. Then, the data transmission energy consumption of each robot in $\mathcal{K}$ is written as [13, 36, 37]

$$\Delta E_{i,d}^k = \begin{cases} l_i \Big(e_{cct} + e_{tx} \dfrac{{r_0}^\alpha}{n^2} \Big), & k \in C_i^{\hat{k}}, \quad (5a) \\ 0, & k \in \mathcal{K} - C_i^{\hat{k}}, \quad (5b) \end{cases}$$

where e_{cct} is the electronics energy, e_{tx} is the amplifier energy parameter and α is the path loss exponent of the link between the robot and the controller.

Next, we discuss the lifetime of robotic networks. Considering the limited battery capacity of robots, extending the lifespan of robotic networks is significant [38]. To achieve this, it is not simply to reduce the total energy consumption, but more importantly to balance the network energy distribution [3]. Thus, we adopt the SDNED to evaluate the energy balancing degree of the network. A smaller SDNED means a more balanced energy distribution and a longer network lifetime. Considering the total energy consumption of the robot, the residual energy of robot k at time slot i is updated as

$$E_i^k = E_{i-1}^k - \Delta E_{i,m}^k - \Delta E_{i,d}^k, \tag{6}$$

and we obtain the network energy distribution

$$E_i = \{E_i^k | k \in \mathcal{K}\}. \tag{7}$$

Then, we calculate the SDNED by

$$f_i^2 = SD(E_i) = \sqrt{\frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} (E_i^k - \mu_i)^2}, \tag{8}$$

where μ_i is the mean value of the network energy distribution E_i.

4 Problem Formulation

The objective of this article is to jointly minimize the average MSLL and SDNED in I time slots by optimizing robot selection decision $C_i^{\hat{k}}$. Accordingly, the multi-objective optimization problem is formulated as

$$
\begin{aligned}
\mathbf{P1}: \quad & \min_{C_i^{\hat{k}}} \quad \frac{1}{I}\sum_{i\in\mathcal{I}}\left(wf_i^1+(1-w)f_i^2\right) && \text{(9a)}\\
& s.t. \quad x_i^k \in [0, X_{max}],\ \forall\, i, \forall\, k, && \text{(9b)}\\
& \qquad\ \ y_i^k \in [0, Y_{max}],\ \forall\, i, \forall\, k, && \text{(9c)}\\
& \qquad\ \ \phi_i^{ML} = \max_{\phi} |AF(\phi, i)|, \phi \in [0, 2\pi), \forall\, i, && \text{(9d)}\\
& \qquad\ \ f_i^1 = \arg\max_{\phi_i^{SL}} \left|AF(\phi_i^{SL}, i)\right|, \phi_i^{SL} \in \left(\phi_i^{FN1}, \phi_i^{FN2}\right), \forall\, i && \text{(9e)}\\
& \qquad\ \ E_i^k \geq pE_0,\ \forall\, i, \forall\, k, && \text{(9f)}
\end{aligned}
$$

where ω is the weight of the objective f_i^1. Constraints (9b)–(9c) limit the motion of robots. Constraint (9d) indicates the direction of the mainlobe. Constraint (9e) describes the relationship between the direction of sidelobe ϕ_i^{SL} and the MSLL f_i^1. Constraint (9f) restricts the residual energy of robots in the network.

For the problem **P1**, the optimization process is to explore the combinatorial space of robots participating in CB. Assuming $|N_i^{\hat{k}}| = m$, then there are 2^m combinations, and the time complexity is $O(2^m)$. So the problem **P1** is a NP-hard problem.

5 Proposed Solution: RoSE

In this section, we first give some preliminaries to the MARL. Then, we provide an overview of the proposed algorithm RoSE. Finally, we introduce details of the RoSE.

5.1 Preliminaries

MARL refers to using **reinforcement learning** (**RL**) methods to construct a system where multiple agents interact in the same environment, thereby solving sequential decision-making. A mainstream MARL framework is **centralized training and distribution execution** (**CTDE**) [20]. Although the CTDE provides execution-efficient learning, the joint state and action space of centralized training will exhibit exponential expansion as the number of agents increases. This brings great difficulties to explorations. One way to reduce the dimension of the joint state and action space is distributed MARL. This method enables each agent to conduct independent offline training and online decision-making [39]. However, distributed MARL faces the following challenges: (i) during learning, the reward information given by the environment is a global reward. How to assess the contribution of different robots and assign the global reward (known as credit assignment) is a significant issue. Credit assignment can affect the convergence and optimality of policies. To tackle the credit assignment problem, DRs is proposed in which each agent learns based on a reward from its marginal contribution to the team reward [20]. Despite the DRs is a powerful way to perform the credit assignment, it only provides a conceptual framework and there is no general computational approach to compute DRs in different settings [40]. (ii) The policies of other robots are uncertain and changing during the learning progress, which results in an unstable learning environment, loss of collaboration advantages, and degraded system performance [41]. Fortunately, learning with experience exchanging or policy sharing through communication has been proposed to tackle the above issues [42].

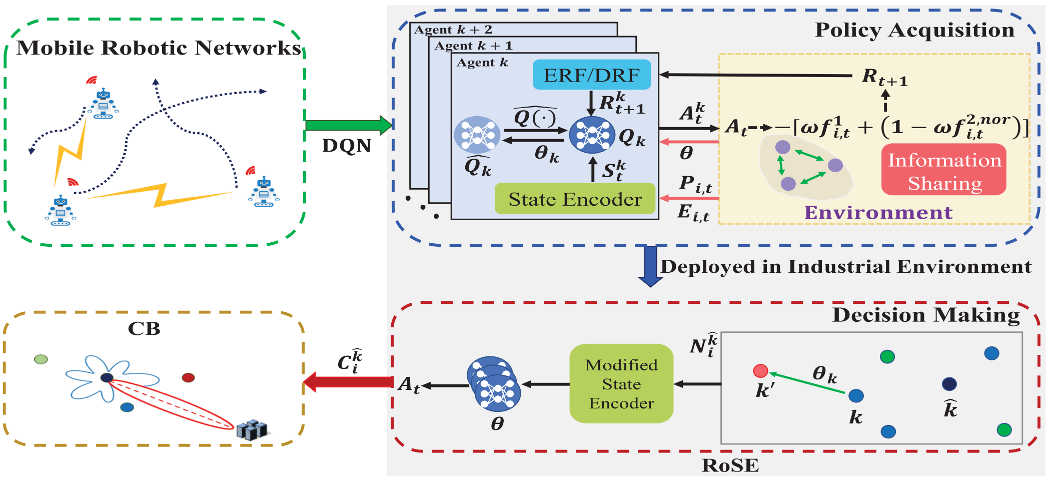

Fig. 4. Procedure of RoSE. It contains two stages: Policy Acquisition and Decision Making.

5.2 RoSE Overview

Since the problem **P1** is NP-hard, most of the existing algorithms are heuristic to get the solution in polynomial time. However, these algorithms are not applicable to mobile robotic networks. Because once the positions or numbers of robots have changed, they must be executed from scratch, leading to low time efficiency. Fortunately, distributed MARL has the advantage of coping with the dynamics and uncertainties of mobile robotic networks. When parameter settings change, there is no need to retrain and the learned policies can be directly used. Therefore, we propose a distributed MARL-based algorithm called RoSE to solve problem **P1**. RoSE includes two stages: *Policy Acquisition* and *Decision Making* as shown in Figure 4.

— **Policy Acquisition.** Each robot in the robotic network is an agent. Agents take actions that maximize their individual rewards based on their local states, with the overarching goal of minimizing the MSLL and SDNED. During training, agents share policies with each other. Upon completion of training, each agent will obtain its optimal policy.

— **Decision Making.** When robot $\hat{k}$ requires data transmission, each robot in $N_i^{\hat{k}}$ encodes its states according to the current number of robots in the network. Based on the states, they make decisions using learned policies. This process culminates in obtaining the subset of robots that will participate in CB.

5.3 Policy Acquisition

In this subsection, we first formalize the sequential decision-making of the robot into an MDP, and then discuss how to encode the robot's local states. Next, the learning scheme with DRF and learned policy sharing is designed to tackle issues of the distributed MARL. Lastly, we show the scalability of learned policies to the positions of robots. The procedure of Policy Acquisition is outlined in Figure 4 and Algorithm 1.

