Cross-Entropy Regularized Policy Gradient for Multirobot Nonadversarial Moving Target Search

Hongliang Guo, Zhaokai Liu, Rui Shi, Wei-Yun Yau, and Daniela Rus, *Fellow, IEEE*

***Abstract*—This article investigates the multirobot efficient search (MuRES) for a nonadversarial moving target problem from the multiagent reinforcement learning (MARL) perspective. MARL is deemed as a promising research field for cooperative multiagent applications. However, one of the main bottlenecks of applying MARL to the MuRES problem is the nonstationarity introduced by multiple learning agents. With learning agents simultaneously updating their policies, the environment cannot be modeled as a *stationary* Markov decision process, which results in the inapplicability of fundamental reinforcement learning techniques such as deep *Q*-network and policy gradient (PG). In view of that, we adopt the centralized training and decentralized execution scheme and thereby propose a cross-entropy regularized policy gradient (CE-PG) method to train the learning agents/robots. We let the robots *commit* to a predetermined policy during execution, collect the trajectories, and then perform centralized training for the corresponding policy improvement. In this way, the nonstationarity problem is overcome, in that the robots do not update their policies during execution. During the centralized training stage, we improve the canonical PG method to consider the interactions among robots by adding a cross-entropy regularization term, which essentially functions to "disperse" the robots in the environment. Extensive simulation results and comparisons with state of the art show CE-PG's superior performance, and we also validate the algorithm with a real multirobot system in an indoor moving target search scenario.**

***Index Terms*—Centralized training and decentralized execution (CTDE), cross-entropy regularized policy gradient (CE-PG), multiagent reinforcement learning (MARL), multirobot efficient search (MuRES), nonadversarial moving target search.**

Manuscript received 11 December 2022; accepted 24 March 2023. Date of publication 14 April 2023; date of current version 8 August 2023. This paper was recommended for publication by Associate Editor P. Tokekar and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. This work was supported in part by the Robotics Horizontal Technology Coordinating Office, Agency for Science, Technology and Research, Singapore, under Grant C221518004. This paper was recommended for publication by Associate Editor Pratap Tokekar and Editor Paolo Robuffo Giordano upon evaluation of the reviewers' comments. *(Corresponding author: Hongliang Guo.)*

Hongliang Guo and Wei-Yun Yau are with the Institute for Infocomm Research, Agency for Science, Technology and Research,, Singapore 138632 (e-mail: guo_hongliang@i2r.a-star.edu.sg; wyyau@i2r.a-star.edu.sg).

Zhaokai Liu and Rui Shi are with the School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: zk_allen@outlook.com; shiruiaa@foxmail.com).

Daniela Rus is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: rus@csail.mit.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2023.3263459.

Digital Object Identifier 10.1109/TRO.2023.3263459

I. Introduction

Multirobot efficient search (MuRES) for a nonadversarial moving target has been a hot research topic, attracting increasing attention from both academic researchers and industrial entrepreneurs over the past several decades. Here, a "nonadversarial" moving target refers to the type of target whose movement dynamics are independent of, and thus do not react to, the searchers' movement strategies. On the one hand, the MuRES problem has many real-world application potentials, such as multirobot search and rescue in hazardous environments [1], [2], [3], [4], [5], collaborative source leakage localization [6], [7], and multirobot security defense and surveillance [8], [9]. On the other hand, MuRES also serves as a representative operation research topic and lies in the intersection of many fundamental research areas, such as multiagent learning [4], [10], [11], game theory [12], swarm dynamics [13], [14], [15], cooperative control [16], [17], and graph theory [18], [19].

Researchers have proposed various algorithms to solve the MuRES problem, and a brief literature review of MuRES will be provided in Section II. Here, we wish to articulate that the prevailing MuRES solutions are *planning methods*, which formulate the MuRES problem into a monolithic mathematical programming paradigm and then employ off-the-shelf optimization solvers, e.g., CPlex [20] and branch and bound [21], or take advantage of the special nature of the problem for distributed solutions [22]. However, to the best of our knowledge, almost all the planning methods for the MuRES problem require, as inputs, the a priori information of the target's motion dynamics and its initial position distribution, both of which are not always available in many real-world applications. On the other hand, *learning methods* are inherently model free, which do not need the pregauged target motion dynamics as inputs.

Therefore, in this article, we turn our attention to the field of multiagent reinforcement learning (MARL) and treat the MuRES problem from the perspective of the decentralized partially observable Markov decision process (Dec-POMDP) framework. MARL has been deemed as a promising field for cooperative multiagent applications. However, the nonstationarity caused by multiple learning agents during the search process prevents its direct application to the MuRES problem. Moreover, in MuRES, the neighbors of a learning robot are dynamically changing during the task execution process, and the total number of robots is not necessarily known to each robot and might even be subject to change during task execution, e.g., some robots might malfunction and quit the team, or new robots are added to the team for reinforcement. Those features of the MuRES problem prohibit us from directly applying canonical MARL methods as the straightforward MuRES solution. In view of the aforementioned challenges, i.e., nonstationarity, dynamic changing neighbors, and unknown and nonstationary total number of robots, we design a cross-entropy regularized policy gradient (CE-PG) method as the MuRES solution. CE-PG adopts the centralized training and decentralized execution (CTDE) scheme, which trains the learning agents[1] in a centralized manner and lets them execute the pretrained policy in a fully decentralized way. Through the CTDE scheme, the nonstationarity problem is resolved, in that the agents, i.e., robots in the MuRES context, do not change their policies during execution, which ensures the stationary Markov decision process. Moreover, during the execution phase, each individual robot uses the online Bayesian computation method to recursively estimate the probabilistic distribution of target's position as its decision-making basis, which does not need the communication or coordination with other robots, and thus, CE-PG avoids designing the complex robot–robot interaction mechanism. Furthermore, we improve the vanilla policy gradient (PG) method for the moving target search problem to include a cross-entropy regularization term, which functions to prevent the multiple robots from conglomeration. The cross-entropy term is calculated between the ego robot's policy and the *average* policy from all the other robots in the system. The average policy is much stabler than an individual policy and, hence, is robust against individual robot failures. Therefore, CE-PG has the unique feature of behaving well in face of individual robot failures, which is also verified in the simulation section.

The contributions of CE-PG can be summarized as follows: 1) CE-PG adopts the CTDE scheme, which resolves the nonstationarity problem during multiagent learning; 2) the execution process of each CE-PG agent is independent from its neighbors, which avoids designing the complex robot–robot interaction mechanism; and 3) the cross-entropy regularization term ensures that the robots are dispersed in the environment, and in the meanwhile, the calculation process of cross entropy between the ego robot's policy and the average policy from all the other robots makes CE-PG robust against individual robot failures. We perform simulations in a range of canonical MuRES test environments and also deploy CE-PG to a real multirobot system for nonadversarial moving target search in a self-constructed indoor environment with satisfying results.

The rest of this article is organized as follows. Section II presents a brief literature review of MuRES along the taxonomies of its objective, environment type, target's behavior, sensor type, and methodology, followed by the MuRES problem formulation and background introduction of the CTDE scheme and the vanilla PG method in Section III. The CE-PG framework, its pseudocode, and computational complexity analysis are introduced in Section IV. We present the simulation results, comparisons, and analysis in Section V, followed by showcasing the deployment of CE-PG to a real multirobot system in Section VI. Finally, Section VII concludes this article. We deliver the proofs of related theorems in the Appendixes.

II. Literature Review

Broadly speaking, the domain of multirobot target search can be divided into two subareas: multirobot guaranteed search (MuRGS) and MuRES. MuRGS aims at coordinating a group of robots in such a way that the target *cannot* escape being detected, regardless of its motion characteristics and/or sensing capabilities [23]. On the other hand, MuRES targets the problem of designing *efficient* multirobot search strategies so that the overall search effort, e.g., the target's expected capture time, is minimized. Since this article tackles the MuRES problem, in this section, we focus on reviewing MuRES-related research along the taxonomies of 1) MuRES objectives; 2) environment types; 3) target's motion behaviors; 4) robot's sensor characteristics; and 5) prevailing MuRES methodologies. For MuRGS-related research, one may refer to [24], [25], [26], [27], [28], and [29]. Fig. 1 presents a bird's-eye view of the MuRES-related research.

A. Taxonomies of the MuRES Problem

This subsection describes the MuRES problem from different perspectives.

1) *Objectives:* There are two mainstream objectives in the MuRES literature, namely, MuRES Problem I, which aims at minimizing the target's expected capture time (min. CT) [22], [30], [31], and MuRES Problem II, whose objective is to maximize the target's probability of detection (max. PD) within a given time budget [20], [21], [32], [33].
2) *Environments:* One may split MuRES environments into *discrete* environments, where the environment is represented by topological graphs [20], [22], [30], [34] or partitioned into Cartesian grids [32], and *continuous* environments [32], [33].
3) *Target's motion dynamics:* The target to be searched for can be dichotomized into the stationary target [32], where the target does not move during the search process, and the moving target [20], [21], [22], [30], [31]. For the moving target, one may further divide it into the nonadversarial moving target [20], [21], [22], whose motion dynamics does not change with respect to the searchers' strategy, and the adversarial moving target[2] [30], [31], who changes its moving pattern based on the observation of the searchers' positions and actions.
4) *Sensor characteristics:* Different types of environments endow different descriptions of the robots' sensor range descriptions. For continuous environments, the sensor's detection range can be circular [26], which detects the target within a certain distance from the sensor, or line of sight [31], which detects the target as long as there is an unblocked straight line connecting the sensor with the target. For discrete environments, the sensor's detection range

[1]Note that in the MuRES domain, "agent" refers to the searching robot, and we use the term "agent" and "robot" interchangeably in the MuRES context.

[2]Note that the adversarial moving target search problem is also called the pursuit–evasion game in the literature, which is not in the domain of this article.

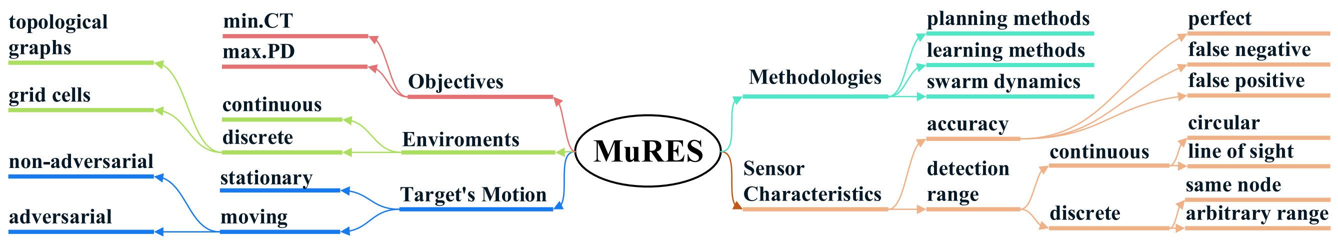


Fig. 1. Bird's-eye view of the MuRES literature.

can be defined as same-node detection [22] and arbitrary range detection [20]. Another dimension of characterizing the sensor characteristics is whether the sensor is *perfect* or *probabilistic*. The perfect sensor always returns the true information of the target, i.e., whether there is a target within the detection range [32], while the probabilistic sensor has a certain false negative detection probability [20], i.e., fail to detect a target even if it is within the sensor detection range, and/or even false positive detection probability [17], i.e., mistakenly deem another object within the sensor range as the target.

B. MuRES Methodologies

Researchers have designed various MuRES methodologies, and in this article, we partition them into three groups: planning methods, learning methods, and swarm dynamics.

Planning methods are deemed as the most canonical MuRES solutions. Within this subgroup, the MuRES problem is usually treated as a mathematical optimization problem. With the models of target's motion dynamics, initial location distributions, and the robots' motion characteristics, researchers establish a set of mathematical equations describing the MuRES objective and related constraints. Then, off-the-shelf solvers, e.g., CPlex and branch and bound, are invoked for the exact solution [20], [21], [32]. To expedite the solution process, the preestablished mathematical optimization problem is decomposed into several small but easy-to-solve subproblems, and distributed solutions are proposed for efficient approximate solutions [20], [22]. For example, Asfora et al. [20] establish the MuRES problem as a mixed-integer linear programming (MILP) problem and use CPlex to solve the problem. In the meanwhile, they make the solution distributed by sequentially allocating the robot's decision sequence to each subsequent robot. In this way, the later robots are able to incorporate the former robots' decisions for better cooperative search strategies. The distributed solution is shown to have the similar level of performance with much less computation time.

