R-FAC: Resilient Value Function Factorization for Multirobot Efficient Search With Individual Failure Probabilities

Hongliang Guo, Qi Kang, Wei-Yun Yau, *Senior Member, IEEE*, Chee-Meng Chew, *Senior Member, IEEE*, and Daniela Rus, *Fellow, IEEE*

***Abstract*—This article investigates the *resilient* multirobot efficient search problem (R-MuRES), which aims at coordinating multiple robots to detect a "nonadversarial" moving target with the minimal expected time. One unique characteristic of R-MuRES among others is the possibility of individual robot's malfunction and withdrawal from the team during task execution, which results in a *variable* number of searchers in the deployment phase and entails that the possibility of team member failures must be considered during the planning stage, particularly in the training phase. We propose a resilient value function factorization (R-FAC) paradigm, which constructs the central value function from individual ones in a resilient manner, taking into account individual robots' failures, and ensures that the constructed central value function has the minimal mean squared temporal difference error across various team compositions. R-FAC stipulates that the individual global maximum principle is satisfied for whichever team configuration and thus any functioning robot contributes positively to the remaining team, as long as it executes the greedy policy with respect to the factorized individual value function. Subsequently, we introduce the *variational* value decomposition network (V2DN) as one of the instantiated R-FAC algorithms. V2DN employs the log-sum-exp mechanism to construct the central value function from individual ones, enabling it to take a varying number of robots' individual value functions as inputs. Then, we explain why, specifically for the multirobot search task, the log-sum-exp mechanism is superior to the brute-force summation operation used in the canonical value decomposition network (VDN), and compare V2DN with state-of-the-art MuRES solutions as well as the vanilla VDN algorithm in two canonical MuRES testing environments and show that it achieves the best resiliency score when one or several individual robots quit the team during task execution. Furthermore, we validate V2DN with a real multirobot system in a self-constructed indoor environment as the proof of concept.**

Received 24 November 2024; revised 27 March 2025; accepted 17 April 2025. Date of publication 6 May 2025; date of current version 29 May 2025. This work was is supported by A∗STAR under Grant C221518004 and Grant SC36/19-000801-A041. This article was recommended for publication by Associate Editor S. L. Smith and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. *(Hongliang Guo and Qi Kang contributed equally to this work.)*

Hongliang Guo, Qi Kang, and Wei-Yun Yau are with the Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A∗STAR),, Singapore 138632 (e-mail: scott0793@gmail.com).

Chee-Meng Chew is with the National University of Singapore (NUS),, Singapore 119077.

Daniela Rus is with the Computer Science, and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), Cambridge, MA 02139 USA.

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2025.3567478, provided by the authors.

Digital Object Identifier 10.1109/TRO.2025.3567478

***Index Terms*—Individual global maximum (IGM), nonadversarial moving target, resilient value function factorization (R-FAC), resilient multirobot efficient search, variational value decomposition network.**

NOMENCLATURE

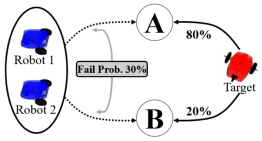
List of Acronyms

CE-PG	Cross entropy regularized policy gradient.
CT	Centralized training.
CTDE	Centralized training and decentralized execution.
DE	Decentralized execution.
Dec-POMDP	Decentralized partially observable Markov decision process.
DQN	Deep Q-network.
DRL-Searcher	Distributional reinforcement learning-based searcher.
E-LOAM	LiDAR odometry and mapping with expanded local structural information.
FHPE	Finite horizon path enumeration.
IGM	Individual global maximum.
MADDPG	Multiagent deep deterministic policy gradient.
MARL	Multiagent reinforcement learning.
MILP	Mixed integer linear programming.
MLP	Multilayer perceptron.
mSAC	Multiagent soft actor critic.
MuRES	Multirobot efficient search.
MuRGS	Multirobot guaranteed search.
MuRRS	Multirobot reliable search.
PD-FAC	Probability density factorization.
R-FAC	Resilient value function factorization.
R-MuRES	Resilient multirobot efficient search.
RS	Resiliency score.
TD	Temporal difference.
V2DN	Variational value decomposition network.
VDN	Value decomposition network.
w.p.	With probability.
w.r.t.	With respect to.