5.3.1 MDP for A Single Robot. MDP is a formalization of the RL environment, and almost all RL environments can be described by the MDP. Here, robot k is an agent whose MDP is defined as a tuple $< S^k, A^k, P_r^k, R^k, \gamma >$, where

— S^k is the local state set of agent k and the state at step t is denoted as S_t^k.

— $A^k = \{0, 1\}$ is the action set of the agent k. If agent k participates in CB at step t, then $A_t^k = 1$ else $A_t^k = 0$.

Algorithm 1: Policy Acquisition

```
Input: maximum distance from the origin $r_{max}$, initial residual energy $E_0$, agent set $\mathcal{K}$, reward
       function $f$, network update cycle $U$
Output: set of model parameters $\theta$
1  Set the sum of step $O = 0$ and initialize replay memory $D_k = \emptyset, \forall k \in \mathcal{K}$;
2  Initialize $\theta_k$ randomly and set $\theta_k^- = \theta_k, \forall k \in \mathcal{K}$;
3  for each episode $i$ do
4      for agent $k = 1, 2, \ldots, K$ do
5          Initialize the agent position $(r_i^k, \phi_i^k, \frac{\pi}{2})$ by randomly sampling from $(0, r_{max}]$ and $(0, 2\pi)$;
6          Initialize $E_{i,0}^k = E_0$ and $A_0^k = 1$;
7      for step $t = 1, 2, \ldots, t_{max}$ do
8          $O = O + 1$;
9          for agent $k = 1, 2, \ldots, K$ do
10             Get $S_t^k$ with Equation (17) and select action $A_t^k$ using Equation (28);
11         Take action $A_t$ on the environment;
12         for agent $k = 1, 2, \ldots, K$ do
13             Update $E_{i,t+1}^k$ with Equation (31);
14             Calculate rewards $R_{t+1}^k$ according to the reward function $f$;
15             Obtain new state $S_{t+1}^k$ and store $(S_t^k, A_t^k, R_{t+1}^k, S_{t+1}^k)$ into buffer $D_k$;
16             Sample a batch of $(S_j^k, A_j^k, R_{j+1}^k, S_{j+1}^k)$ from buffer $D_k$;
17             Update $\theta_k$ by minimize $L_{TD}$;
18             if $O$ mod $U = 0$ then
19                 Transmit $\theta_k$ to other agents;
20                 Receive other agents' model parameters and update $\theta_k^-$ with Equation (29).
```

— P_r^k is the state transition probability.
— R^k is the reward function. After taking action A_t^k, the environment sends reward R_{t+1}^k to agent k.
— γ is the discount factor.

The goal of problem **P1** is to seek out the combinations that minimize the formula (9a) in the robot combination space. We formalize this process as sequential decision-making for RL which finds the optimal robot combination for every time slot. Specifically, each time slot serves as an episode. The goal of agents during episode i is to minimize the $\omega f_i^1 + (1 - \omega) f_i^2$. To realize the above purposes, we need to elaborately devise the state encoding and the reward function of agents.

5.3.2 *State Encoding.* According to the objective of problem **P1**, the agent needs to take both the position and residual energy into account when it encodes the state. Whereas, only based on local information, the agent cannot learn effective policies. Fortunately, at step t of the episode i, the positions $P_{i,t} = \{(r_{i,t}^1, \phi_{i,t}^1, \frac{\pi}{2}), (r_{i,t}^2, \phi_{i,t}^2, \frac{\pi}{2}), \ldots, (r_{i,t}^K, \phi_{i,t}^K, \frac{\pi}{2})\}$ and the residual energy information of all agents $E_{i,t} = \{E_{i,t}^1, E_{i,t}^2, \ldots, E_{i,t}^K\}$ can be shared between agents as the global information. Since agents are static during the agent selection, we have $(r_{i,t}^k, \phi_{i,t}^k, \frac{\pi}{2}) = (r_i^k, \phi_i^k, \frac{\pi}{2})$. Moreover, we set the residual energy information $E_{i,0}^k$ of agent k at step 0 as the agent's initial residual energy. Agents encode local states following the global information. The states include two parts: the *CB Part* and the *Energy Part*.

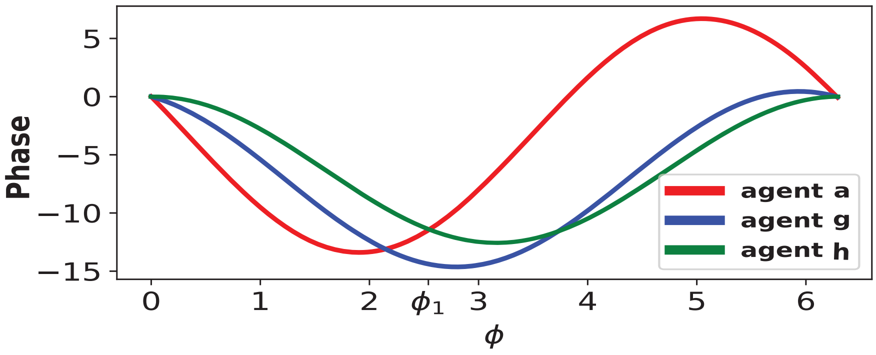

Fig. 5. Phase values of agents a, g, and h.

(1) CB Part: According to the Euler formula, Equation (2) at step t can be written as

$$AF(\phi, t) = \frac{1}{n}\left(\sum_{k \in C_t} \cos \alpha_i^k + \sum_{k \in C_t} j \sin \alpha_i^k\right), \tag{10}$$

$$|AF(\phi, t)| = \frac{1}{n}\sqrt{\left(\sum_{k \in C_t} \cos \alpha_i^k\right)^2 + \left(\sum_{k \in C_t} \sin \alpha_i^k\right)^2}, \tag{11}$$

where C_t is the set of agents selected for CB at step t and $\alpha_i^k = \frac{2\pi}{\lambda} r_i^k[\cos \phi_i^k - \cos(\phi - \phi_i^k)]$. With a fixed value of ϕ, AF can be regarded as the addition of n vectors in the complex plane, and the contribution of the selected agent k is the vector $B_i^k = (\cos \alpha_i^k, \sin \alpha_i^k)$, where $|B_i^k| = 1$ and the phase is $\beta_i^k = \tan \alpha_i^k$. According to Equations (10) and (11), when $\phi = 0$, all vectors are collinear and $|AF| = 1$. Therefore, the direction of the controller is the mainlobe and $|AF|$ is the largest.

When two vectors with the fixed module are added, the smaller the angle between vectors, the larger the result. Thus, we use the minimum phase difference between vectors $\{B_i^k | k \in C_t\}$ in $\phi \in (0, 2\pi)$ to characterize the CB part of the agent's state. As an example, suppose there are three agents a, g, h for CB, and Figure 5 shows the phase value of the three agents. When $\phi = \phi_1$, agent h obtains the minimum sum of the phase difference with the agent a and g, that is

$$\phi_1 = \arg\min_{\phi} \left(\left|\beta_i^h - \beta_i^a\right| + \left|\beta_i^h - \beta_i^g\right|\right). \tag{12}$$

Then, the state of agent h is encoded as $(|\beta_i^h - \beta_i^a|, |\beta_i^h - \beta_i^g|)$ with $\phi = \phi_1$.

However, if only the agents for CB are considered, the dimension of states changes at each step. Therefore, we include all agents. Note that if the phase difference of two vectors with the same module is π, then the vectors add up to 0. Hence, for the agent not participating in CB, we set its phase difference with other agents as π. This ensures that the result of AF is not affected. Moreover, considering the sequential decision-making for RL, the state at step t should be encoded based on the decision C_{t-1} at step $t-1$. According to the above analysis, the minimum phase difference $\Delta\beta_{i,t}^{k,k'}$ between the agent k and the agent k' ($k' \neq k$) at step t is calculated by

$$\Delta\beta_{i,t}^{k,k'} = \begin{cases} |\beta_i^k - \beta_i^{k'}|, & k' \in C_{t-1}, \phi = \phi_{min}, & (13a) \\ \pi, & k' \in \mathcal{K} - C_{t-1}, & (13b) \end{cases}$$

where $\phi_{min} = \min_{\phi} \sum_{k' \in C_{t-1}} |\beta_i^k - \beta_i^{k'}|$. Then, the CB part of the agent k's state at step t is composed of the phase difference between agent k and other $|\mathcal{K}| - 1$ agents,

$$S_{t,cp}^k = \left(\Delta\beta_{i,t}^{k,1}, \dots, \Delta\beta_{i,t}^{k,k-1}, \Delta\beta_{i,t}^{k,k+1}, \dots, \Delta\beta_{i,t}^{k,K}\right). \tag{14}$$