Learning methods are recently emerging methods for the MuRES problem. Researchers within this category treat MuRES as a multirobot sequential decision-making problem and usually establish it within the framework of Dec-POMDP. After that, various (decentralized) policy optimization algorithms, such as deep deterministic policy gradient [17], deep Q-network (DQN) [35], and Monte Carlo tree search [30], are proposed. For example, Qin et al. [35] design a DQN method for MuRES within the four-connected grid world for stationary targets. To overcome the sparse reward problem, the authors additionally incorporate the environmental uncertainty reduction as the auxiliary reward for decision making. Multiagent learning for the MuRES problem is a promising direction; however, as we have stated, nonstationarity during the robots' simultaneous learning process and dynamic neighborhood information make it difficult to directly transplant the prevailing MARL methods into the MuRES domain.

The third group of methods for the MuRES problem is swarm dynamics. Researchers in this group design various agent-based interaction mechanisms as the behavior guideline for each robot to follow. While each robot is executing its own dynamics, the robot swarm, as a team, is exhibiting a certain group-level behavior for efficient target search [7], [13], [15], [36], [37], [38]. For example, Tang et al. [36] revise the grey wolf optimization method to dynamically decide the next-goal point for each individual robot; with the next goal information, each robot also takes its observed obstacle information and momentum into consideration and reaches the finally merged dynamics. Designing swarm dynamics for the MuRES problem is easy to implement, and the team behavior is naturally robust to individual robot failures or new team member additions. However, to the best of our knowledge, it is very difficult, if not impossible, to establish a clear relationship between the MuRES objective and the robot–robot interaction mechanism. It means that one has to try different types of robot–robot interaction mechanisms and "hope" that one of the emerged team behaviors from the individual interactions fits the MuRES objective.

In this article, we propose CE-PG as a new multiagent learning method for the MuRES problem. Different from state-of-the-art learning methods, we adopt the CTDE scheme, which defers the learning process to the centralized training (CT) stage and lets the robots commit to the precalculated policies during the execution stage. In this way, we overcome the nonstationary problem. Furthermore, we improve the vanilla PG method to incorporate a cross-entropy regularization term, which functions to disperse the robots in the field. In this way, we do not need to design the complex robot–robot interaction mechanism during the execution stage; instead, each robot just follows its precalculated policy, and the robots are dispersed automatically.

III. Problem Formulation and Background

In this section, we first lay down the MuRES problem formulation and then introduce the basics of CTDE and PG, both of

TABLE I
LIST OF MAJOR NOTATIONS USED IN THIS ARTICLE

Notations	Descriptions
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	the 'unit-cost' graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$
$\mathcal{V}$	set of nodes, $\lvert\mathcal{V}\rvert = n$
$\mathcal{E}$	set of edges, $\lvert\mathcal{E}\rvert = m$
N	the total number of robots
e_t	the target's position at time t, $e_t \in \mathcal{V}$
Γ	the target's motion dynamics
$\hat{\Gamma}$	the estimated target's motion dynamics
$\boldsymbol{b}_0$	the target's initial position distribution
$p_t^{(i)}$	robot i position at time t, $p_t^{(i)} \in \mathcal{V}$
$a_t^{(i)}$	robot i's action at time t
$s_t^{(i)}$	robot i's state at time t
$r_t^{(i)}$	robot i's instant reward at time t, $r_t^{(i)} = -1$
$z_t^{(i)}$	robot i's observation at time t, $z_t^{(i)} \in \{0, 1\}$
$p_{\leq t}^{(i)}$, $z_{\leq t}^{(i)}$	robot i's position/observation sequence up to time t
T	length of the position/observation sequence
$\pi^{(i)}(p_{\leq t}^{(i)}, z_{\leq t}^{(i)})$	robot i's decision making policy at time t
η_{fp}, η_{fn}	the sensor's false positive/negative ratio
$\boldsymbol{b}_t^{(i)}$	robot i's probabilistic target belief at time t
$\boldsymbol{b}_t$	collective probabilistic target belief at time t
$J(\pi_{\boldsymbol{\theta}})$	the expected return of policy $\pi_{\boldsymbol{\theta}}$
$\mathcal{H}(\pi(\boldsymbol{\theta}_j), \pi(\boldsymbol{\theta}_i))$	the cross entropy term from robot j to robot i
β_i	robot i's balance parameter between J and $\mathcal{H}$
α_i	robot i's learning rate
$t_{\text{cap}}^{(i)}$	the target's capture time by robot i
t_{cap}	the target's capture time, $t_{\text{cap}} = \min_i\{t_{\text{cap}}^{(i)}\}$

which serve as the background knowledge of CE-PG. Table I presents a list of major notations used throughout this article. Note that we will also state the related symbol's definition when it is introduced in the main contents for the first time.

A. MuRES Problem Formulation

The MuRES problem that we are investigating is to deploy a team of N robots, also named as searchers, in a discrete environment to search for one nonadversarial moving target with the minimal expected time. The term "nonadversarial" means that the target is moving according to its own motion dynamics and does not react to the searchers' positions or search strategies. In the following, we will provide descriptions of 1) the environment; 2) the target (position and motion); 3) the robots (position, action, observation, reward, and policy); and 4) the capture event.

1) The *environment* is represented by an undirected and connected "unit-cost" graph, $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ ($\lvert\mathcal{V}\rvert = n$) refers to the set of nodes and $\mathcal{E}$ ($\lvert\mathcal{E}\rvert = m$) refers to the set of edges. The word "unit-cost" means that each robot's action, i.e., executing an edge or staying in the same node, has a time cost at 1. Note that the "unit-cost" assumption is a common one in the MuRES literature for the discrete environments; see [17], [20], [22], and [32] for examples.
2) The *target*'s position at time t is denoted as e_t. Note that this article presumes that both the robots and the target can only reside in nodes, i.e., $e_t \in \mathcal{V}$. The target moves according to its own motion dynamics, represented by a stochastic matrix $\boldsymbol{\Gamma}$, which stochastically transits the target from its current position to one of its neighboring according to $\mathcal{G}$ or makes the target stay at the same node, i.e., $\mathbb{P}[e_{t+1}|e_t] = \boldsymbol{\Gamma}(e_t, e_{t+1})$.
3) The *robot*'s position at time t is denoted as $p_t^{(i)}$, where $i \in \{1, 2, \ldots, N\}$ is the index of the robot. The robot's action, denoted as $a_t^{(i)}$, can be to execute any edge connected to $p_t^{(i)}$ at time t or to select to stay at the same node, which results in $p_{t+1}^{(i)} = p_t^{(i)}$. The instant reward of robot i after taking $a_t^{(i)}$, denoted as $r_t^{(i)}$, is equal to the negative of the action's cost, i.e., $r_t^{(i)} = -1$. The robot's observation at time t is a binary variable $z_t^{(i)} \in \{0, 1\}$, with $z_t^{(i)} = 0$, meaning that the robot "believes" that the target is *not* in node $p_t^{(i)}$ at time t, and with $z_t^{(i)} = 1$ meaning that the robot "thinks" that the target is in node $p_t^{(i)}$ at time t. Note that in this article, we consider the probabilistic sensors with both false negative and false positive detection probabilities. Moreover, since we assume that there is no communication or explicit coordination among robots during execution, the robot's decision-making policy at time t, which is denoted as $\pi^{(i)}$, should depend only on the robot's own history of positions and observations, i.e., $p_{\leq t}^{(i)}$ and $z_{\leq t}^{(i)}$. Here, $p_{\leq t}^{(i)}$ refers to the robot's position sequence from time 0 (initial position) to time t, and $z_{\leq t}^{(i)}$ refers to the robot's observation sequence from time 0 to time t, i.e., $z_0^{(i)}, z_1^{(i)}, \ldots, z_t^{(i)}$. Therefore, robot i's decision-making policy $\pi^{(i)}$ is a function of $p_{\leq t}^{(i)}$ and $z_{\leq t}^{(i)}$, i.e., $\pi^{(i)}(p_{\leq t}^{(i)}, z_{\leq t}^{(i)})$.
4) The target is *captured*[3] if $\exists i$, such that $p_t^{(i)} = e_t$ and $z_t^{(i)} = 1$. It means that robot i and the target reside in the same node at time t, and in the meanwhile, robot i detects the target ($z_t^{(i)} = 1$). We denote the target's capture time as t_{cap} and use $t_{\text{cap}}^{(i)}$ to indicate the capture time by robot i, apparently, $t_{\text{cap}} = \min_i\{t_{\text{cap}}^{(i)}\}$. The MuRES problem is then defined as finding the optimal joint policy $\pi = \{\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(N)}\}$, which minimizes the expected capture time, i.e., $\mathbb{E}[t_{\text{cap}}]$.

B. Centralized Training and Decentralized Execution

In MARL, there are many decision-making agents, which simultaneously learn and interact with the environment. On the one extreme, one may be tempted to train each agent completely independently by treating other agents' behaviors as part of the environment, e.g., independent Q-learning (IQL) [39]. On the other extreme, one may treat all the agents as one monolithic global agent and uses the centralized reinforcement learning (RL) methods to learn the joint optimal policy. However, both extremes suffer from severe drawbacks. On the one hand, the simultaneous learning agents make the environmental transitions nonstationary, and methods like IQL cannot even guarantee

[3]It is worth mentioning that in this article, we do not differentiate the meanings of "detected" and "captured" in the MuRES problem.

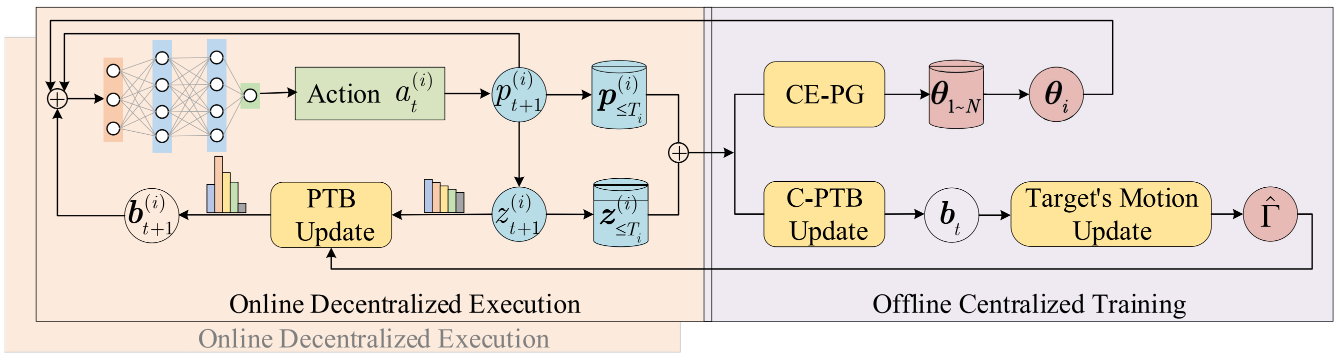

Action $a_t^{(i)}$
$p_{t+1}^{(i)}$
$\boldsymbol{p}_{\le T_i}^{(i)}$
$z_{t+1}^{(i)}$
$\boldsymbol{z}_{\le T_i}^{(i)}$
$\boldsymbol{b}_{t+1}^{(i)}$
PTB Update
CE-PG
$\boldsymbol{\theta}_{1\sim N}$
$\boldsymbol{\theta}_i$
C-PTB Update
$\boldsymbol{b}_t$
Target's Motion Update
$\hat{\Gamma}$
Online Decentralized Execution
Offline Centralized Training
Online Decentralized Execution

Fig. 2. CE-PG's framework: CE-PG follows the CTDE scheme, which consists of the DE module and CT module. The DE module is deployed to each robot, collects online information, e.g., robot's position ($p_t^{(i)}$) and observation ($z_t^{(i)}$), updates PTB ($b_t^{(i)}$) online, and chooses actions according to pre-trained policy network ($\pi(\boldsymbol{\theta}_i)$). The CT module runs on the centralized server, collects all the robots' position and observation sequences, updates the centralized target's motion dynamics, and trains all the robots' policy network with the cross-entropy regularization in an offline centralized manner. The symbol "$\oplus$" indicates the concatenation operation.

its ultimate convergence. On the other hand, the computational process of training a global joint policy with centralized methods is ultracomplex, and in MuRES, the total number of agents is also subject to change, which prohibits us from applying the centralized RL methods.