List of Major Notations Used in the Paper

Notations	Descriptions.
N	Number of robots in the predeployment phase.

$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Search environment (represented as a graph).
$\mathcal{V}$	Set of nodes, $\lvert\mathcal{V}\rvert = n$.
$\mathcal{E}$	Set of edges, $\lvert\mathcal{E}\rvert = m$.
e_t	Target's position at time t, $e_t \in \mathcal{V}$.
$\mathbf{\Gamma}$	Target's motion dynamics.
i	Robot index, $i \in \{1, 2, \ldots, N\}$.
$p_t^{(i)}$	Robot i's position at time t.
$\boldsymbol{p}_{\leq t}^{(i)}$	robot i's position sequence up to time t.
$\boldsymbol{s}_t^{(i)}$	Robot i's embedded state vector at time t.
Ψ	Sequence encoding function (compress $\boldsymbol{p}_{\leq t}^{(i)}$ to $\boldsymbol{s}_t^{(i)}$).
β	Time-discounted factor for Ψ.
$Q^{(i)}$	Individual value function of robot i, represented by MLP.
d	Number of hidden nodes of MLP for $Q^{(i)}$.
$\boldsymbol{\theta^{(i)}}$	Parameter vector of the individual value function.
$\pi^{(i)}$	Robot i's policy (ϵ-greedy w.r.t. individual value function)
$\rho^{(i)}$	Robot i's failure probability at any time step.
Q^{tot}	Central value function.
r_t	Team reward at time t.
r_{cap}	Target capture reward.
δ_t	TD error at time t w.r.t. the central value function.
J	Objective of V2DN's centralized training module.
α	Learning rate of V2DN's centralized training module.
η	Resiliency score.
$t_{\max}$	Maximum allowed target capture time.



(a)

	Canonical MuRES Solution	R-MuRES Solution
R1/ R2 Action	A/B or B/A	A/A
Target Capture Prob.	70%	**72.8%**

(b)

Fig. 1. Illustration of an R-MuRES problem. Fig. 1(a) shows a simple multirobot search scenario, where two robots collaborate to search for one target, which has 80% probability of residing in Node A and 20% probability of residing in Node B. The robots can choose Action A (leading to Node A) or Action B (leading to Node B), but both actions have a 30% failure probability, causing the robot to malfunction and withdraw from the team. Fig. 1(b) shows the solution quality comparison between canonical MuRES solution, e.g., model-based algorithms and MARL-based solution, which essentially disperse the robots to Node A and Node B, respectively, and the optimal R-MuRES solution which conglomerates both robots at Node A for larger target capture probability, considering potential robot failures. (a) R-MuRES scenario. (b) Solution quality comparison.

I. Introduction

MURES has emerged as a prominent and interdisciplinary research subject, situated at the confluence of fundamental research and industrial applications. Here, we would like to articulate that the term "MuRES" originates from [1], and it refers to exactly the same moving target search problem as defined in [2], namely, multirobot efficient search path-planning.

From the fundamental research standpoint, MuRES serves as a practical and challenging experimental platform for exploring diverse domains related to artificial intelligence, including MARL, game theory, and multirobot coordination and exploration. In addition, MuRES represents a vital research frontier in numerous industrial applications related to multiple robots, such as search and rescue in hazardous environments [3], collaborative source leakage detection [4], and surveillance and monitoring for regional security [5]. The research challenges in these applications entail effectively coordinating multiple robots to explore an area and detect specific targets and/or anomalies with high precision, accuracy, and efficiency, necessitating innovative multirobot coordinated search methodologies.