(2) Energy Part: We encode energy states utilizing residual energy difference, which can characterize the unique state of each robot compared to the direct use of residual energy. The residual energy difference between the robot k and other robots at the beginning of step t can be written as

$$\Delta E_{i,t}^{k,k'} = E_{i,t-1}^{k'} - E_{i,t-1}^k, k' \in \mathcal{K}, k' \neq k. \tag{15}$$

Then, the energy part of the agent k's state at step t is defined as

$$S_{t,ep}^k = \left(\Delta E_{i,t}^{k,1}, \dots, \Delta E_{i,t}^{k,k-1}, \Delta E_{i,t}^{k,k+1}, \dots, \Delta E_{i,t}^{k,K}\right). \tag{16}$$

Finally, the state of agent k at step t is given by

$$S_t^k = \left[S_{t,cp}^k, S_{t,ep}^k\right]. \tag{17}$$

5.3.3 Learning Schemes. To solve the credit assignment challenge of the distributed MARL, we design the DRF. Whilst, we design the **equal-rewards function (ERF)** as a comparison to verify the effectiveness of the DRF, where every agent has the same reward. Through reward signals, agents are incentivized to minimize the $\omega f_i^1 + (1-\omega) f_i^2$.

(1) ERF: At the step t of episode i, the objective f_i^1 is denoted as $f_{i,t}^1$ and is given by

$$f_{i,t}^1 = \max_{\phi_i^{SL}} \left| \frac{1}{\sum_{k \in \mathcal{K}} A_t^k} \sum_{k \in \mathcal{K}} A_t^k e^{j \frac{2\pi}{\lambda} r_i^k [\cos \phi_i^k - \cos(\phi_i^{SL} - \phi_i^k)]} \right|. \tag{18}$$

Notice that objective $f_i^1 \in [0,1]$ and objective f_i^2 are performance metrics with varying orders of magnitude. To deal with it, we conduct min-max normalization to the network energy distribution and it is calculated by

$$\begin{aligned} E_{i,t}^{nor} &= \left\{e_{i,t}^k | k \in \mathcal{K}\right\}, \\ e_{i,t}^k &= \frac{E_{i,t}^k - E_{i,t}^{min}}{E_{i,t}^{max} - E_{i,t}^{min}}, \end{aligned} \tag{19}$$

where $E_{i,t}^{min}$ and $E_{i,t}^{max}$ are the minimum and maximum residual energy of agents at step t, respectively. Based on the above analysis, the normalized objective f_i^2 at step t, denoted as $f_{i,t}^{2,nor}$, is given by

$$f_{i,t}^{2,nor} = SD\left(E_{i,t}^{nor}\right). \tag{20}$$

The minimization objective in problem **P1** is attained by maximizing the reward in MARL. As a result, the global reward R_{t+1} at step t is calculated as

$$R_{t+1} = -\left[\omega f_{i,t}^1 + (1-\omega) f_{i,t}^{2,nor}\right]. \tag{21}$$

Then, the ERF is defined as the function $f_{ERF} : R_{t+1} \rightarrow [R_{t+1}^1, R_{t+1}^2, \dots, R_{t+1}^K]$, in which $R_{t+1}^1 = R_{t+1}^2 = \cdots = R_{t+1}^K = R_{t+1}$.

(2) DRF: The DRF considers the marginal contribution of the agent. More precisely, the marginal contribution of the agent k to the objective f_i^1 at step t is given by

$$f_{i,t}^{1,k} = \max_{\phi_i^{SL}} \left| \frac{1}{A_t^k + A_{t-1}^{-k}} \left(RP_t^k + RP_{t-1}^{-k}\right) \right|, \tag{22}$$

where $A_{t-1}^{-k} = \sum_{k' \in \mathcal{K}, k' \neq k} A_{t-1}^{k'}$, $RP_t^k = A_t^k e^{j\frac{2\pi}{\lambda} r_i^k [\cos\phi_i^k - \cos(\phi_i^{SL} - \phi_i^k)]}$ denotes the contribution of agent k to the objective f_i^1 at step t. $RP_{t-1}^{-k} = \sum_{k' \in \mathcal{K}, k' \neq k} A_{t-1}^{k'} e^{j\frac{2\pi}{\lambda} r_i^{k'} [\cos\phi_i^{k'} - \cos(\phi_i^{SL} - \phi_i^{k'})]}$ denotes the contributions of all agents except agent k to the objective f_i^1 at step $t-1$.

For the objective f_i^2, we take the residual energy of agent k as the baseline and get the updated network energy distribution

$$\begin{aligned} E_{i,t}^{-k} &= \left\{ E_{i,t}^{-k,k'} | k' \in \mathcal{K}, k' \neq k \right\}, \\ E_{i,t}^{-k,k'} &= E_{i,t-1}^{k'} - E_{i,t}^{k}. \end{aligned} \tag{23}$$

Similar to the ERF, we conduct min-max normalization to the network energy distribution $E_{i,t}^{-k}$ and the normalized network energy distribution is

$$\begin{aligned} E_{i,t}^{-k,nor} &= \left\{ e_{i,t}^{-k,k'} | k' \in \mathcal{K}, k' \neq k \right\}, \\ e_{i,t}^{-k,k'} &= \frac{E_{i,t}^{-k,k'} - E_{i,t}^{-k,min}}{E_{i,t}^{-k,max} - E_{i,t}^{-k,min}}, \end{aligned} \tag{24}$$

where $E_{i,t}^{-k,min}$ and $E_{i,t}^{-k,max}$ are the minimum and maximum values in $E_{i,t}^{-k}$, respectively. Then, the contribution of agent k to the normalized objective f_i^2 at step t can be written as

$$f_{i,t}^{2,k} = SD\left(E_{i,t}^{-k,nor}\right). \tag{25}$$

Based on the above analysis, the reward of agent k at step t is defined as

$$R_{t+1}^k = -\left[\omega f_{i,t}^{1,k} + (1-\omega) f_{i,t}^{2,k}\right]. \tag{26}$$

Lastly, the DRF is defined as the function $f_{DRF} : R_{t+1} \to [R_{t+1}^1, R_{t+1}^2, \ldots, R_{t+1}^K]$.

(3) Independent DQN: Each agent adopts DQN [43] as the RL algorithm and learns independently based on its local states. The DQN of agent k includes two neural networks: Q-function network Q_k and target Q-function network $\hat{Q}_k$. DQN trains the agent by minimizing the **temporal-difference** (**TD**) loss of the Q-function network. The TD loss of the agent k at step t is calculated as follows

$$L_{TD} = \left(R_{t+1}^k + \gamma \max_{A_{t+1}^k} \hat{Q}_k\left(S_{t+1}^k, A_{t+1}^k; \theta_k^-\right) - Q_k\left(S_t^k, A_t^k; \theta_k\right) \right)^2, \tag{27}$$

where θ_k^- and θ_k are model parameters of $\hat{Q}_k$ and Q_k, respectively. Agent k takes action based on Q value and follows ϵ-greedy policy, that is,

$$A_t^k = \begin{cases} \max\limits_{A_t^k} Q_k(S_t^k, A_t^k), & \text{with probability } 1-\epsilon,, \quad (28a) \\ \text{random } A_t^k \in A^k, & \text{with probability } \epsilon, \quad (28b) \end{cases}$$

where $\epsilon \in [0,1]$ is a random number. Moreover, the experience replay buffer stores historical transition tuples. During training, a batch of transitions is sampled from the experience replay buffer to train the neural networks, effectively reducing the data correlation. Typically, the size M of the experience replay buffer in RL is set as the scale of thousands (K). Considering the memory constraints of robots, we have chosen a reduced buffer size [37].