On the other hand, the CTDE scheme [40] lies between these two extremes. CTDE performs the policy training process in a *centralized* way, where each agent is able to access the global state as well as all the other agents' action–observation history. However, during the execution stage, each agent only has access to its own position–observation history and makes local decisions. The benefits of CTDE are twofold: first, during the training stage, the learning agent is able to make full use of the centralized information and has the potential to learn the optimal policy; second, during the execution stage, each agent is functioning in a fully decentralized manner by committing itself to the pretrained policy. Committing to a precalculated policy also makes the environment behave in a stationary way, which provides the theoretical foundation of the related RL algorithm's ultimate convergence. In summary, the essence of CTDE is to train an *online decentralized* policy for each agent in an *offline centralized* way.

C. Policy Gradient

PG methods aim at maximizing the expected return based on the PG theorem [41], by directly computing an estimate of the gradient of policy parameters. With the help of (deep) neural networks, the PG algorithms have become the prevalent RL methods. Defining $J(\pi_{\boldsymbol{\theta}})$ as the expected return, the gradient of $J(\pi_{\boldsymbol{\theta}})$ is calculated as $\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = \mathbb{E}[\sum_{t=0}^{T} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log(\pi(a_t|s_t)) G^{\pi_{\boldsymbol{\theta}}}(s_t, a_t)]$, where $G^{\pi_{\boldsymbol{\theta}}}(s_t, a_t)$ is the estimated accumulated reward from (s_t, a_t) and T is the length of the episode. One may refer to [42, p. 325] for the derivation details of the PG theorem. In this article, we make use of the vanilla PG to let the robot learn to search for the moving target and, in the meanwhile, add a cross-entropy regularization term to foster cooperation through dispersing the robots in the environment.

IV. Cross-Entropy Regularized Policy Gradient

This section presents the CE-PG algorithm for the MuRES problem. As stated previously, CE-PG follows the CTDE scheme, which consists of two modules, namely, the *offline* CT module and the *online decentralized execution* (DE) module. The CT module collects all the agents' trajectories, i.e., $p_{\le t}^{(i)}$ and observations, i.e., $z_{\le t}^{(i)}$, and performs the offline CT procedure, which recalculates 1) the estimated target's motion dynamics, i.e., $\hat{\boldsymbol{\Gamma}}$; 2) the target's initial position distribution, i.e., $\boldsymbol{b}_0$; and 3) the updated parameterized policy for each agent, i.e., $\pi_{\boldsymbol{\theta}_i}^{(i)}$. On the other hand, the DE module is deployed to each agent, which 1) gets the agent's real-time position, i.e., $p_t^{(i)}$ and observation i.e., $z_t^{(i)}$; 2) estimates the agent-specific probabilistic target belief (PTB), denoted as $\boldsymbol{b}_t^{(i)}$, online; and 3) makes the real-time decision based on the pretrained policy, i.e., $\pi_{\boldsymbol{\theta}_i}^{(i)}$. Fig. 2 presents the overall framework of CE-PG. In the following subsections, we will first introduce CE-PG's decentralized execution module, which consists of the online PTB update and policy execution and then, with two subsections, introduce the CT module, which corrects the estimates of $\hat{\boldsymbol{\Gamma}}$ and $\boldsymbol{b}_0$ and thereby trains each agent's policy with interagent cross-entropy regularization. Thereafter, we present the computational complexity analysis of CE-PG's DE module and skip the corresponding computational complexity analysis of the CT module, in that CE-PG's DE module determines the online decision-making time of the deployed algorithm, while the CT module can be executed in an offline manner. We extend CE-PG's application to robots with a broad field of view, e.g., aerial robots, and discuss the use case of an "all-to-all" communication scheme toward the end of this section.

A. Online Decentralized Execution

The online decentralized execution module of CE-PG is deployed to each agent and will 1) update the agent's PTB ($\boldsymbol{b}_t^{(i)}$) with the newly collected information, i.e., $p_t^{(i)}$ and $z_t^{(i)}$; and 2) execute the parameterized policy based on the robot's current state, i.e., $s_t^{(i)}$. Before proceeding to the contents, we need to present the formal definition of PTB, which constitutes the robot's state and serves as the basis for policy parameterization.

Definition 1 (Probabilistic target belief): An agent's PTB is the agent's estimated probabilistic distribution of the target's position.

Note that different agents may have different PTBs, as they have different "experiences" while interacting with the environment, and even for the same agent, its PTB will change as it interacts with the environment and collects more information. The PTB for agent i at time t is denoted as $\boldsymbol{b}_t^{(i)}$, and we have $\boldsymbol{b}_t^{(i)} \in \mathcal{R}^n$. All the agents have the same PTB at time 0, and we denote it as $\boldsymbol{b}_0$, which is also referred to as the target's initial position distribution.

With PTB, we are ready to define the robot's *state*. Robot i's state at time t, denoted as $s_t^{(i)}$, is constituted of its current position and its current PTB, i.e., $s_t^{(i)} = (p_t^{(i)}, \boldsymbol{b}_t^{(i)})$. Note that in this article, we use the one-hot encoding scheme to represent $p_t^{(i)}$, which means that $p_t^{(i)} \in \{0,1\}^n$, and in this way, $s_t^{(i)} \in \mathcal{R}^{2n}$. Next, we provide the definitions of two quantities of sensor characteristics, namely, false positive ratio (η_{fp}) and false negative ratio (η_{fn}), which will be used in the online PTB update procedure.

Definition 2 (False positive ratio): The sensor's false positive ratio refers to the probability that the sensor mistakenly believes that the target is currently in the same node as the ego robot. $\forall\, t$, we have $\eta_{\text{fp}} = \mathbb{P}[z_t^{(i)} = 1 | e_t \neq p_t^{(i)}]$.

Definition 3 (False negative ratio): The sensor's false negative ratio refers to the probability that the sensor fails to detect the target when they are in the same node. $\forall\, t$, we have $\eta_{\text{fn}} = \mathbb{P}[z_t^{(i)} = 0 | e_t = p_t^{(i)}]$.

Armed with the definition of η_{fn} and η_{fp}, we state the online calculation process of $\boldsymbol{b}_t^{(i)}$ with the following theorem.

Theorem 1 (The online PTB update theorem): With $\hat{\boldsymbol{\Gamma}}$, $\boldsymbol{b}_{t-1}^{(i)}$, $p_t^{(i)}$, $z_t^{(i)}$, η_{fp}, and η_{fn}, robot i's PTB at time t is updated as

$$\boldsymbol{b}_t^{(i)} \propto \boldsymbol{\Lambda}\hat{\boldsymbol{\Gamma}}\boldsymbol{b}_{t-1}^{(i)} \tag{1}$$

where $\boldsymbol{\Lambda} \in \mathcal{R}^{n\times n}$ is a diagonal matrix, with its elements at the main diagonal set as

$$\lambda_{jj} = \begin{cases} \eta_{\text{fn}}^{1-z_t^{(i)}}(1-\eta_{\text{fn}})^{z_t^{(i)}}, & \text{if } j = p_t^{(i)} \\ \eta_{\text{fp}}^{z_t^{(i)}}(1-\eta_{\text{fp}})^{1-z_t^{(i)}}, & \text{if } j \neq p_t^{(i)} \end{cases}. \tag{2}$$

We defer the proof process of Theorem 1 to Appendix A, to keep the article's main contents succinct. Note that 1) for $t = 1$, we assign $\boldsymbol{b}_{t-1}^{(i)} = \boldsymbol{b}_0^{(i)} = \boldsymbol{b}_0$; and 2) since $\boldsymbol{b}_t^{(i)}$ is essentially a vector of probabilities, we can calculate the exact values through normalization with respect to its L_1 norm.

With the online updated PTB ($\boldsymbol{b}_t^{(i)}$) according to (1), and the pretrained policy parameter ($\boldsymbol{\theta}_i$), the robot executes the policy by choosing $a_t^{(i)}$ with the following probability:

$$\mathbb{P}[a_t^{(i)} | s_t^{(i)}] = \pi(s_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta}_i) \tag{3}$$

Algorithm 1: Online Decentralized Execution.

```
Input: (1) robot $i$'s policy network: $\pi(\boldsymbol{\theta}_i)$; (2) sensor's
       false positive ratio: $\eta_{\text{fp}}$; (3) false negative ratio:
       $\eta_{\text{fn}}$; (4) robot $i$'s initial PTB: $\boldsymbol{b}_0^{(i)} = \boldsymbol{b}_0$; (5)
       estimated target's motion dynamics $\hat{\boldsymbol{\Gamma}}$; (6)
       target's motion dynamics: $\boldsymbol{\Gamma}$ (for simulation
       purpose only);
Output: robot $i$'s decision making sequence: $a_t^{(i)}$;
        robot $i$'s updated PTB sequence: $\boldsymbol{b}_t^{(i)}$;
Init: $t \leftarrow 0$; $p_t^{(i)}$; $e_t$; $z_t^{(i)}$;
1 while the target is not captured do
2     $s_t^{(i)} = (p_t^{(i)}, \boldsymbol{b}_t^{(i)})$;
3     Robot $i$ chooses $a_t^{(i)}$ according to Eq. (3),
      i.e., $a_t^{(i)} \sim \pi(s_t, a; \boldsymbol{\theta}_i)$;
4     Robot $i$ executes $a_t^{(i)}$ and reaches $p_{t+1}^{(i)}$;
5     The target reaches $e_{t+1}$ based on $e_t$ and $\boldsymbol{\Gamma}$;
6     Robot $i$ observes $z_{t+1}^{(i)}$ based on $e_{t+1}$ and $p_{t+1}^{(i)}$;
7     Robot $i$ updates its PTB $\boldsymbol{b}_{t+1}^{(i)}$ according to Eq. (1);
8     $t \leftarrow t + 1$;
9 Final.
```

where $s_t^{(i)} = (p_t^{(i)}, \boldsymbol{b}_t^{(i)})$ and $\pi(s_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta}_i)$ is the policy network parameterized by $\boldsymbol{\theta}_i$. While the implementation details will be introduced in Section V, we note here that the policy network, $\pi(s_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta}_i)$, is a fully connected feedforward neural network, which maps the featured states, i.e., $s_t^{(i)}$ to the action selection probabilities.

The pseudocode of the online DE module for robot i is presented in Algorithm 1. Note that, in Algorithm 1, the target's movement (line 5) is for simulation purpose only, and the robot does not need e_t to execute its decision-making policy. Since there are both false negative and false positive detection rates from the sensors, we deem the target as "captured" by robot i only when $e_t = p_t^{(i)}$ and in the meanwhile $z_t^{(i)} = 1$.

B. *Offline Centralized Training*

The offline CT module provides two functionalities: 1) estimating $\boldsymbol{b}_0$ and $\hat{\boldsymbol{\Gamma}}$; and 2) updating each agent's policy network's parameters, i.e., $\boldsymbol{\theta}_i$, with the CE-PG. This subsection presents the first functionality of the offline CT module, and we introduce the CE-PG with a new subsection, as it serves as the core part of multirobot decision making for efficient target search. The following theorem presents the posterior estimation process of $\boldsymbol{b}_0$.

Theorem 2 (The posterior initial PTB update theorem): With η_{fp} and η_{fn}, the a priori initial PTB $\boldsymbol{b}_0$, and $\forall\, i \in \{1, 2, \ldots, N\}$: $p_0^{(i)}$, $z_0^{(i)}$, the posterior initial PTB $\boldsymbol{b}_0$ is updated as

$$\boldsymbol{b}_0 \propto \boldsymbol{\Lambda}_0 \boldsymbol{b}_0 \tag{4}$$

where $\boldsymbol{\Lambda}_0 \in \mathcal{R}^{n\times n}$ is a diagonal matrix, with its elements at the main diagonal set as

$$\lambda_{jj}^{(0)} = \prod_{i=1}^{N}\left(\eta_{\text{fp}}^{\mathbb{1}\{j\neq p_0^{(i)}\}z_0^{(i)}}(1-\eta_{\text{fp}})^{\mathbb{1}\{j\neq p_0^{(i)}\}(1-z_0^{(i)})}\right) \times \prod_{i=1}^{N}\left(\eta_{\text{fn}}^{\mathbb{1}\{j=p_0^{(i)}\}(1-z_0^{(i)})}(1-\eta_{\text{fn}})^{\mathbb{1}\{j=p_0^{(i)}\}z_0^{(i)}}\right) \quad (5)$$

where $\mathbb{1}\{\text{statement}\}$ is the indicator function, which returns 1 if the statement is true, and returns 0 if the statement is false.

While the proof process of Theorem 2 is deferred to Appendix B, here, we wish to state that the derivation process makes use of the Bayesian rule to calculate the conjugate relationship between the prior and posterior $\boldsymbol{b}_0$. Before presenting the update process of the $\hat{\boldsymbol{\Gamma}}$, we present the definition of collective PTB and then lay down the *offline centralized* collective PTB update theorem, which makes use of all the robots' position and observation information. We will deliver the proof of Theorem 3 in Appendix C.