As a result, MuRES has garnered increasing attentions from both academic professionals and industrial entrepreneurs over the past several decades. However, despite its popularity, there is a surprising lack of prior research specifically targeting the R-MuRES problem. Specifically, the R-MuRES problem is to coordinate a team of possibly *faulty* robots to search for a non-adversarial moving target with the minimal expected detection time. In R-MuRES, some robots might malfunction and leave the team during task execution, while the remaining multirobot system must continue to function resiliently with a good overall performance.[1] In practice, it is not uncommon for robots to suddenly malfunction during execution, resulting in the inability to fulfill previously allocated tasks. Therefore, designing a resilient MuRES algorithm which maintains up-to-standard performance when facing (potential) individual failures is quite beneficial. Fig. 1(a) presents a simple yet illustrative example highlighting the necessity of developing a dedicated R-MuRES algorithm to solve the R-MuRES problem, as canonical MuRES solutions, including model-based baselines, may not be capable of delivering the optimal solution.

This article proposes the R-FAC paradigm as a guideline for addressing the R-MuRES problem. The R-FAC paradigm stipulates three critical requirements: 1) adherence to the IGM principle, which mandates a positive relationship between the central value function and any individual value function; 2) the use of a *variational* central value function construction mechanism capable of accommodating the varying number of robot inputs; and 3) the minimization of mean squared TD error across various functional team configurations. To instantiate the R-FAC paradigm, we present the V2DN, which leverages the log-sum-exp mechanism to construct the central value function from individual ones, ensuring compliance with the IGM principle and enabling seamless adaptation to varying number of robots. We evaluate and benchmark V2DN's performance in two canonical MuRES testing environments, namely MUSEUM and OFFICE [2], against state-of-the-art MuRES solutions as well as the canonical VDN algorithm, and also validate the applicability of V2DN with a real multirobot system in a self-constructed indoor environment.

This article makes the following contributions to the field of multirobot search: 1) initiation of a new MuRES problem, namely, R-MuRES, which has been surprisingly overlooked

[1]Note that the term "resilient" in the R-MuRES problem is different from the term "reliable." Multirobot *reliable* search refers to a new multirobot search objective, meaning that the moving target needs to be detected "reliably," e.g., with the maximal target detection probability before a given deadline.

in the field of multirobot search; 2) proposal of the R-FAC paradigm, which outlines the critical requirements for an R-MuRES solution, and introduction of V2DN as an instantiated algorithm under the R-FAC paradigm; and 3) design of a resiliency performance metric, namely, RS, to evaluate and compare the performance of V2DN with several state-of-the-art MuRES solutions in two canonical multirobot search testing environments, with the results reported on a publicly accessible leaderboard.

The rest of this article is organized as follows. In Section II, we provide a brief literature review of multirobot search and MARL. Then, we formulate the R-MuRES problem and introduce related background knowledge of value function factorization (VFF) in Section III. Next, in Section IV, we introduce the R-FAC paradigm and its instantiated algorithm, V2DN, and then benchmark its performance in terms of the RS against several state-of-the-art MuRES solutions in Section V. We then deploy V2DN to a real multirobot system and demonstrate its applicability and effectiveness in Section VI. Finally, Section VII concludes this article.

II. Literature Review

Multirobot search involves the coordination of multiple robots to search for one or more moving targets within a given environment, and depending on the search objective, the research field can be roughly categorized into three main branches, namely, MuRES, MuRRS, and MuRGS. MuRES seeks to find a joint search strategy which minimizes the *expected* search time, while MuRRS aims to maximize the reliability metric based on a predefined reliability function over the search effort. In a sense, MuRES can be considered as a special case of MuRRS, where the reliability function is simply the search time itself. MuRGS, on the other hand, coordinates the robots to ensure that all moving targets in the environment are eventually detected, regardless of their motion and sensing characteristics.

In this section, we present a concise overview of the MuRES literature, along the taxonomies of 1) operating environments; 2) target's motion behavior; 3) robot team's sensing characteristics; and 4) canonical MuRES methodologies. For MuRRS and MuRGS related literature, readers may refer to [6] and [7], respectively. Moreover, we briefly review the recent trends in MARL toward the end of this section as the R-FAC paradigm is essentially targeting the MuRES problem from MARL's perspective.