To solve problems caused by an unstable environment, the agent will share policy with other agents during learning. In conventional DQN, the model parameters θ_k^- will be updated to the model parameters θ_k every U step. At this time, the θ_k is the optimal model parameter, and we design agent k sending θ_k to other agents. After receiving the model parameters of other agents, agent k updates the $\hat{Q}_k$ with model parameters

$$\theta_k^- = \frac{\sum_{k' \in \mathcal{K}} \theta_{k'}}{|\mathcal{K}|}. \tag{29}$$

The proposed Policy Acquisition is illustrated in detail as shown in the Algorithm 1. At the start of training, we initialize the experience replay memory $D_k = \emptyset$ (Line 1). Subsequently, the Q-network of agent k is initialized with random parameters θ_k and these parameters are then copied into the target Q-network (Line 2). During the exploration stage (Lines 3–15), the agent k selects an action A_t^k based on its local observed state S_t^k. All actions of agents form a joint action A_t, which acts on the environment. This interaction prompts a state transition and gives rise to a reward R_{t+1}^k for the agent k. The corresponding transition tuple $(S_t^k, A_t^k, R_{t+1}^k, S_{t+1}^k)$ is then stored in the experience replay buffer D_k. In the update stage (Lines 16–17), a batch of transitions is randomly sampled from D_k to train the Q-network based on the TD loss. Every U step, agents share policies and the target Q-network is updated using the Q-network's parameters (Lines 18–20). Finishing learning, we save the Q-function network's model parameters of all agents as

$$\theta = \{\theta_k | k \in \mathcal{K}\}. \tag{30}$$

5.3.4 Scalability to Robot Positions. RoSE's scalability to robot positions allows the policies obtained to be directly applied without retraining, even when the robots' positions change. To achieve this property, the position of each robot is initialized randomly at the beginning of each episode and remains unchanged during learning, as shown in the 5th step of Algorithm 1. Such scalability to positions ensures RoSE's applicability in our scenario where there exists a relationship between robots' positions across different episodes. Based on the aforementioned setting, agents only have the energy consumption of data transmissions during episode i. Besides, the energy consumption of data sharing between agents can be negligible. Therefore, we mainly consider the CB energy consumption and update the residual energy of agent k by

$$E_{i,t}^k = E_{i,t-1}^k - \Delta E_{t,d}^k, \tag{31}$$

where $\Delta E_{t,d}^k$ can be calculated by Equation (5) with $C_i^{\hat{k}} = C_t$.

5.4 Decision Making

5.4.1 Policy Execution. At time slot i, robot $\hat{k}$ requires data transmission, and then the robots in $N_i^{\hat{k}}$ use their learned policies to make decisions. We set the global information $P_{i,0}$ at step 0 as P_i, where P_i is the set of robot positions at time slot i. Moreover, the residual energy information $E_{i,0}$ at step 0 is set as the network energy distribution after robots movement at time slot i, that is

$$E_{i,0} = \left\{E_{i-1}^1 - \Delta E_{i,m}^1, E_{i-1}^2 - \Delta E_{i,m}^2, \ldots, E_{i-1}^K - \Delta E_{i,m}^K\right\}. \tag{32}$$

Let those robots execute the decision-making t_{max} times and we choose the robot set with the minimum objective $\omega f_{i,t}^1 + (1-\omega) f_{i,t}^{2,nor}$. The procedure of Decision Making is outlined in Figure 4 and Algorithm 2.

5.4.2 Scalability to Robot Numbers. According to Equation (17), the state dimension of the trained policies is fixed as $2(|\mathcal{K}| - 1)$. When the number of robots $|\mathcal{K}|$ in the network changes due to running out of power, breakdowns, or robot addition, it is impossible to use the learned policies directly. Thus, we modify the state encoding at the Decision Making stage to raise the scalability of RoSE. The *modified state encodings* for robot k under different number cases of robots are defined as follows:

Algorithm 2: Decision Making

```
Input: $P_i, E_{i-1}, \theta, t_{max}, N_i^{\hat{k}}$
Output: $C_i^{\hat{k}}$
1  Initialize $O_i = 2.0$;
2  Initialize $E_{i,0}$ with Equation (32) and $P_{i,0} = P_i$;
3  for robot $k = 1, 2, \ldots, K$ do
4      if $k \in N_i^{\hat{k}}$ then
5          Initialize $A_0^k = 1$;
6      else
7          Initialize $A_0^k = 0$;
8  for step $t = 1, \ldots, t_{max}$ do
9      Initialize $C_t = \emptyset$;
10     for robot $k \in N_i^{\hat{k}}$ do
11         Get $S_t^k$ using the modified state encodings and select $A_t^k = \max_{A_t^k} Q(S_t^k, A_t^k)$ based on $\theta_k$;
12         if $A_t^k = 1$ then
13             Add robot $k$ into $C_t$;
14     Update residual energy of robots with Equation (31);
15     Compute the normalized objective $O_t = \omega f_{i,t}^1 + (1-\omega) f_{i,t}^{2,nor}$;
16     if $O_t < O_i$ then
17         Set $C_i^{\hat{k}} = C_t$ and $O_i = O_t$.
```

(1) Less Than $|\mathcal{K}|$: When the number of robots is less than $|\mathcal{K}|$, the information of all robots cannot construct the state S_t^k. According to the analysis of the state encoding in Section 5.3.2, π can be used to construct the state $S_{t,cp}^k$ without affecting the decision-making of the robot k. For the state $S_{t,ep}^k$, we set $\Delta E_{i,t}^{k,k'} = 0$ if robot k' doesn't exist.

(2) Equal To $|\mathcal{K}|$: Get S_t^k using Equation (17).

(3) More Than $|\mathcal{K}|$: Since a new robot $k' \notin \mathcal{K}$ participating in the network doesn't have policies about CB, we design the closet robot sharing its policies with robot k', and then we modify the state encoding. Because the robots for CB can affect the decision-making of robot k, the encoding of the $S_{t,cp}^k$ should prior consider those robots. Excluding robot k, if the number of robots for CB $|C_{t-1} - \{k\}|$ is less than $|\mathcal{K}|$, then robot k encodes S_t^k using Equation (17). On the contrary, if $|C_{t-1} - \{k\}| \geq |\mathcal{K}|$, we sort the phase difference between robot k and robots in $C_{t-1} - \{k\}$ incrementally, and the first $|\mathcal{K}|-1$ difference values are selected to encode the $S_{t,cp}^k$. This is because a smaller phase difference has a great impact on the MSLL according to the analysis of the state $S_{t,cp}^k$. Similarly, we sort the residual energy difference in descending order and select the first $|\mathcal{K}|-1$ difference values to encode $S_{t,ep}^k$.

5.5 Complexity Analysis

The complexity of the proposed RoSE algorithm includes two main components: Policy Acquisition and Decision Making. Considering that all agents execute training and decision-making processes simultaneously, we only analyze the complexity of an individual agent.

(1) Policy Acquisition Complexity:

State Encoding: At step t, encoding the CB part of agent k's state involves computing the minimum phase difference between k and other agents.

Since the phase α_i^k of agent k is a non-convex continuous function, we discretize $\phi \in [0, 2\pi)$ into segments of length $\Delta\phi$, leading to a state encoding complexity of $O(\Phi(|\mathcal{K}| - 1))$ for CB part, where $\Phi = \frac{2\pi}{\Delta\phi}$. The complexity of encoding the energy part is $O(|\mathcal{K}| - 1)$. Thus, the complexity of state encoding at each step t is $O((\Phi + 1)(|\mathcal{K}| - 1))$.

Reward Computation: At step t, the reward consists of two parts: $f_{i,t}^{1,k}$ and $f_{i,t}^{2,k}$. The computation of $f_{i,t}^{1,k}$ requires an exhaustive search over the beampattern with an incremental step of $\Delta\phi$ to find the MSLL, resulting in a complexity of $O\big(\Phi|\mathcal{K}|\big)$. The complexity of computing $f_{i,t}^{2,k}$ is $O\big(|\mathcal{K}|\big)$. Thus, the complexity of reward computation is $O\big((\Phi + 1)|\mathcal{K}|\big)$.

Q-network Training: The Q-network's structure includes two hidden fully-connected layers with h_1 and h_2 neurons, respectively. Considering the input dimension is $2(|\mathcal{K}| - 1)$ and the output dimension is $|A^k| = 2$. The complexity at step t is $O\big(2(|\mathcal{K}| - 1)h_1 + h_1h_2 + 2h_2\big)$.

Next, we denote the batch size and the number of episodes in the training process as B and E, respectively. The number of steps per episode is t_{max}, and then the complexity of the Policy Acquisition component is expressed as $B \times E \times t_{max} \times \big((\Phi + 1)(|\mathcal{K}| - 1) + (\Phi + 1)|\mathcal{K}| + 2(|\mathcal{K}| - 1)h_1 + (h_1 + 2)h_2\big)$.

(2) Decision Making Complexity: For the agent k, the complexity of making decisions based on trained policies is $I \times t_{max} \times \big((\Phi + 1)(|\mathcal{K}| - 1) + 2(|\mathcal{K}| - 1)h_1 + (h_1 + 2)h_2\big)$.