Definition 4 (Collective PTB): The multirobot system's collective PTB at time t, denoted as $\boldsymbol{b}_t$, is defined as the estimation of the target's position distribution at time t when considering all the robots' position and observation sequences up to time t.

Theorem 3 (The offline collective PTB update theorem): With $\hat{\boldsymbol{\Gamma}}, \eta_{\text{fp}}, \eta_{\text{fn}}, \forall\, i \in \{1, 2, \ldots, N\}$: $p_t^{(i)}, z_t^{(i)}$, and the collective PTB at time $t-1$: $\boldsymbol{b}_{t-1}$, the collective PTB at time t is updated as

$$\boldsymbol{b}_t \propto \boldsymbol{\Lambda}_c \hat{\boldsymbol{\Gamma}} \boldsymbol{b}_{t-1} \quad (6)$$

where $\boldsymbol{\Lambda}_c \in \mathcal{R}^{n\times n}$ is a diagonal matrix, with its elements at the main diagonal set as

$$\lambda_{jj}^{(c)} = \prod_{i=1}^{N}\left(\eta_{\text{fp}}^{\mathbb{1}\{j\neq p_t^{(i)}\}z_t^{(i)}}(1-\eta_{\text{fp}})^{\mathbb{1}\{j\neq p_t^{(i)}\}(1-z_t^{(i)})}\right) \times \prod_{i=1}^{N}\left(\eta_{\text{fn}}^{\mathbb{1}\{j=p_t^{(i)}\}(1-z_t^{(i)})}(1-\eta_{\text{fn}})^{\mathbb{1}\{j=p_t^{(i)}\}z_t^{(i)}}\right). \quad (7)$$

With the updated collective PTB from (6), we can estimate $\hat{\boldsymbol{\Gamma}}$ with the maximum likelihood estimate as follows:

$$\hat{\boldsymbol{\Gamma}} = \boldsymbol{B}_{1:T}\boldsymbol{B}_{0:T-1}^{\top}(\boldsymbol{B}_{0:T-1}\boldsymbol{B}_{0:T-1}^{\top})^{-1} \quad (8)$$

where T is the length of the position/observation sequence, and $\boldsymbol{B}_{t:k} = [\boldsymbol{b}_t, \boldsymbol{b}_{t+1}, \ldots, \boldsymbol{b}_k]$. The derivation process is straightforward and, hence, is omitted in this article. Note that, theoretically, one can loop between (6) and (8) for improved estimates of both the collective PTB and target's motion dynamics, which is in accordance to the expectation–maximization algorithm [43] in most unsupervised machine learning methods. However, in practice, we find that one round of update for the MuRES problem is enough to have well-behaved collective PTB and $\hat{\boldsymbol{\Gamma}}$ values, and thus, we stick with only one round of the collective PTB and $\hat{\boldsymbol{\Gamma}}$ updates in the CT procedure.

C. Cross-Entropy Regularized Policy Gradient

This subsection introduces CE-PG's policy optimization process, which updates each agent's parameterized policy network. The underlying rationale of CE-PG is to maximize the individual robot's expected return, i.e., $J(\pi(\boldsymbol{\theta}_i))$ and, in the meanwhile, disperse the robots from each other through maximizing the cross entropy. In the following, we begin with the definition of cross entropy between robots and then introduce CE-PG's policy optimization objective followed by the objective's gradient derivation process.

Definition 5 (Cross entropy from robot j to robot i): The cross entropy from robot j to robot i, denoted as $\mathcal{H}(\pi(\boldsymbol{\theta}_j), \pi(\boldsymbol{\theta}_i))$, refers to the *expected* cross entropy with respect to robot i's position sequence, i.e., $\mathcal{H}(\pi(\boldsymbol{\theta}_j), \pi(\boldsymbol{\theta}_i)) = -(1/T_i)\sum_{t=0}^{T_i} \pi(s_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta_j})\log(\pi(s_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta_i}))$, where T_i denotes the length of robot i's position sequence.

Note that the smaller $\mathcal{H}(\pi(\boldsymbol{\theta}_j), \pi(\boldsymbol{\theta}_i))$ is, the more robot j "agrees" with robot i's policy. While in MuRES, we do not want the robots to conglomerate. Thus, we want to add a regularization term to disperse the robots from each other, which corresponds to the cross-entropy *maximization*.

The objective of CE-PG for robot i, denoted as $\tilde{J}(\boldsymbol{\theta}_i)$, is to *maximize* the weighted summation of robot i's expected return and the average cross entropy from all the other robots to robot i, which is stated as

$$\tilde{J}(\boldsymbol{\theta}_i) = \beta_i J(\pi(\boldsymbol{\theta}_i)) + \frac{1-\beta_i}{N-1}\sum_{j\neq i}\mathcal{H}\left(\pi(\boldsymbol{\theta}_j), \pi(\boldsymbol{\theta}_i)\right) \quad (9)$$

where $0 \leq \beta_i \leq 1$ is robot i's balance parameter between the two subobjectives. Note that different robots in the robot team may have different balance parameter values. The gradient of $\tilde{J}(\boldsymbol{\theta}_i)$ can be expressed as

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}_i}\left(\tilde{J}(\boldsymbol{\theta}_i)\right) = \beta_i \boldsymbol{\nabla}_{\boldsymbol{\theta}_i}(J(\pi(\boldsymbol{\theta}_i))) + \frac{1-\beta_i}{N-1}\sum_{j\neq i}\boldsymbol{\nabla}_{\boldsymbol{\theta}_i}(\mathcal{H}(\pi(\boldsymbol{\theta}_j), \pi(\boldsymbol{\theta}_i))) \quad (10)$$

where the first term, i.e., $\boldsymbol{\nabla}_{\boldsymbol{\theta}_i}(J(\pi(\boldsymbol{\theta}_i)))$, can be calculated from the canonical PG method in a straightforward way, and we have

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}_i}\left(J(\pi(\boldsymbol{\theta}_i))\right) = \sum_{t=0}^{T_i}\left(\boldsymbol{\nabla}_{\boldsymbol{\theta}_i}\log(\pi(s_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta}_i))\,\mathbb{E}\left(G(s_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta}_i)\right)\right). \quad (11)$$

In (11), T_i has the same meaning as stated in Definition 5 and $G(s_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta}_i)$ refers to the accumulated rewards if the robot starts from $s_t^{(i)}$, executes $a_t^{(i)}$, and follows $\pi(\boldsymbol{\theta}_i)$ thereafter. The second term (cross entropy) in (10), i.e., $\boldsymbol{\nabla}_{\boldsymbol{\theta}_i}(\mathcal{H}(\pi(\boldsymbol{\theta}_j), \pi(\boldsymbol{\theta}_i)))$, is calculated as

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}_i}\left(\mathcal{H}(\pi(\boldsymbol{\theta}_j), \pi(\boldsymbol{\theta}_i))\right) = -\frac{1}{T_i}\sum_{t=0}^{T_i}\pi(s_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta_j})\boldsymbol{\nabla}_{\boldsymbol{\theta}_i}\log(\pi(s_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta_i})). \quad (12)$$

With the derivative of $\tilde{J}(\boldsymbol{\theta}_i)$, one can update robot i's policy network's parameter with the gradient ascent algorithm as follows:

```
Algorithm 2: Offline Centralized Training.
    Input: (1) sensor's false positive ratio: $\eta_{\text{fp}}$; (2) false
        negative ratio: $\eta_{\text{fn}}$; (3) $\forall\, i \in \{1, 2, \ldots, N\}$:
        $p^{(i)}_{\leq T_i}$, $z^{(i)}_{\leq T_i}$; (4) the apriori initial collective
        PTB: $\boldsymbol{b}_0$; (5) $\forall\, i \in \{1, 2, \ldots, N\}$: $\beta_i$ (balance
        parameter), $\alpha_i$ (learning rate);
    Output: (1) updated estimated target's motion
        dynamics $\hat{\boldsymbol{\Gamma}}$; (2) the posterior initial PTB: $\boldsymbol{b}_0$;
        (3) robot $i$'s updated policy network: $\pi(\boldsymbol{\theta}_i)$;
    Init: $t \leftarrow 0$; $\hat{\boldsymbol{\Gamma}} \leftarrow \hat{\boldsymbol{\Gamma}}_0$;
1   Update the posterior initial PTB $\boldsymbol{b}_0$ based on Eq. (4);
2   while the target is not captured do
3       $t \leftarrow t + 1$;
4       Update $\boldsymbol{b}_t$ with Eq. (6);
5   Update $\hat{\boldsymbol{\Gamma}}$ according to Eq. (8);
6   foreach $i \in \{1, 2, \ldots, N\}$ do
7       Calculate $\nabla_{\boldsymbol{\theta}_i}\big(J(\pi(\boldsymbol{\theta}_i))\big)$ with Eq. (11);
8       Calculate $\nabla_{\boldsymbol{\theta}_i}\big(\mathcal{H}(\pi(\boldsymbol{\theta}_j), \pi(\boldsymbol{\theta}_i))\big)$ with Eq. (12);
9       Calculate $\nabla_{\boldsymbol{\theta}_i}\big(\tilde{J}(\boldsymbol{\theta}_i)\big)$ with Eq. (10);
10      Update $\boldsymbol{\theta}_i$ with Eq. (13);
11  Final.
```

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_i + \alpha_i \nabla_{\boldsymbol{\theta}_i} \tilde{J}(\boldsymbol{\theta}_i) \tag{13}$$

where $0 < \alpha_i < 1$ is robot i's learning rate.

The pseudocode for the offline CT module is presented in Algorithm 2. In Algorithm 2, we first collect all the robots' trajectories, e.g., position sequences and observation sequences, update the collective PTB, i.e., $\boldsymbol{b}_t$ (from line 1 to line 4) and, hence, reach better estimates of the target's motion dynamics, i.e., $\hat{\boldsymbol{\Gamma}}$ (line 5). After that, we update each robot's policy network, i.e., $\pi(\boldsymbol{\theta}_i)$, with CE-PG (from line 7 to line 10).

D. Computational Complexity Analysis

This subsection analyzes the computational complexity of CE-PG. In this article, we express the computational cost of an operation through the number of floating-point operations (flops). A flop is defined as an addition, subtraction, multiplication, or division of two floating-point numbers [44]. To evaluate the computational complexity of an algorithm, we count the total number of flops, express it as a function (usually a polynomial) of the dimensions of the involved matrices and vectors, and simplify the expression by ignoring all terms except for the leading ones. Note that we focus on analyzing the computational complexity of CE-PG's online DE module, in that this module *dictates* the algorithm's actual reaction time in deployment, while the offline CT module does not need to provide the real-time reaction functionalities and, thus, bears the heavy computational load at the server side.

Examining Algorithm 1, we can see that the core computational load happens within the "while" loop (lines 3 and 7 to be more specific), in that the initialization procedure can be done offline. Line 3 corresponds to the robot's action selection process, which requires calculating all $\pi(s_t, a; \boldsymbol{\theta}_i)$ values from neural networks. In practice, we design a three-layer neural network, with the input dimension ($n_i = n$), the number of hidden nodes ($n_h = 2 \times n_i = 2n$) and the output dimension ($n_o = n + m$); therefore, the computational complexity of line 3 is $\mathcal{O}(n_i \times n_h \times n_o) = \mathcal{O}(2n^2(n+m))$. While line 7 refers to the agent-based online PTB update process, referring to (1), we can see that executing line 7 has a computational complexity of $\mathcal{O}(n^3)$. Summing line 3's and line 7's computational complexity together and ignoring the constant factors, we conclude that the online DE module has a computational complexity at $\mathcal{O}(n^3 + n^2\, m)$. Note that the online DE module is deployed to each robot, and there is no communication among robots during execution. Therefore, the computational complexity of CE-PG's online DE module does not contain the number of robots, i.e., N.

E. CE-PG+: Robot Team With a Broader Field of View

So far, we have presented both the offline CT module and the online DE module of CE-PG. However, the basic CE-PG algorithm assumes that the robots only have the same node detection ability, which fits to the use case of ground robots. In this subsection, we extend CE-PG's application to the robot team with a broader field of view, e.g., aerial robots, and also discuss the use case of an "all-to-all" communication scheme to see how it enhances CE-PG's performance.

When the robots have a broader field of view than the same node detection, e.g., aerial robots, we need to upgrade Theorems 1 and 3 to the respective versions, which consider the robot's broad sensing capabilities. Before that, we present the following formal definition of sensing range.