Combining the complexities of both components, the total complexity of the RoSE algorithm is $(BE + I)t_{max} \times \big((\Phi + 1)(|\mathcal{K}| - 1) + 2(|\mathcal{K}| - 1)h_1 + (h_1 + 2)h_2\big) + BEt_{max}((\Phi + 1)|\mathcal{K}|)$.

6 Performance Evaluation

In this section, we evaluate the performance of our algorithm and present simulation results. First, we describe the simulation setup. Then, we investigate some important parameter settings of DQN at the Policy Acquisition stage. We assess the performance of DQN through the **average reward** (**AR**) and the **reward range** (**RR**). AR is calculated by the mean value of episodic reward for all robots and RR is the range of episodic reward among robots. At last, we present benchmark schemes, followed by results and analysis. Particularly, to be suitable for a comprehensive evaluation, we adopt the normalized objective $\mathbb{F} = \frac{1}{I}\sum_{i \in I} \omega f_i^1 + (1 - \omega) f_i^{2,nor}$ as the system performance metric.

6.1 Simulation Setup

We consider a 100 m×100 m area that needs to be monitored, in which 10 robots with the same residual energy are randomly deployed. The position of the controller is $(1000, 0, 0)$ m. The number of time slots is $I = 100$. Within each time slot, robots move with different velocities $v_i^k \in [0, 1]$ m/s and the time of movement $\tau_i^{mov} \in [5, 10]$ s. Figure 6 shows the initial positions of robots and their trajectories within the first 10 time slots. For simplicity, we assume robots are within the communication range of each other, namely, $N_i^{\hat{k}} = \mathcal{K}, \forall i \in \mathcal{I}$. When extended to practical scenarios, the proposed algorithm and the following analysis still apply. To train robots, we have developed a simulator in Python and implemented DQN utilizing PyTorch. Related codes are made publicly available on Github (https://github.com/MonaHe123/RoSE). The simulations are conducted on a computer with a CORE i7 CPU @ 2.90 GHz. For clarity, more simulation parameters are shown in Table 2.

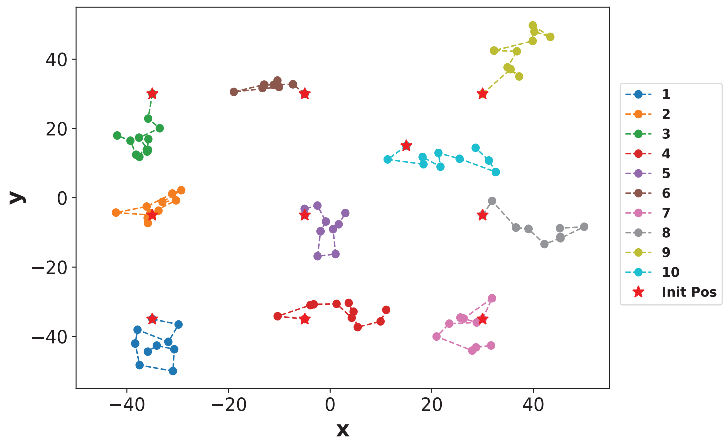

Fig. 6. Robot deployment in the monitored area and their trajectories of the first 10 time slots.

Table 2. Simulation Parameters

Parameter	Symbol	Value
Batch Size	B	128
Buffer Size	M	5,000
Discounter Factor	γ	0.95
Learning Rate	a	0.001
Transmitting Data Size	l_i	2 Mbits
Carrier Frequency	f_c	2.4 GHz [44]
Constant Motion Parameter	p_1	7.4 [35]
Constant Motion Parameter	p_2	0.29 [35]
Maximum Velocity	V_{max}	1 m/s [35]
Electronics Energy	e_{cct}	10^{-7} J/bit [36, 45]
Amplifier Energy Parameter	e_{tx}	10^{-12} J/bit/m^3 [36]
Path Loss Exponent	α	3 [36]
Initial Energy of Robots	E_0	3×10^5 J [46]
Percentage Coefficient	p	5%
Weight of the Objective f_i^1	w	0.5

6.2 Parameter Settings of RoSE

6.2.1 Learned Policy Sharing. We study the effect of learned policy sharing on DQN performance.

Effectiveness. In this simulation, we set the network update cycle as $U = 100$ and the reward function as DRF. Figure 7 shows the convergence and effectiveness of the DQN. For the Com, the agent updates θ_k^- according to the proposed algorithm RoSE. As a comparison, the agent updates $\theta_k^- = \theta_k$ following traditional DQN labeled as NoCom. It can be seen from Figure 7(a) that the learning curve for Com smooths out more rapidly. In addition, the Com has a smaller RR as shown in Figure 7(b). A smaller RR means more fairness between robots and a more stable network. A large RR may cause robots with high rewards to participate in CB frequently, resulting in rapid energy depletion. Thus, we draw the conclusion that learned policy sharing is effective as it can accelerate the convergence of the training and obtain a stable network.

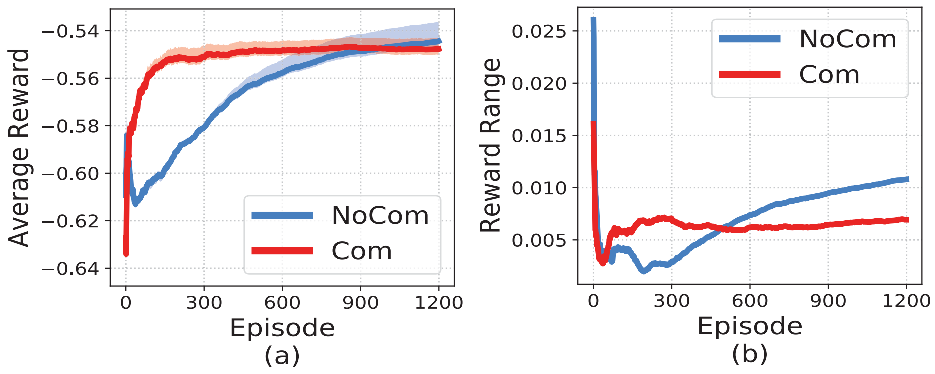

Fig. 7. Training convergence of the proposed algorithm with (Com) or without (NoCom) learned policy sharing. (a) Impact of the learned policy sharing on the AR. (b) Impact of the learned policy sharing on the RR.

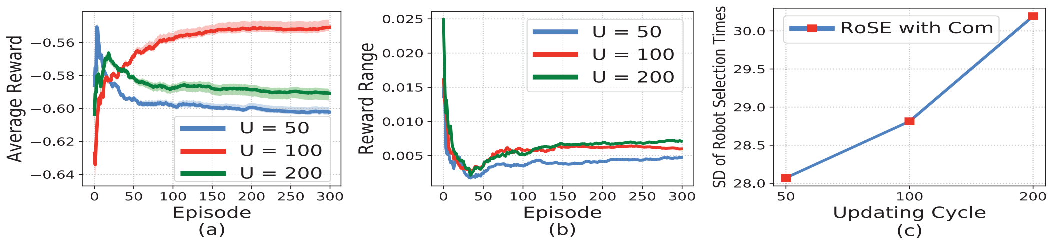

Fig. 8. Impact of the network updating cycle on the RoSE: (a) impact on the AR; (b) impact on the RR; (c) impact on the times of different robots participating in CB.

Network Updating Cycle. In Figure 8, we show the effect of different network updating cycles on the DQN performance for the proposed algorithm RoSE. As can be seen from Figure 8(a), RoSE has the highest reward and converges fastest when $U = 100$. This is because a short network updating cycle (e.g., $U = 50$) means frequent model parameter sharing, which may lead to unstable learning. Moreover, a long network updating cycle (e.g., $U = 200$) decreases the times of the policy sharing and degrades the training performance. However, it can be seen from Figure 8(b) that the RR grows as the network updating cycle increases. Figure 8(c) shows the standard deviation of the times of different robots participating in CB. The standard deviation increases as the network updating cycle grows, which verifies the results in Figure 8(b). For the purpose of taking into account both achievable reward and network stability, we set $U = 100$. If there is no special statement later, the values of U will not change.

6.2.2 *Step Number of Policy Execution.* Figure 9 shows the impact of different step numbers of policy execution t_{max} on the system performance. We can observe that as t_{max} increases from 50 to 200, the $\mathbb{F}$ decreases. When t_{max} exceeds 200, the system performance is hardly changed as t_{max} increases. For the proposed algorithm RoSE, each episode of policy execution has no clear finishing flag. Intuitively, the more steps robots execute the learned policies, the more likely it is to get the optimal solution, but it takes more time. On the contrary, if the step number is small, the optimal solution may not be found, resulting in poor performance. Thus, we set $t_{max} = 200$ in the subsequent simulations.