Definition 6 (Sensing range): Robot i's sensing range at $p^{(i)}$, denoted as $\delta(p^{(i)})$, refers to the set of nodes that can be detected by the ego robot when it resides in $p^{(i)}$.

With Definition 6, we are ready to deliver the augmented PTB update theorem and the augmented offline collective PTB update theorem, which collectively upgrade the basic CE-PG to CE-PG+. Here, we denote the related CE-PG algorithm for robots with broader field of view as "CE-PG+."

Theorem 4 (The augmented PTB update theorem): With $\hat{\boldsymbol{\Gamma}}$, $\boldsymbol{b}^{(i)}_{t-1}$, $p^{(i)}_t$, $z^{(i)}_t$, η_{fp}, and η_{fn}, robot i's PTB at time t is updated as

$$\boldsymbol{b}^{(i)}_t \propto \boldsymbol{\Lambda}\hat{\boldsymbol{\Gamma}}\boldsymbol{b}^{(i)}_{t-1} \tag{14}$$

where $\boldsymbol{\Lambda} \in \mathcal{R}^{n\times n}$ is a diagonal matrix, with its elements at the main diagonal set as

$$\lambda_{jj} = \begin{cases} \eta_{\text{fn}}^{1-z^{(i)}_t}(1-\eta_{\text{fn}})^{z^{(i)}_t}, & \text{if } j \in \delta(p^{(i)}_t) \\ \eta_{\text{fp}}^{z^{(i)}_t}(1-\eta_{\text{fp}})^{1-z^{(i)}_t}, & \text{if } j \notin \delta(p^{(i)}_t) \end{cases} \tag{15}$$

where $\delta(p^{(i)}_t)$ refers to the set of nodes that can be detected by robot i when it resides in $p^{(i)}_t$.

The proof process of Theorem 4 is quite similar to that of Theorem 1 and, hence, is omitted. Note that with Theorem 4, each robot can update the individual PTB with a broader sensing information, and hence, the individual PTB is closer to the true value than the original one, which makes the decision making more efficient. Similarly, the offline collective PTB update process needs to be adapted to the broader sensing capabilities as follows.

Theorem 5 (The augmented offline collective PTB update theorem): With $\hat{\boldsymbol{\Gamma}}$, η_{fp}, η_{fn}, $\forall\, i \in \{1, 2, \ldots, N\}$: $p_t^{(i)}$, $z_t^{(i)}$, and the collective PTB at time $t-1$: $\boldsymbol{b}_{t-1}$, the collective PTB at time t is updated as

$$\boldsymbol{b}_t \propto \boldsymbol{\Lambda}_c \hat{\boldsymbol{\Gamma}} \boldsymbol{b}_{t-1} \tag{16}$$

where $\boldsymbol{\Lambda}_c \in \mathcal{R}^{n\times n}$ is a diagonal matrix, with its elements at the main diagonal set as

$$\lambda_{jj}^{(c)} = \prod_{i=1}^{N} \left(\eta_{\text{fp}}^{\mathbb{1}\{j \notin \delta(p_t^{(i)})\} z_t^{(i)}} (1-\eta_{\text{fp}})^{\mathbb{1}\{j \notin \delta p_t^{(i)}\}(1-z_t^{(i)})} \right) \times \prod_{i=1}^{N} \left(\eta_{\text{fn}}^{\mathbb{1}\{j \in \delta(p_t^{(i)})\}(1-z_t^{(i)})} (1-\eta_{\text{fn}})^{\mathbb{1}\{j \in \delta(p_t^{(i)})\} z_t^{(i)}} \right). \tag{17}$$

With Theorems 4 and 5, we can upgrade the basic CE-PG algorithm to CE-PG+. The pseudocode of CE-PG+ is omitted here, in that it is quite similar to CE-PG. One can merely replace line 7 of Algorithm 1 with the augmented PTB update equation in (14) and replace line 4 of Algorithm 2 with the augmented collective PTB update equation in (16).

So far, we have upgraded CE-PG to CE-PG+, which fits to robots with a broader field of view, e.g., aerial robots. Another dimension of enhancing the basic CE-PG algorithm is to make use of the interrobot communication during deployment. Currently, CE-PG assumes that the robots do not communicate with each other during deployment, in that the intermittent communication will alter the decision-making process of the individual robot and make the overall environment nonstationary. However, when the all-to-all communication among robots is always available, we can reach a centralized CE-PG algorithm, named as "C-CE-PG," which updates the collective PTB with all the robots' observations at each time step and broadcasts the real-time collective PTB to each robot. With C-CE-PG, each robot is having a much better PTB information, and thus, the decision-making process is more efficient than that of CE-PG. We will evaluate and compare the performance of CE-PG, CE-PG+, and C-CE-PG in the next section.

V. Simulation Results and Analysis

In this section, we evaluate and compare CE-PG's performance with the state-of-the-art MuRES algorithms in a range of MuRES test environments. For state of the art, we select 1) the finite horizon path enumeration (FHPE) method proposed in [22]; 2) FHPE's improvement with implicit coordination among robots through sequential allocation (FHPE-SA) [22]; 3) MILP for MuRES path planning proposed in [20]; and 4) MILP's distributed implementation version (D-MILP) [20]. In addition, since CE-PG is essentially an MARL algorithm under the CTDE scheme, we also apply canonical MARL algorithms under the CTDE scheme, i.e., multiagent deep deterministic policy gradient (MADDPG) [45], FACMAC [46], MASAC [47], and MAMBPO [48], for the MuRES problem and include them as the baseline algorithms. The algorithm-related parameter configurations are summarized in Table II. Note that: 1) for FHPE, the computational complexity is too high, and thus, we set the planning horizon (h) to be 2, while for its distributed version (FHPE-SA), we set h to be 5; and 2) since FACMAC, MASAC, and MAMBPO have never been applied to the MuRES problem before, we provide the specifications in the publicly available code repository, detailing the related parameter configurations and state-space definition in the MuRES domain. While for MADDPG, there exists a prior work [17] for multirobot search, and we adopt related parameters specified in this article for the implementation.

TABLE II
Algorithm's Main Parameters

Param	Description	Algorithm	Value
h	planning horizon	FHPE, FHPE-SA, MILP, D-MILP	5 (2)
γ	discount factor	FHPE, FHPE-SA, MILP, D-MILP	0.97
$\text{E}_{\max}$	maximum # of episodes	CE-PG, MADDPG	4×10^5
$\boldsymbol{b}_0$	initial PTB	CE-PG	$(1/n, 1/n, \ldots, 1/n)^\top$
$\hat{\Gamma}_0$	initial estimate of Γ	CE-PG	$\boldsymbol{I}$ (the identity matrix)
β_i	balance parameter for robot i	CE-PG	0.5
α_i	learning rate for robot i	CE-PG, MADDPG	1×10^{-3}

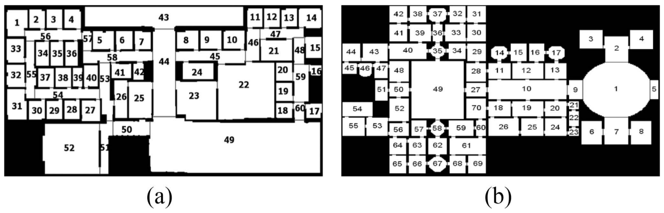


Fig. 3. Canonical MuRES test environments from [22], each room is associated with a corresponding node number (a) OFFICE. (b) MUSEUM.

For MuRES test environments, we select two canonical ones in the multirobot search domain, namely, OFFICE and MUSEUM, which are shown in Fig. 3(a) and (b), respectively. All the algorithms are implemented in Python 3.7, with source code publicly available.[4] We evaluate the algorithms on a 2.30-GHz, Intel(R) Core(TM) i5 8300H CPU computer with the 64-bit version of Windows 10 operating system and 16-GB RAM. In the following subsections, we will 1) showcase how CE-PG is able to generate diverse strategies of the robot team with exactly the same PTB information, with a simple yet illustrative example; 2) evaluate the impact of β_i in the two MuRES test environments and select the best β_i values for the subsequent baseline comparison; 3) compare the performance of CE-PG, CE-PG+, and C-CE-PG in OFFICE and MUSEUM environments; 4) benchmark CE-PG's performance with state-of-the-art MuRES solutions as well as canonical MARL baselines for the

[4]Source code is available at https://github.com/kkallengit/CE-PG.git

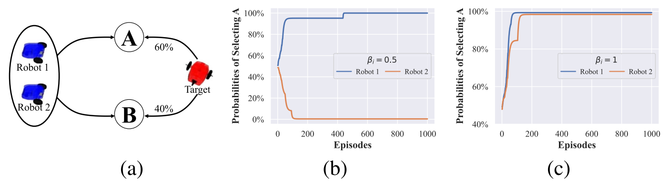


Fig. 4. Simple search scenario and learning curve illustration. (a) Search scenario. (b) Learning curve for CE-PG. (c) Learning curve for PG.

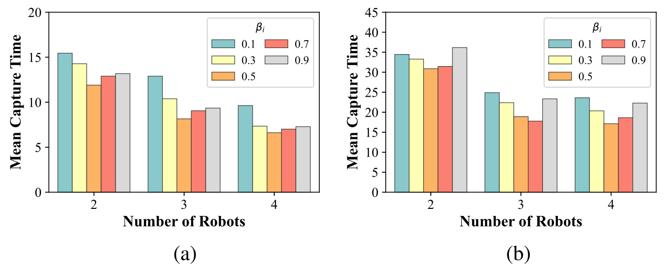


Fig. 5. Impact of β_i to CE-PG for the two MuRES test environments. (a) OFFICE environment. (b) MUSEUM environment.

MuRES problem; 5) evaluate and compare the impact of sensor inaccuracies, i.e., η_{fn} and η_{fp}, to the performance of CE-PG and other baseline algorithms; and 6) compare CE-PG's robustness with MARL baselines when one or multiple robots malfunction during deployment.

A. Simple Use Case

In this subsection, we construct a simple yet illustrative multirobot search scenario, as shown in Fig. 4(a), to illustrate the per-episode training processes of two robots and how the two robots with exactly the same PTB information learn to make different/diversified decisions due to the cross-entropy regularization term.

In Fig. 4(a), two robots, starting at the robot depot, try to search for the target, which starts at the target spot. Both the robots and the target have two actions, i.e., go to "A" or go to "B," and the simulation stops after one time step. If any robot resides in the same node with the target, e.g., both Robot 1 and the target reside in node "A" after one time step, a reward of 1 is given to the robot team. Otherwise, reward is 0. The target has a 60% probability to go to "A" and 40% probability to go to "B," and we set $\beta_i = 0.5$ for both robots. Fig. 4(b) shows one representative learning curve of both robots' policies (since any robot can only select to go to "A" or "B," we just simply plot each robot's policy's probability of selecting "A."). In the figure, we can see that, as the number of episodes increases, the robots separate their action selection preferences so as to simultaneously cover both "A" and "B." On the other hand, Fig. 4(c) shows the learning curves of vanilla PG without the cross-entropy regularization term by simply setting $\beta_i = 1$. In the figure, we can see that, without the cross-entropy regularization, both robots will conglomerate to "A" for the large return. Note that for the specific example, which robot ultimately converges to "A" depends on the randomly initialized policy parameters. In theory, both Robot 1 and Robot 2 have equal probabilities of converging to "A," but they cannot concurrently converge to "A" due to the cross-entropy term, and Fig. 4(b) just indicates one representative learning curve.

B. Evaluating the Impact of β_i

In this subsection, we evaluate the impact of β_i to CE-PG's performance in the two canonical MuRES test environments, i.e., OFFICE and MUSEUM. First, β_i can (theoretically) be set differently for different robots, which endows us with a wider selection range than simply setting β_i to be the same value across different robots. However, in practice, it is difficult to tune the parameters if we set β_i differently. Therefore, we keep β_i to the same value for all robots and test the MuRES performance for a range of β_i values, i.e., $\beta_i \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, and select the best one on average for the subsequent comparison with baseline algorithms.