6.2.3 *Reward Function.* According to the used reward function, we label different learning schemes as DQN-DRF and DQN-ERF. In Figure 10, we compare the performance of DQN-DRF and

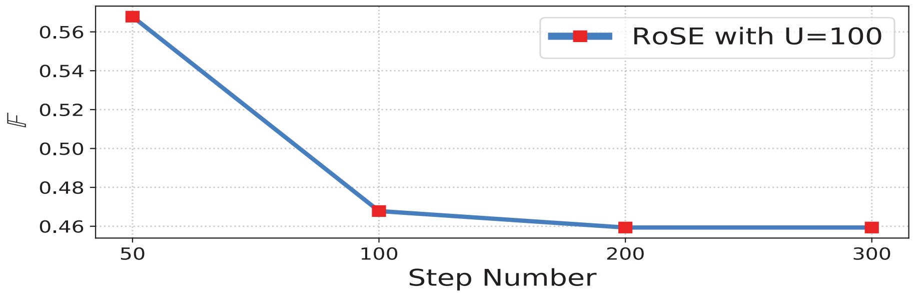

Fig. 9. Impact of different step numbers of policy execution on the $\mathbb{F}$.

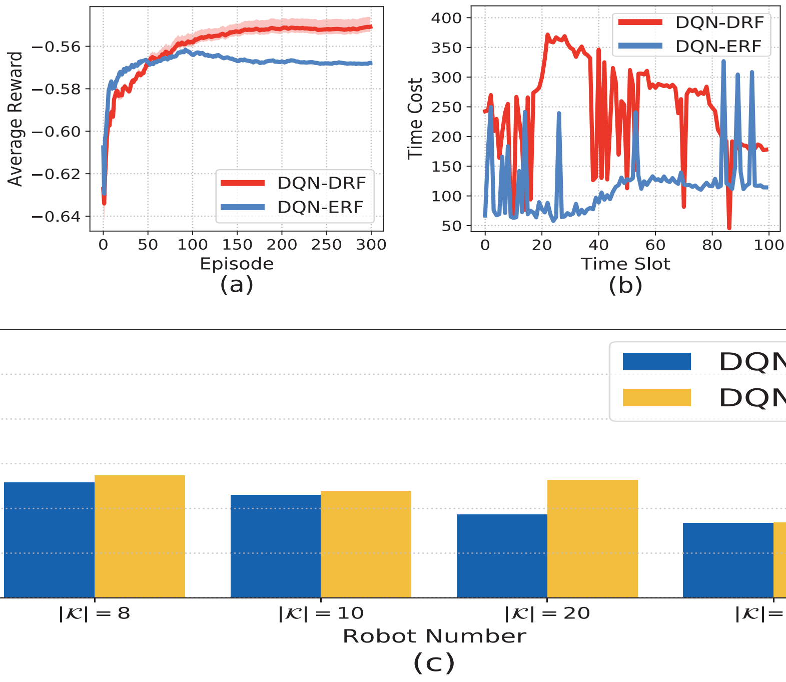

Fig. 10. Effectiveness of the proposed reward function DRF. (a) AR of different reward functions. (b) Time consumption of robot selection at each time slot for different reward functions. (c) $\mathbb{F}$ of reward functions under different numbers of robots.

DQN-ERF to verify the effectiveness of the proposed reward function DRF. As can be observed from Figure 10(a), the DQN-DRF can achieve a higher reward. However, DQN-DRF takes more time as shown in Figure 10(b). We apply the policies learned in the scenario with 10 robots to cases with varying robot numbers at the Decision Making stage. Their $\mathbb{F}$ performance is depicted in Figure 10(c). It is observed that DQN-DRF can obtain lower $\mathbb{F}$, indicating higher scalability to the number of robots. This is attributed to the characteristic of DQN-ERF that the robots in different states at each step get the same reward, whereas DQN-DRF individualizes rewards for robots. However, the scalability of both reward functions diminishes when the difference increases between the number of robots used at the Policy Acquisition stage and those used at the Decision Making stage. Consequently, their performance becomes similar, as illustrated in Figure 10(c) with

Table 3. Performance of Different Algorithms

Algorithm	Average MSLL	Average Normalized SDNED	$\mathbb{F}$
RoSE	0.6371	**0.2816**	**0.4593**
CHCSA	**0.6349**	0.2894	0.4622
SCNSA	0.7414	0.3235	0.5324
CCBO	0.7388	0.3961	0.5674
RNS	0.9072	0.2973	0.6023

$|\mathcal{K}| = 30$. The above observations demonstrate the effectiveness of the proposed reward function DRF in tackling the credit assignment problem.

6.3 Benchmark Schemes

To evaluate the performance of the proposed algorithm RoSE, we adopt the recent node selection algorithms mentioned in the related work as benchmarks.

- **RNS**: It is an ablation study of robot selection without optimization algorithms and selects robots for CB randomly.
- **SCNSA**: In this scheme, robot selection is performed by taking a VLA as the reference [16].
- **CCBO**: A concentric circular ring array is used to control the robot selection [17].
- **CHCSA**: It is a recent heuristics-based robot selection algorithm in which the cuckoo search algorithm is used to optimize CB performance [13].

It should be noted that the above benchmark schemes are for static scenarios. In the dynamic scenario of this article, benchmark schemes need to be executed from scratch to select robots at each time slot.

6.4 Performance Evaluation

6.4.1 System Performance. To prove the superiority of our proposed algorithm, we execute benchmark algorithms and the RoSE. The simulation results are shown in Table 3. As can be seen, even though CHCSA performs best in the average MSLL $\frac{1}{I}\sum_{i\in\mathcal{I}} f_i^1$, RoSE's average MSLL is just increased by 0.35% compared with CHCSA. What's more, RoSE reduces 14.07%, 13.77%, and 29.77% of the average MSLL compared with SCNSA, CCBO, and RNS, respectively. Considering the optimization objective f_i^2, we find that RoSE achieves the best performance. Specifically, RoSE can reduce the average normalized SDNED $\frac{1}{I}\sum_{i\in\mathcal{I}} f_i^{2,nor}$ by 2.70%, 12.95%, 28.91%, and 5.28% in comparison with CHCSA, SCNSA, CCBO, and RNS, respectively. Moreover, the $\mathbb{F}$ of RoSE is reduced by 0.63%, 13.73%, 19.05%, and 23.74% compared with CHCSA, SCNSA, CCBO, and RNS, respectively. It can be seen from the above results that RoSE can achieve a low MSLL and efficiently balance the network energy distribution.

6.4.2 Scalability to Robot Positions. In Figure 11, we show the time consumption of robot selection at each time slot for different algorithms. Compared with the proposed algorithm RoSE, SCNSA, CCBO, and RNS are observed to be more time-efficient. However, these benchmark algorithms show significant performance degradation in the MSLL, SDNED, and $\mathbb{F}$. Compared with CHCSA, RoSE significantly reduces the average time by 66.57% and has lower complexity. This is because when the robot's position changes, RoSE can use pre-trained policies without the need for retraining. For the CHCSA, once the robot's position changes, it needs to be executed from scratch. The above analysis indicates that the RoSE has a significant advantage in time complexity compared to the benchmark algorithm with the best performance, which also confirms the RoSE has good scalability to robot positions.

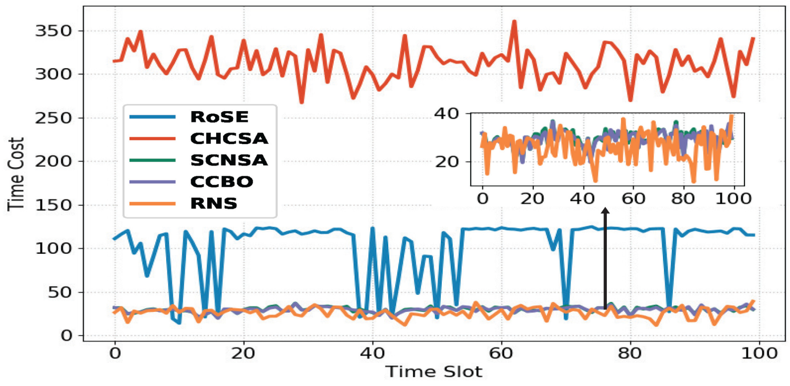

Fig. 11. The CPU time (*s*) of robot selection at each time slot for different algorithms.

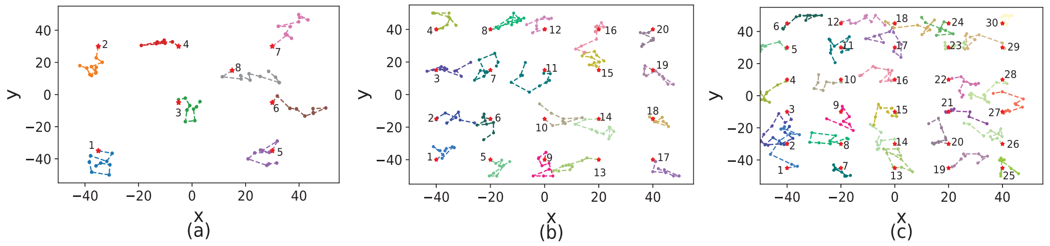

Fig. 12. Robot deployment in the monitored area and their trajectories of the first 10 time slots: (a) $|\mathcal{K}| = 8$; (b) $|\mathcal{K}| = 20$; (c) $|\mathcal{K}| = 30$.

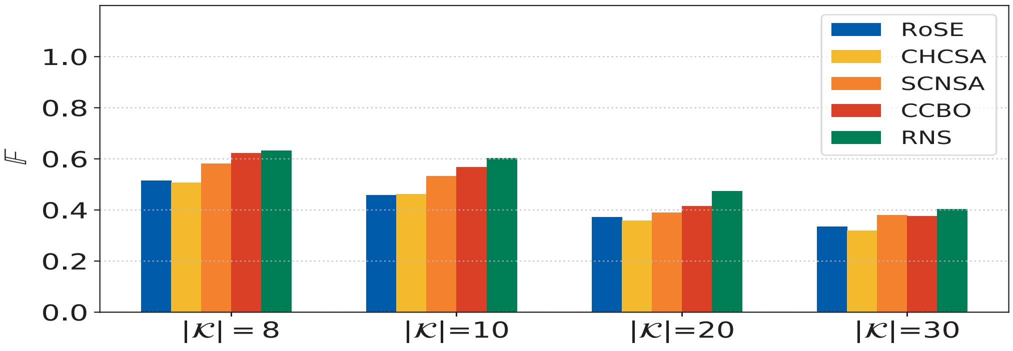

Fig. 13. $\mathbb{F}$ of different algorithms with different robot numbers.

6.4.3 Scalability to Robot Numbers. As shown in Figure 12, we use the learned policies based on 10 robots to explore the scalability of the RoSE to robot numbers under numbers of robots with 8, 20, and 30. Figure 13 shows the system performance $\mathbb{F}$ over different robot numbers. We can observe that the RoSE outperforms others when $|\mathcal{K}| = 10$. When $|\mathcal{K}| = 8$, $|\mathcal{K}| = 20$, and $|\mathcal{K}| = 30$, the $\mathbb{F}$ of CHCSA is slightly better than RoSE. But CHCSA takes longer time and is not suitable for practical scenarios. The above results show that when robot numbers change, RoSE can achieve system performance comparable to CHCSA without retraining. This also proves that RoSE has good scalability to robot numbers.

7 Conclusion

In this article, we formulate a multi-objective optimization problem intended to realize reliable and energy-efficient communications by robot selection in CB-based mobile robotic networks. To be competitive with existing algorithms in terms of CB performance, network lifetime and scalability, we propose a novel algorithm RoSE based on distributed MARL to solve the formulated problem. RoSE contains two stages: Policy Acquisition and Decision Making. At Policy Acquisition, a DQN with reward function DRF and learned policy sharing is designed for learning robot selection policies to optimize the MSLL and SDNED. At Decision Making, a simple but effective state encoding scheme is utilized to improve the scalability of the RoSE. Extensive simulation results have validated the superiority of the proposed algorithms in comparison with benchmark algorithms. Specifically, it can reduce the SDNED by 2.70% and the time by 66.57% compared with the CHCSA, which has the best system performance among benchmark algorithms. Meanwhile, RoSE has high scalability to robot positions and numbers. In the future work, we will consider the case where the distance between the MRVAA and the controller exceeds the maximum transmission range of CB, and explore multi-hop transmissions. Moreover, the endurance of robots is limited due to their inability to harvest energy during tasks. To address this issue, we will explore potential technologies such as simultaneous wireless information and power transfer [38, 39].

References

[1] Romulo Gonçalves Lins and Sidney N. Givigi. 2021. Cooperative robotics and machine learning for smart manufacturing: Platform design and trends within the context of industrial Internet of Things. *IEEE Access* 9 (2021), 95444–95455. DOI: 10.1109/ACCESS.2021.3094374

[2] Yu Guo, Zhenqiang Mi, Yang Yang, and Mohammad S. Obaidat. 2019. An energy sensitive computation offloading strategy in cloud robotic network based on GA. *IEEE Syst. J.* 13, 3 (2019), 3513–3523.

[3] You Wu Liu, Syazwina Binti Alias, Ming-yue Liu, and Bian-bian Jiao. 2022. Improved routing protocol based on multiobjective optimization in industrial robot networks. *Adv. Multim.* 2022 (2022), 7012779:1–7012779:8. DOI: 10.1155/2022/7012779

[4] Jinbo Xiong, Rong Ma, Lei Chen, Youliang Tian, Qi Li, Ximeng Liu, and Zhiqiang Yao. 2020. A personalized privacy protection framework for mobile crowdsensing in IIoT. *IEEE Trans. Ind. Informatics* 16, 6 (2020), 4231–4241.

[5] Jiafu Wan, Shenglong Tang, Qingsong Hua, Di Li, Chengliang Liu, and Jaime Lloret. 2018. Context-aware cloud robotics for material handling in cognitive industrial Internet of Things. *IEEE Internet Things J.* 5, 4 (2018), 2272–2281.

[6] Andrew Wichmann, Turgay Korkmaz, and Ali Saman Tosun. 2018. Robot control strategies for task allocation with connectivity constraints in wireless sensor and robot networks. *IEEE Trans. Mob. Comput.* 17, 6 (2018), 1429–1441.

[7] Wenbo Zhang, Xin Wang, Guangjie Han, Yan Peng, and Mohsen Guizani. 2021. SFPAG-R: A reliable routing algorithm based on sealed first-price auction games for industrial Internet of Things networks. *IEEE Trans. Veh. Technol.* 70, 5 (2021), 5016–5027.

[8] Isidro Calvo, Eneko Villar, Cristian Napole, Aitor Fernández, Oscar Barambones, and José Miguel Gil-García. 2021. Reliable control applications with wireless communication technologies: Application to robotic systems. *Sensors* 21, 21 (2021), 7107.

[9] Mustafa Ozger, Michal Vondra, and Cicek Cavdar. 2018. Towards beyond visual line of sight piloting of UAVs with ultra reliable low latency communication. In *IEEE Global Commun. Conf.* 1–6.

[10] Giuseppe Faraci, Christian Grasso, and Giovanni Schembra. 2020. Design of a 5G network slice extension with MEC UAVs managed with reinforcement learning. *IEEE J. Sel. Areas Commun.* 38, 10 (2020), 2356–2371.

[11] Yalcin Sadi and Sinem Coleri Ergen. 2015. Energy and delay constrained maximum adaptive schedule for wireless networked control systems. *IEEE Trans. Wirel. Commun.* 14, 7 (2015), 3738–3751.

[12] Yalcin Sadi and Sinem Coleri Ergen. 2017. Joint optimization of wireless network energy consumption and control system performance in wireless networked control systems. *IEEE Trans. Wirel. Commun.* 16, 4 (2017), 2235–2248.

[13] Geng Sun, Yanheng Liu, Zhaoyu Chen, Aimin Wang, Ying Zhang, Daxin Tian, and Victor CM Leung. 2021. Energy efficient collaborative beamforming for reducing sidelobe in wireless sensor networks. *IEEE Trans. Mob. Comput.* 20, 3 (2021), 965–982.

[14] Vahid Tarokh. 2009. *New Directions in Wireless Communications Research: Collaborative Beamforming*. Springer Publishing Company, Incorporated.