We set up the MuRES for a nonadversarial moving target problem in the two canonical MuRES test environments as follows: 1) the robots are initialized at node 43 for OFFICE and at node 1 for MUSEUM, respectively; 2) the target is randomly initialized in the environments according to the discrete uniform distribution and moves randomly with respect to its available actions, i.e., at each time step, the target moves with equal probability to one of the adjacent nodes or stays in the same node; 3) CE-PG's policy network for each robot is a randomly initialized three-layer neural network, with $n_i = n$ inputs, $n_h = 2 \times n_i = 2n$ hidden nodes, and $n_o = m + n$ output channels, with "softmax" as the activation function; 4) the maximal allowed capture time is $3n$, i.e., $t_{\text{cap}} = 3n$ if the employed MuRES algorithm fails to detect the target before $3n$ time steps; and 5) we set $\eta_{\text{fn}} = \eta_{\text{fp}} = 0$ for fairness in comparison, because neither FHPE nor FHPE-SA applies to the use case of stochastic sensors. Note that the same MuRES problem setup will be used in the remaining subsections as well. Fig. 5 shows the performance of different β_i values across different settings of the MuRES problem. When making comparisons with baseline algorithms, we select $\beta_i = 0.5$, which yields the best performance on average for the remaining experiments.

C. Comparison Among CE-PG, CE-PG+, and C-CE-PG

In Section IV-E, we extend CE-PG to CE-PG+, which fits to robots with a broader field of view, e.g., aerial robots, and also consider the use case of an all-to-all communication scheme by proposing C-CE-PG. In this subsection, we compare the performance of CE-PG, CE-PG+, and C-CE-PG in the two MuRES test environments following the problem setup as stated in the last subsection. Fig. 6 shows the comparative simulation results in both OFFICE and MUSEUM. In the figure, we can see that both C-CE-PG and CE-PG+ are having a better performance than CE-PG, in that CE-PG+ augments the agent's sensing capability and C-CE-PG endows each agent with an online collective PTB information, which is more accurate than the individual one.

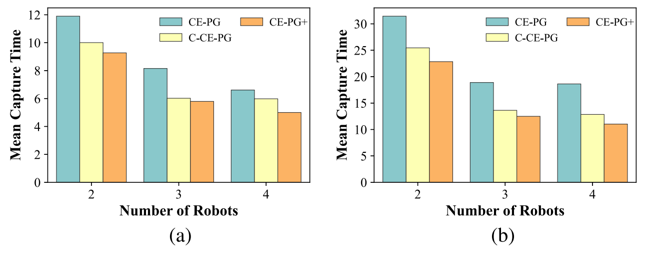


Fig. 6. Performance comparison of CE-PG with CE-PG+ and C-CE-PG for OFFICE and MUSEUM; CE-PG+ refers to the algorithm for robot team with a broader field of view; C-CE-PG refers to the algorithm that robots can interact with all other robots during the search process. (a) OFFICE environment. (b) MUSEUM environment.

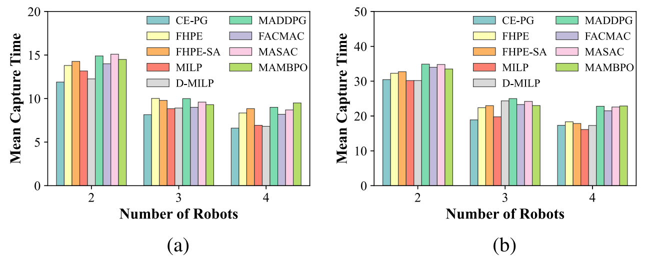


Fig. 7. Performance comparison of CE-PG with state-of-the-art MuRES solutions and canonical MARL algorithms in OFFICE and MUSEUM. (a) OFFICE environment. (b) MUSEUM environment.

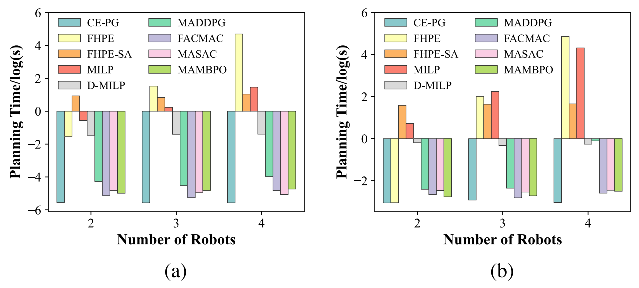


Fig. 8. Decision-making time comparison (time scale is log seconds.). (a) OFFICE environment. (b) MUSEUM environment.

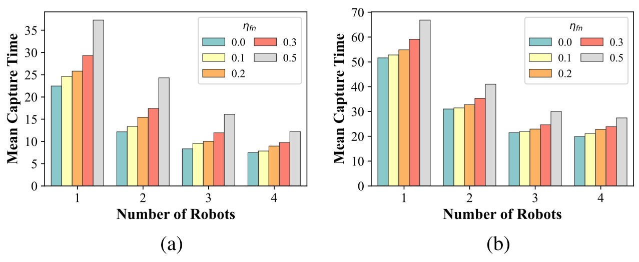


Fig. 9. Impact of η_{fn} to CE-PG for the two MuRES test environments. (a) Impact of η_{fn}: OFFICE. (b) Impact of η_{fn}: MUSEUM.

D. Performance Comparison With State of the Art

This subsection compares CE-PG's performance and efficiency with state-of-the-art MuRES solutions as well as the canonical MARL algorithms. Fig. 7(a) and (b) shows the performance comparison of CE-PG with state of the art in OFFICE and MUSEUM, respectively. We rerun each experiments 1500 independent times and report the corresponding algorithm's mean target capture time (t_{cap}) for different number of robots. In the figure, we can see that: 1) as the number of robots increases, all the algorithms' mean target capture time decreases, which is reasonable; and 2) CE-PG achieves the top three performance across different MuRES problem setups; moreover, the disparity between CE-PG's performance with the baseline algorithms' best performance is barely discernible. Here, we wish to note that the target's motion dynamics and the target's initial position distribution are not provided to CE-PG as the inputs; instead, they are learned while the robots are interacting with the environments in the offline CT stage. For FHPE, MILP, and the corresponding distributed versions, both the target's motion dynamics ($\boldsymbol{\Gamma}$) and the target's initial position distribution ($\boldsymbol{b}_0$) are provided as inputs, which is unrealistic in certain application scenarios.

The decision-making time comparisons are presented in Fig. 8(a) and (b), respectively. From the figures, we can see that the decision-making time of all MARL algorithms including CE-PG is significantly smaller than that of FHPE, MILP, and their variations for both MuRES test environments. The underlying reason is that for learning-based algorithms, during deployment, the decision-making time depends solely on the neural network's forward computation time, which is usually in the range of milliseconds, while for planning-based methods, such as FHPE and MILP, the decision-making process involves solving the formulated mathematical problem with enumeration or commercial solvers, which incur nonpolynomial complexity and cost a lot of time, as shown in Fig. 8.

E. Impacts of Sensor Characteristics: η_{fn} and η_{fp}

In the last subsection, we have compared the performance and efficiency of CE-PG with state of the art in two canonical MuRES test environments. However, we set $\eta_{\text{fn}} = \eta_{\text{fp}} = 0$ for comparison fairness. In this subsection, we evaluate the impacts of η_{fn} and η_{fp} to the performance of CE-PG. Fig. 9(a) and (b) shows the performance changing trends of CE-PG with different η_{fn} values for the two MuRES test environments when we set $\eta_{\text{fp}} = 0$. We rerun the same experimental setup for 1500 independent times and report the target's mean capture time as the performance indicator.

In Fig. 9, we observe that with the same experimental setup for other parameters, i.e., N, β_i, increasing the value of η_{fn} will increase the target's mean capture time. This is reasonable, in that the larger η_{fn} is, the more *inaccurate* the related sensor is. Thus, it will result in more target searching time on average. However, on the flip side, the impact of η_{fp} is not on the target's capture time, but on the overall multirobot system's false alarm rate. Here, we define the multisystem's false alarm rate as the system's total number of times that false alarm happens. Fig. 10(a) and (b) shows the false alarm rates' changing trends of CE-PG with different η_{fp} values for the two MuRES test environments while setting $\eta_{\text{fn}} = 0$.

From Fig. 10, we can see that with increase η_{fp} values, the multirobot system's false alarm rate increases. Another phenomenon is that the number of robots does not affect the

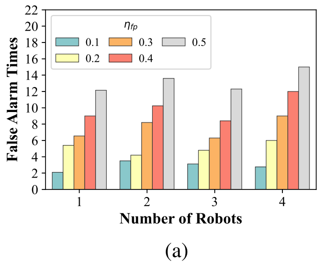


Fig. 10. Impact of η_{fp} to CE-PG for the two MuRES test environments. (a) Impact of η_{fp}: OFFICE. (b) Impact of η_{fp}: MUSEUM.

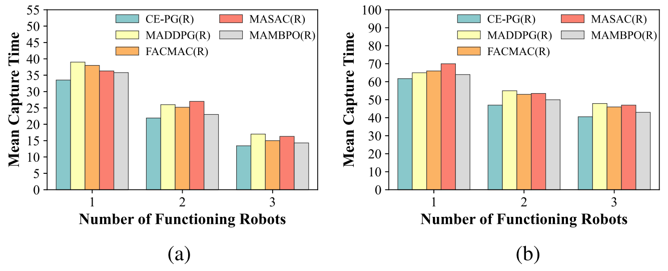


Fig. 12. Robustness comparison of CE-PG with other MARL algorithms in OFFICE and MUSEUM environments. (R) refers to the case that there is one robot is dropped out during deployment. Horizontal labels show the number of *remaining* functioning robots. (a) OFFICE environment. (b) MUSEUM environment.

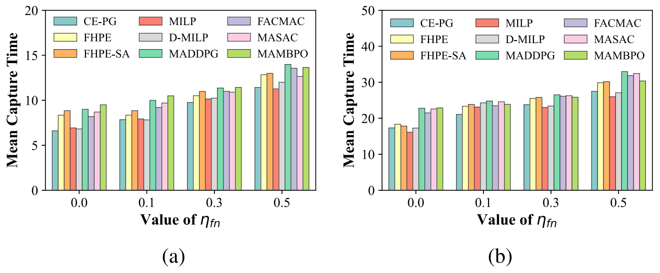


Fig. 11. Performance comparison of CE-PG with state of the art in OFFICE and MUSEUM environments for different η_{fn} values, with $N = 4$ robots. (a) OFFICE environment. (b) MUSEUM environment.

false alarm rates, in that although more robots will result in smaller search time, but with each robot subject to a certain false alarm probability, the overall system's false alarm rate does not decrease. In addition, note that in both experiments, we set the maximum value of both η_{fn} and η_{fp} as 0.5, which means that the sensor's false negative detection probability (η_{fn}) is less than the sensor's *true* negative detection probability, and similarly, the sensor's false positive detection probability (η_{fp}) is less than the sensor's *true* positive detection probability.

So far, the evaluation of the impacts of sensor characteristics, i.e., η_{fn} and η_{fp}, to CE-PG is stand-alone, without making comparisons with baseline algorithms, and it is more convincing to compare the related performance with baseline algorithms. First, since all selected baseline algorithms do not consider η_{fp}, and thus, we cannot perform the comparative evaluations. For η_{fn}, we set the number of robots as $N = 4$ and evaluate the performance of CE-PG and different baseline algorithms for a range of η_{fn} values, i.e., $\eta_{\mathrm{fn}} \in \{0.0, 0.1, 0.3, 0.5\}$, and report the comparative results in Fig. 11. In the figure, we can see that CE-PG achieves the top three performance (sometimes MILP or D-MILP performs better) for different η_{fn} values in the two environments. The underlying reason is that CE-PG takes η_{fn} into calculation during the PTB derivation process, while most baseline algorithms (except for MILP and D-MILP) do not consider the effect of η_{fn} explicitly.

F. Robustness Evaluation With Random Robot Failures

As we have claimed in Section IV, one of the main characteristics of CE-PG is the DE, which means that during the execution process, the robots do not need to communicate with each other and will *commit* to a predetermined policy for the target search process. During the execution process, the robot do not rely on other robots' information to make decisions, which makes CE-PG naturally[5] robust to random robot failures. In this subsection, we evaluate and compare the robustness performance of CE-PG with state of the art through randomly withdrawing a robot (mimic a malfunctioning robot) from the multirobot system.

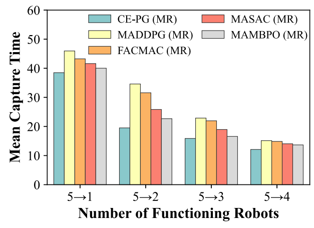
Fig. 12(a) and (b) shows the comparative results of the robustness performance between CE-PG and other MARL algorithms. In the figure, we can see that CE-PG has the best performance in various settings when we withdraw one robot out of the robot team during execution. Note that Fig. 12(a) and (b) does not include planning-based algorithms, such as FHPE, FHPE-SA, MILP, and D-MILP, in that planning-based methods inherently assume that the complete knowledge of all the functioning robots is available all the time, and they cannot be applied to the case with one or more malfunctioning robots.