[15] Suhanya Jayaprakasam, Sharul Kamal Abdul Rahim, and Chee Yen Leow. 2017. Distributed and collaborative beamforming in wireless sensor networks: Classifications, trends, and research directions. *IEEE Commun. Surv. Tutorials* 19, 4 (2017), 2092–2116.

[16] Geng Sun, Yanheng Liu, Aimin Wang, Jing Zhang, Xu Zhou, and Zhao Liu. 2016. Sidelobe control by node selection algorithm based on virtual linear array for collaborative beamforming in WSNs. *Wirel. Pers. Commun.* 90, 3 (2016), 1443–1462.

[17] Shuang Liang, Zhiyi Fang, Geng Sun, Yanheng Liu, Xiaohui Zhao, Guannan Qu, Ying Zhang, and Victor C. M. Leung. 2019. JSSA: Joint sidelobe suppression approach for collaborative beamforming in wireless sensor networks. *IEEE Access* 7 (2019), 151803–151817. DOI: 10.1109/ACCESS.2019.2948091

[18] Xuecai Bao, Hao Liang, and Longzhe Han. 2018. A novel node selection algorithm for collaborative beamforming in wireless sensor networks. In *2018 IEEE Int. Conf. iThings., IEEE GreenCom., IEEE CPSCom., IEEE Smart Data.* 345–349.

[19] Xuecai Bao, Hao Liang, Yuan Liu, and Fenghui Zhang. 2019. A stochastic game approach for collaborative beamforming in SDN-based energy harvesting wireless sensor networks. *IEEE Internet Things J.* 6, 6 (2019), 9583–9595.

[20] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proc. 32th AAAI Conf. Artif. Intell.* 2974–2982.

[21] Geng Sun, Xiaohui Zhao, Guojun Shen, Yanheng Liu, Aimin Wang, Suhanya Jayaprakasam, Ying Zhang, and Victor CM Leung. 2020. Improving performance of distributed collaborative beamforming in mobile wireless sensor networks: A multiobjective optimization method. *IEEE Internet Things J.* 7, 8 (2020), 6787–6801.

[22] Jiahui Li, Hui Kang, Geng Sun, Shuang Liang, Yanheng Liu, and Ying Zhang. 2021. Physical layer secure communications based on collaborative beamforming for UAV networks: A multi-objective optimization approach. In *IEEE INFOCOM 2021 - IEEE Conf. Comput. Commun.* 1–10.

[23] Nik Noordini Nik Abd Malik, Mazlina Esa, and Sharifah Kamilah Syed Yusof. 2009. Optimization of adaptive linear sensor node array in wireless sensor network. In *2009 Asia Pacific Microw. Conf.* 2336–2339.

[24] Geng Sun, Yanheng Liu, Jing Zhang, Aimin Wang, and Xu Zhou. 2016. Node selection optimization for collaborative beamforming in wireless sensor networks. *Ad Hoc Networks* 37, P2 (2016), 389–403.

[25] Geng Sun, Xiaohui Zhao, Shuang Liang, Yanheng Liu, Ying Zhang, and Victor C. M. Leung. 2019. A hybrid optimization approach for suppressing sidelobe level and reducing transmission power in collaborative beamforming. In *IEEE 90th Veh. Technol. Conf.* 1–6.

[26] Jing Zhang, Li Lei, and Xin Feng. 2019. Energy-efficient collaborative transmission algorithm based on potential game theory for beamforming. *Int. J. Distributed Sens. Networks* 15, 9 (2019). DOI: 10.1177/1550147719877630

[27] N. N. Nik Abd Malik, Mazlina Esa, S. K. Syed Yusof, Shipun Anuar Hamzah, and Mohd Khairul Hisham Ismail. 2013. Circular collaborative beamforming for improved radiation beampattern in WSN. *Int. J. Distributed Sens. Networks* 9 (2013). DOI: 10.1155/2013/125423

[28] Wei Zhang, Haifen Yang, Zhen Yang, Letian Huang, and Zhiyong Guo. 2013. Collaborative beamforming for wireless sensor networks with sector-based node selection. In *Int. Conf. Commun., Circuits Syst.* 141–144.

[29] Mohammed F. A. Ahmed and Sergiy A. Vorobyov. 2010. Sidelobe control in collaborative beamforming via node selection. *IEEE Trans. Signal Process.* 58, 12 (2010), 6168–6180.

[30] Jung-Chieh Chen, Chao-Kai Wen, and Kai-Kit Wong. 2016. An efficient sensor-node selection algorithm for sidelobe control in collaborative beamforming. *IEEE Trans. Veh. Technol.* 65, 8 (2016), 5984–5994.

[31] Haitao Zhao, Jibo Wei, Shengchun Huang, Li Zhou, and Qi Tang. 2019. Regular topology formation based on artificial forces for distributed mobile robotic networks. *IEEE Trans. Mob. Comput.* 18, 10 (2019), 2415–2429.

[32] Haejoon Jung, In-Ho Lee, and Jingon Joung. 2022. Security energy efficiency analysis of analog collaborative beamforming with stochastic virtual antenna array of UAV swarm. *IEEE Trans. Veh. Technol.* 71, 8 (2022), 8381–8397.

[33] Xiaokun Fan, Min Liu, Yali Chen, Sheng Sun, Zhongcheng Li, and Xiaobing Guo. 2023. RIS-assisted UAV for fresh data collection in 3D urban environments: A deep reinforcement learning approach. *IEEE Trans. Veh. Technol.* 72, 1 (2023), 632–647.

[34] Hideki Ochiai, Patrick Mitran, H. Vincent Poor, and Vahid Tarokh. 2005. Collaborative beamforming for distributed wireless ad hoc sensor networks. *IEEE Trans. Signal Process.* 53, 11 (2005), 4110–4124.

[35] Yunlong Wu, Bo Zhang, Shaoshi Yang, Xiaodong Yi, and Xuejun Yang. 2017. Energy-efficient joint communication-motion planning for relay-assisted wireless robot surveillance. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications.* 1–9.

[36] Chia-Ching Ooi and Christian Schindelhauer. 2009. Minimal energy path planning for wireless robots. *Mob. Networks Appl.* 14, 3 (2009), 309–321.

[37] Praveen Kumar Donta, Satish Narayana Srirama, Tarachand Amgoth, and Chandra Sekhara Rao Annavarapu. 2023. iCoCoA: Intelligent congestion control algorithm for CoAP using deep reinforcement learning. *J. Ambient Intell. Humaniz. Comput.* 14, 3 (2023), 2951–2966.

[38] JainShing Liu, Chun-Hung Richard Lin, Yu-Chen Hu, and Praveen Kumar Donta. 2022. Joint beamforming, power allocation, and splitting control for SWIPT-Enabled IoT networks with deep reinforcement learning and game theory. *Sensors* 22, 6 (2022), 2328.

[39] Jain-Shing Liu, Chun-Hung Richard Lin, Yu-Chen Hu, and Praveen Kumar Donta. 2023. Joint data transmission and energy harvesting for MISO downlink transmission coordination in wireless IoT networks. *Sensors* 23, 8 (2023), 3900.

[40] Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. 2018. Credit assignment for collective multiagent RL with global rewards. In *Advances Neural Inf. Process. Syst.*. Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.), 8113–8124.

[41] Woojun Kim, Myungsik Cho, and Youngchul Sung. 2019. Message-Dropout: An efficient training method for multi-agent deep reinforcement learning. In *33th AAAI Conf. Artif. Intell.* 6079–6086.

[42] Jiechuan Jiang and Zongqing Lu. 2018. Learning attentional communication for multi-agent cooperation. In *Proc. 32th Int. Conf. Neural Inf. Process. Syst.* 7265–7275.

[43] 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. *Nat.* 518, 7540 (2015), 529–533.

[44] Pouria Zand, Supriyo Chatterjea, Kallol Das, and Paul J. M. Havinga. 2012. Wireless industrial monitoring and control networks: The journey so far and the road ahead. *J. Sens. Actuator Networks* 1, 2 (2012), 123–152.

[45] Sanjai Prasada Rao Banoth, Praveen Kumar Donta, and Tarachand Amgoth. 2023. Target-aware distributed coverage and connectivity algorithm for wireless sensor networks. *Wirel. Networks* 29, 4 (2023), 1815–1830.

[46] Yongguo Mei, Yung-Hsiang Lu, Yu Charlie Hu, and C. S. George Lee. 2006. Deployment of mobile robots with energy and timing constraints. *IEEE Trans. Robotics* 22, 3 (2006), 507–522.

Received 16 January 2024; revised 24 June 2024; accepted 2 July 2024