One more challenging scenario for MuRES problem's robustness evaluation is the multirobot system's performance in face of *multiple* malfunctioning robots. We evaluate and compare the MARL algorithms' performance when we withdraw multiple robots from the environment during the execution stage in Fig. 13 (we set the initial number of robots as $N = 5$). In the figure, we can see that CE-PG achieves the best performance for the two MuRES test environments. We conjecture that the underlying reason is CE-PG endows the robots with a completely DE policy without communication or coordination during execution, while all other MARL algorithms are working on top of a common factorized value function, which changes greatly with the varying number of functioning robots.

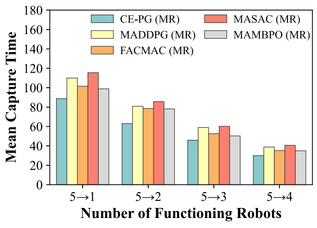
VI. Experimental Results and Analysis

So far, we have conducted simulation comparisons between CE-PG and state of the art for the MuRES problem in the OFFICE and MUSEUM environments. Next, we deploy CE-PG

[5]Here, the word "naturally" means that we do not design the robustness controller intentionally, but CE-PG automatically functions in face of random robot failures, in that the robot does not need to communicate with the teammate during the DE stage.

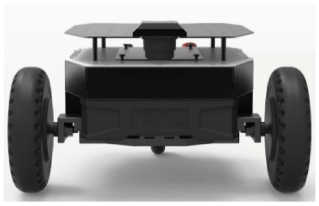


(a)

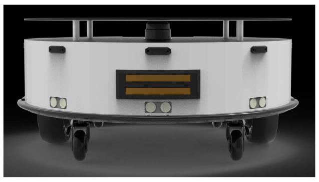


(b)

Fig. 13. Robustness comparison of CE-PG with other MARL algorithms in OFFICE and MUSEUM environments. (MR) refers to there are multiple robots dropped out during deployment. Horizontal labels show the transition from five robots to the corresponding number of remaining functioning robots. For example, 5 → 1 means that there are five robots during training, and during deployment, there is only one remaining functional robot. (a) OFFICE environment. (b) MUSEUM environment.

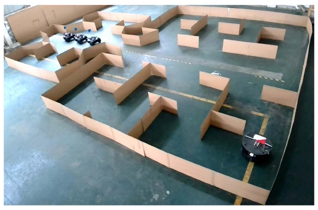


(a)

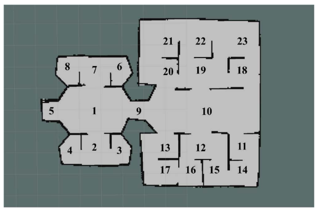


(b)

Fig. 14. Autonomous robot testbed (DM3008) and the moving target (C30) for MuRES experiments. (a) DM3008 robot. (b) C30 moving target.



(a)



(b)

Fig. 15. Indoor environment and the mapping result for MuRES tests. (a) Indoor environment. (b) Constructed map.

to a real multirobot system and test its functionality in a self-constructed indoor environment for the moving target search.

The autonomous robot testbed is a DM3008 differential drive robot,[6] as shown in Fig. 14(a), with an embedded single beam LiDAR (LDS-50C-2) for map construction and obstacle detection. The DM3008 robot already offers the simultaneous localization and mapping functionality, as well as the autonomous navigation and obstacle avoidance module, which navigates the robot within the preconstructed map. We integrate the CE-PG algorithm, which essentially assigns the next-step goal position to the robot, into DM3008. The moving target is a C30 differential drive robot with the random next-step goal position. The indoor environment, as shown in Fig. 15(a), mimics one half of the MUSEUM test environments. The mapping result of the indoor environment is shown in Fig. 15(b), with labeled topological grids for trace representation in Table III.

[6]Details of DM3008 and C30 can be referred to www.puweii.com.

TABLE III
CE-PG FOR A MOVING TARGET SEARCH RESULTS

Experiment #	Capture Time	Capture Node	Target's Trajectory
1	8	18	[14,14,14,11,10,18,18,18]
2	8	18	[14,14,14,11,11,10,10,18]
3	6	10	[15,15,12,10,10,10]
4	4	20	[12,10,19,20]
5	3	8	[7,8,8]
6	9	14	[14,11,10,10,11,14,11,14,14]
7	3	13	[17,13,13]
8	3	20	[10,19,20]
9	4	19	[22,22,23,19]
10	3	10	[12,10,10]

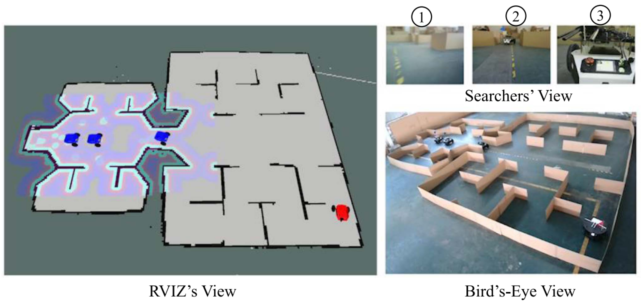


Fig. 16. Snapshot of the multirobot search process.

In the experiment, we deploy three DM3008 robots into the environment and let them cooperatively search for one randomly moving target which is a C30 differential drive robot, as shown in Fig. 14(b). The robots are initialized at node 1, and the moving target is initialized randomly in the environment. While a demonstrative video is uploaded together with the manuscript, and publicly available,[7] here, we present ten sets of the multirobot moving target search results in Table III. In the table, we can see that with three DM3008 robots, we can capture the randomly moving target within the indoor environment in approximately 5.1 time steps.

Fig. 16 shows a snapshot of the operating scenario of the multirobot search process, where the left half figure visualizes the status information from the robot operating system perspective, the three top right subfigures show DM3008 robots' local vision, and the bottom right subfigure shows the bird's-eye view of the real multirobot search process.

VII. Conclusion

This article presented CE-PG for the MuRES problem. CE-PG adopted the CTDE scheme to train and deploy the learning agents into the environments. In addition, during the execution stage, each CE-PG agent used the online Bayesian update procedure to estimate the probabilistic target distribution beliefs and use it together with its position information as the decision-making basis. We evaluated and compared the performance of CE-PG with state-of-the-art multirobot search algorithms in two canonical MuRES test environments, deployed CE-PG to a real multirobot system (three DM3008 differential drive robots), and demonstrated the multirobot search process in a self-constructed indoor environments with satisfying results.

[7]https://github.com/kkallengit/CE-PG.git

Currently, CE-PG applies to the multirobot search for a *non-adversarial* moving target problem. In the future, we would like to improve CE-PG to the use case of an adversarial moving target, which acts to avoid being captured by the robots. Moreover, we are also keen on improving CE-PG to apply to a heterogeneous team of robots, which possess different motion and vision capabilities. Currently, CE-PG does not apply to the robot team with heterogeneous motion dynamics, in that the cross-entropy regularization term assumes that all robots are possessing the same motion capabilities when residing in the same node. In the meanwhile, we are also interested in designing a completely decentralized but coordinated multirobot search algorithm, which does not need any centralized computation and coordinates the robot team on the fly with only local information.

Appendix A
Proof of Theorem 1

Proof: We prove Theorem 1 in its element form as follows:

$$\begin{aligned}
&\mathbb{P}\left(e_t = j | p_t^{(i)}, z_t^{(i)}; \hat{\boldsymbol{\Gamma}}, \eta_{\text{fn}}, \eta_{\text{fp}}\right)\\
&\quad\propto \mathbb{P}\left(p_t^{(i)}, z_t^{(i)} | e_t = j; \eta_{\text{fn}}, \eta_{\text{fp}}\right) \times \mathbb{P}\left(e_t = j; \hat{\boldsymbol{\Gamma}}\right)\\
&\quad\propto \mathbb{P}\left(p_t^{(i)}, z_t^{(i)} | e_t = j; \eta_{\text{fn}}, \eta_{\text{fp}}\right)\\
&\quad\times \sum_{k=1}^{n} \mathbb{P}\left(e_t = j, e_{t-1} = k; \hat{\boldsymbol{\Gamma}}\right)\\
&\quad\propto \eta_{\text{fn}}^{\mathbb{1}\{p_t^{(i)}=j\}\left(1-z_t^{(i)}\right)} \times (1-\eta_{\text{fn}})^{\mathbb{1}\{p_t^{(i)}=j\}z_t^{(i)}}\\
&\quad\times \eta_{\text{fp}}^{\mathbb{1}\{p_t^{(i)}\neq j\}z_t^{(i)}} \times (1-\eta_{\text{fp}})^{\mathbb{1}\{p_t^{(i)}\neq j\}\left(1-z_t^{(i)}\right)}\\
&\quad\times \sum_{k=1}^{n} \mathbb{P}\left(e_t = j | e_{t-1} = k; \hat{\boldsymbol{\Gamma}}\right) \times \mathbb{P}(e_{t-1} = k; \boldsymbol{b}_{t-1}^{(i)})\\
&\quad\propto \lambda_{jj} \times \sum_{k=1}^{n} \hat{\boldsymbol{\Gamma}}(k,j)\boldsymbol{b}_{t-1}^{(i)}(k).
\end{aligned}$$

■

Appendix B
Proof of Theorem 2

Proof: We prove Theorem 2 in its element form as follows:

$$\begin{aligned}
&\mathbb{P}\left(e_0 = j | p_0^{(1)}, \ldots, p_0^{(N)}, z_0^{(1)}, \ldots, z_0^{(N)}; \eta_{\text{fp}}, \eta_{\text{fn}}, \boldsymbol{b}_0\right)\\
&\quad\propto \mathbb{P}\left(p_0^{(1)}, \ldots, p_0^{(N)}, z_0^{(1)}, \ldots, z_0^{(N)} | e_0 = j; \eta_{\text{fp}}, \eta_{\text{fn}}\right)\\
&\quad\times \mathbb{P}(e_0 = j | \boldsymbol{b}_0)\\
&\quad\propto \prod_{i=1}^{N} \mathbb{P}\left(p_0^{(i)}, z_0^{(i)} | e_0 = j; \eta_{\text{fp}}, \eta_{\text{fn}}\right)\\
&\quad\times \mathbb{P}(e_0 = j | \boldsymbol{b}_0)\\
&\quad\propto \prod_{i=1}^{N} \left(\eta_{\text{fp}}^{\mathbb{1}\{j\neq p_0^{(i)}\}z_0^{(i)}} (1-\eta_{\text{fp}})^{\mathbb{1}\{j\neq p_0^{(i)}\}(1-z_0^{(i)})}\right)\\
&\quad\times \prod_{i=1}^{N} \left(\eta_{\text{fn}}^{\mathbb{1}\{j=p_0^{(i)}\}(1-z_0^{(i)})} (1-\eta_{\text{fn}})^{\mathbb{1}\{j=p_0^{(i)}\}z_0^{(i)}}\right)\\
&\quad\times \boldsymbol{b}_0(j)\\
&\quad\propto \lambda_{jj} \times \boldsymbol{b}_0(j).
\end{aligned}$$

■

Appendix C
Proof of Theorem 3

Proof: We prove Theorem 3 in its element form as follows:

$$\begin{aligned}
&\mathbb{P}\left(e_t = j | p_t^{(1)}, \ldots, p_t^{(N)}, z_t^{(1)}, \ldots, z_t^{(N)}; \hat{\boldsymbol{\Gamma}}, \eta_{\text{fn}}, \eta_{\text{fp}}, \boldsymbol{b}_{t-1}\right)\\
&\quad\propto \mathbb{P}\left(p_t^{(1)}, \ldots, p_t^{(N)}, z_t^{(1)}, \ldots, z_t^{(N)} | e_t = j; \eta_{\text{fn}}, \eta_{\text{fp}}\right)\\
&\quad\times \mathbb{P}\left(e_t = j; \hat{\boldsymbol{\Gamma}}\right)\\
&\quad\propto \prod_{i=1}^{N} \mathbb{P}\left(p_t^{(i)}, z_t^{(i)} | e_t = j; \eta_{\text{fn}}, \eta_{\text{fp}}\right)\\
&\quad\times \sum_{k=1}^{n} \mathbb{P}\left(e_t = j, e_{t-1} = k; \hat{\boldsymbol{\Gamma}}\right)\\
&\quad\propto \prod_{i=1}^{N} \left(\eta_{\text{fp}}^{\mathbb{1}\{j\neq p_t^{(i)}\}z_t^{(i)}} (1-\eta_{\text{fp}})^{\mathbb{1}\{j\neq p_t^{(i)}\}(1-z_t^{(i)})}\right)\\
&\quad\times \prod_{i=1}^{N} \left(\eta_{\text{fn}}^{\mathbb{1}\{j=p_t^{(i)}\}(1-z_t^{(i)})} (1-\eta_{\text{fn}})^{\mathbb{1}\{j=p_t^{(i)}\}z_t^{(i)}}\right)\\
&\quad\times \sum_{k=1}^{n} \mathbb{P}\left(e_t = j | e_{t-1} = k; \hat{\boldsymbol{\Gamma}}\right) \times \mathbb{P}(e_{t-1} = k; \boldsymbol{b}_{t-1})\\
&\quad\propto \lambda_{jj}^{(c)} \times \sum_{k=1}^{n} \hat{\boldsymbol{\Gamma}}(k,j)\boldsymbol{b}_{t-1}(k).
\end{aligned}$$

■

References

[1] M. Rajesh and S. R. Nagaraja, "An energy-efficient communication scheme for multi-robot coordination deployed for search and rescue operations," in *Communication and Intelligent Systems*, H. Sharma, M. K. Gupta, G. S. Tomar, and W. Lipo, Eds. Singapore: Springer, 2021, pp. 187–199.

[2] A. V. Nazarova and M. Zhai, "The application of multi-agent robotic systems for earthquake rescue," in *Robotics: Industry 4.0 Issues and New Intelligent Control Paradigms*. New York, NY, USA: Springer, 2020, pp. 133–146.

[3] H. Surmann et al., "Integration of UAVs in urban search and rescue missions," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot.*, 2019, pp. 203–209.

[4] V. Sadhu, C. Sun, A. Karimian, R. Tron, and D. Pompili, "Aerial-DeepSearch: Distributed multi-agent deep reinforcement learning for search missions," in *Proc. IEEE Int. Conf. Mobile Ad Hoc Sensor Syst.*, 2020, pp. 165–173.

[5] J. P. Queralta et al., "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *IEEE Access*, vol. 84, pp. 191617–191643, 2020.

[6] T. Ma, S. Liu, and H. Xiao, "Location of natural gas leakage sources on offshore platform by a multi-robot system using particle swarm optimization algorithm," *J. Natural Gas Sci. Eng.*, vol. 84, 2020, Art. no. 103636.

[7] H. Che, C. Shi, X. Xu, J. Li, and B. Wu, "Research on improved ACO algorithm-based multi-robot odor source localization," in *Proc. IEEE Int. Conf. Robot. Autom. Sci.*, 2018, pp. 1–5.

[8] T. Alam, M. M. Rahman, P. Carrillo, L. Bobadilla, and B. Rapp, "Stochastic multi-robot patrolling with limited visibility," *J. Intell. Robot. Syst.*, vol. 97, no. 2, pp. 411–429, 2020.

[9] L. Freda et al., "3D multi-robot patrolling with a two-level coordination strategy," *Auton. Robots*, vol. 43, no. 7, pp. 1747–1779, 2019.

[10] C. Yu, Y. Dong, Y. Li, and Y. Chen, "Distributed multi-agent deep reinforcement learning for cooperative multi-robot pursuit," *J. Eng.*, vol. 2020, no. 13, pp. 499–504, 2020.

[11] M. Li, W. Yang, Z. Cai, S. Yang, and J. Wang, "Integrating decision sharing with prediction in decentralized planning for multi-agent coordination under uncertainty," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 450–456.

[12] M. E. H. Souidi, A. Siam, Z. Pei, and S. Piao, "Multi-agent pursuit-evasion game based on organizational architecture," *J. Comput. Inf. Technol.*, vol. 27, no. 1, pp. 1–11, 2019.

[13] M. S. A. Khan, T. Ahmed, and M. F. Uddin, "Multi-robot search algorithm using timed random switching of exploration approaches," in *Proc. IEEE Region 10 Symp.*, 2020, pp. 868–871.

[14] M. Abdelkader, S. Güler, H. Jaleel, and J. S. Shamma, "Aerial swarms: Recent applications and challenges," *Curr. Robot. Rep.*, vol. 2, pp. 309–320, 2021.

[15] H. F. Ahmad, M. K. Hardhienata, and K. Priandana, "Integration of N-GCPSO algorithm with spatial particle extension algorithm for multi-robot search," in *Proc. IEEE Int. Conf. Smart Technol. Appl.*, 2020, pp. 1–4.

[16] L. A. Ricciardi and M. Vasile, "Improved archiving and search strategies for multi agent collaborative search," in *Advances in Evolutionary and Deterministic Methods for Design, Optimization Control in Engineering and Sciences*. New York, NY, USA: Springer, 2019, pp. 435–455.

[17] Y. Wang, L. Dong, and C. Sun, "Cooperative control for multi-player pursuit-evasion games with reinforcement learning," *Neurocomputing*, vol. 412, pp. 101–114, 2020.

[18] A. Kehagias, G. Hollinger, and S. Singh, "A graph search algorithm for indoor pursuit/evasion," *Math. Comput. Model.*, vol. 50, no. 9, pp. 1305–1317, 2009.

[19] S. Sundaram and D. W. C. K. Kalyanam, "Pursuit on a graph under partial information from sensors," in *Proc. Amer. Control Conf.*, 2017, pp. 4279–4284.

[20] B. A. Asfora, J. Banfi, and M. Campbell, "Mixed-integer linear programming models for multi-robot non-adversarial search," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6805–6812, Oct. 2020.

[21] H. Lau, S. Huang, and G. Dissanayake, "Probabilistic search for a moving target in an indoor environment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 3393–3398.

[22] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias, "Efficient multi-robot search for a moving target," *Int. J. Robot. Res.*, vol. 28, no. 2, pp. 201–219, 2009.

[23] G. Hollinger, A. Kehagias, and S. Singh, "GSST: Anytime guaranteed search," *Auton. Robots*, vol. 29, no. 1, pp. 99–118, 2010.

[24] A. Kehagias and A. Papazoglou, "An algorithm for limited visibility graph searching," 2021, *arXiv.2105.06150*.

[25] A. Kolling, A. Kleiner, and S. Carpin, "Coordinated search with multiple robots arranged in line formations," *IEEE Trans. Robot.*, vol. 34, no. 2, pp. 459–473, Apr. 2018.

[26] A. Kolling, A. Kleiner, and P. Rudol, "Fast guaranteed search with unmanned aerial vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 6013–6018.

[27] T. D. Parsons, "Pursuit-evasion in a graph," in *Theory and Applications of Graphs*. New York, NY, USA: Springer, 1978, pp. 426–441.

[28] G. Hollinger, A. Kehagias, S. Singh, D. Ferguson, and S. Srinivasa, "Anytime guaranteed search using spanning trees," Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR-08-36, 2008.

[29] J. W. Durham, A. Franchi, and F. Bullo, "Distributed pursuit-evasion without mapping or global localization via local frontiers," *Auton. Robots*, vol. 32, no. 1, pp. 81–95, 2012.

[30] R. J. Marcotte, A. Haggenmiller, G. Ferrer, and E. Olson, "Probabilistic multi-robot search for an adversarial target," Univ. Michigan APRIL Lab., Ann Arbor, MI, USA, Tech. Rep. UMAL2019, Jun. 2019.

[31] A. Quattrini Li, R. Fioratto, F. Amigoni, and V. Isler, "A search-based approach to solve pursuit-evasion games with limited visibility in polygonal environments," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2018, pp. 1693–1701.

[32] J. Berger and N. Lo, "An innovative multi-agent search-and-rescue path planning approach," *Comput. Oper. Res.*, vol. 53, pp. 24–31, 2015.

[33] K.-S. Tseng and B. Mettler, "Near-optimal probabilistic search using spatial fourier sparse set," *Auton. Robots*, vol. 42, no. 2, pp. 329–351, 2018.

[34] W. Sheng, H. Guo, W.-Y. Yau, and Y. Zhou, "PD-FAC: Probability density factorized multi-agent distributional reinforcement learning for multi-robot reliable search," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 8869–8876, Oct. 2022.

[35] X. Qin, X. Li, Y. Liu, R. Zhou, and J. Xie, "Multi-agent cooperative target search based on reinforcement learning," *J. Phys., Conf. Ser.*, vol. 1549, 2020, Art. no. 022104.

[36] H. Tang, W. Sun, A. Lin, M. Xue, and X. Zhang, "A GWO-based multi-robot cooperation method for target searching in unknown environments," *Expert Syst. Appl.*, vol. 186, 2021, Art. no. 115795.

[37] T. A. El-Mihoub, L. Nolle, and C. Tholen, "On localisation errors and cooperative search for a swarm of AUVs," in *Proc. Glob. Oceans: Singapore—US Gulf Coast*, 2020, pp. 1–7.

[38] J. Ebert, F. Berlinger, B. Haghighat, and R. Nagpal, "A hybrid PSO algorithm for multi-robot target search and decision awareness," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 11520–11527.

[39] L. Matignon, G. Laurent, and N. Fort-Piat, "Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems," *Knowl. Eng. Rev.*, vol. 27, pp. 1–31, 2012.

[40] J. Foerster, Y. Assael, N. Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Proc. Int. Conf. Inf. Process. Syst.*, 2016, pp. 2145–2153.

[41] P. Marbach and J. Tsitsiklis, "Simulation-based optimization of Markov reward processes," *IEEE Trans. Autom. Control*, vol. 46, no. 2, pp. 191–209, Feb. 2001.

[42] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[43] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, vol. 382. Hoboken, NJ, USA: Wiley, 2007.

[44] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K: Cambridge Univ. Press, 2004.

[45] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *in Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 6382–6393.

[46] B. Peng et al., "FACMAC: Factored multi-agent centralised policy gradients," *in Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 12208–12221.

[47] Z. He, L. Dong, C. Song, and C. Sun, "Multiagent soft actor-critic based hybrid motion planner for mobile robots," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, doi: 10.1109/TNNLS.2022.3172168.

[48] D. Willemsen, M. Coppola, and G. C. de Croon, "MAMBPO: Sample-efficient multi-robot reinforcement learning using learned world models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5635–5640.

Hongliang Guo received the B.E. degree in mechanical engineering and the dual M.E. degree in automation engineering and electrical and computer engineering from the Beijing Institute of Technology, Beijing, China, in 2005 and 2007, respectively, and the Ph.D. degree from the Stevens Institute of Technology, Hoboken, NJ, USA, in 2012.

He was a Postdoctoral Researcher with Almende, Rotterdam, The Netherlands; Nanyang Technological University, Singapore; and Singapore MIT Alliance for Research and Technology. In 2021, he joined the Institute for Infocomm Research, Agency for Science Technology and Research, Singapore, as a Scientist. His research interests include reliable path planning and learning under uncertainties, and multirobot efficient search.

Zhaokai Liu received the bachelor's (Hons.) degree in mechanical engineering from the School of Mechanical Engineering, Sichuan University, Chengdu, China, in 2020. He is currently working toward the master's degree with the School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu.

His research interests include multirobot efficient search, multirobot guaranteed search, and simultaneous localization and mapping for mobile robots.

Rui Shi received the bachelor's degree in automation engineering from the School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu, China in 2019, where he is currently working toward the master's degree majoring in control science and engineering.

His research interests include reliable path planning and decision making under uncertainties with application to autonomous navigation in dynamic environments.

Wei-Yun Yau received the B.Eng. degree in electrical engineering from the National University of Singapore, Singapore, in 1992, and the M.Eng. degree in electrical engineering and Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 1995 and 1999, respectively.

He is currently with the Institute for Infocomm Research, Agency for Science Technology and Research, Singapore, as the Head of the Department of Robotics and Autonomous Systems. Since 2019, he has been a Deputy Director (Technology) of Rehabilitation Research Institute of Singapore. He has authored or coauthored widely, with 13 patents granted and more than 150 publications. His research interests include intelligent robots, biometrics, robotic perception and navigation, and human–robot interaction.

Dr. Yau was the recipient of TEC Innovator Award in 2002, the Tan Kah Kee Young Inventors Award in 2003 (Merit), the IES Prestigious Engineering Achievement Award in 2006, and the Standards Council Distinguished Award in 2007.

Daniela Rus (Fellow, IEEE) received the Ph.D. degree in computer science from Cornell University, Ithaca, NY, USA, in 1993.

She is currently the Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science and the Director of the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA. Her research interests include robotics, mobile computing, and data science.

Dr. Rus is a Class of 2002 MacArthur Fellow, a Fellow of the Association for Computing Machinery and Association for the Advancement of Artificial Intelligence, and a Member of the National Academy of Engineering and the American Academy for Arts and Science.