A. Multirobot Efficient Search: General Introduction

Various perspectives can be used to partition the MuRES literature, and this subsection describes it along the taxonomies of the 1) operating environments; 2) target's motion behavior; and 3) robot team's sensing characteristics.

From the operating environment's perspective, the MuRES literature can be divided into continuous environments and discrete ones. The continuous environments can be bounded and constrained by polygons or circles [8], [9], [10], or unbounded and unstructured with stationary obstacles [3], [11]. On the other hand, discrete environments are described by occupancy grid maps [12], [13] or topological graphs [1], [2], [14], [15].

The target's motion behavior also plays a significant role in designing corresponding multirobot search strategies. In the MuRES literature, the target may remain stationary during the search process [12], [16], [17], move in a nonadversarial manner that is independent of the robot team's search strategy [1], [18], [19], or move adversarially in an attempt to evade detection [9], [10], [20].

Finally, the robot team's sensor characteristics, i.e., sensor range, sensor accuracy, are also essential factors for categorizing the MuRES literature. In continuous environments, the robot's sensor range can be circular [10], [21] or line-of-sight [22], [23], detecting the target within a certain distance from the sensor or along an unblocked straight line between the sensor and target. In discrete environments, the sensor's detection range can be the same-node detection [1], [2], [24] or arbitrary range detection [15], [25]. In addition, sensors can be perfect or probabilistic, with perfect sensors always accurately detecting the target within their range [2], while probabilistic sensors have a probability of false negative and/or false positive detections [1], [13], [15], [26].

One closely related work to the multirobot resilient search problem investigated in this article is the risk-sensitive multirobot search studied in [27]. Shree et al. [27] proposed a *risk-averse* multirobot search algorithm for hazardous environments, where the robot team plans the search paths for moving targets (e.g., victims) while actively avoiding dangerous locations based on real-time observations. In contrast, our work focuses on the multirobot *resilient* search problem, where robot failures occur probabilistically and cannot be entirely avoided. The key challenge here is to develop planning strategies that ensure effective search performance despite potential individual failures.

B. Multirobot Efficient Search: Methodologies

The last section dissects the MuRES literature along various taxonomies, and in this section, we focus on describing the prevailing MuRES methodologies. To date, researchers have developed many algorithms as MuRES solutions, which can be roughly categorized into four main groups, namely, 1) model-based algorithms, 2) swarm intelligence, 3) MARL, and (4) game theory.

Model-based algorithms have been widely deemed as the most canonical methods to solve the MuRES problem. Algorithms within this category formulate the MuRES problem into mathematical models, and utilize existing optimization solvers to find the optimal solution [2], [14], [15], [28]. For instance, Asfora et al. presume the availability of the target's motion dynamics, its initial distribution, and the robot team's motion and sensor characteristics, and then establish a set of mathematical equations describing the MuRES objective and related constraints. In this way, MuRES is formulated as a MILP problem, and the canonical optimization solver, i.e., Complex Linear Programming Expert (CPLEX), is employed for the optimal solution [15]. However, the method requires explicit assumptions regarding the target's behavior and the robot team's

motion and sensor characteristics, which may not be entirely accurate in practice. In addition, the MILP formulation, which is essentially a centralized approach, may suffer from scalability issues when the size of the problem or the number of robots increases. Nonetheless, model-based algorithms remain a powerful and widely used approach for solving the MuRES problem, particularly in scenarios where the environment is well-understood and the assumptions made in the modeling process are reasonable.

Swarm intelligence serves as the most intuitive and straightforward approach for addressing the MuRES problem. Algorithms within this group involve designing various agent-based mechanisms as the individual behavior guideline. As each robot performs its own dynamics, the robot team, as a whole, is exhibiting a certain group-level behavior for the efficient target search task [29], [30], [31], [32]. For example, Li and Tan [31] designed a three-state finite state machine, and the individual robots switch among the three states, i.e., search, diffuse and target tracking, probabilistically, to achieve the overall emerging group behavior for collective target search. Generally speaking, swarm intelligence, which guides the individual robot's behavior, is a useful approach for collective search with advantages of being scalable and robust against various uncertainties. However, current swarm intelligence methods lack a clear relationship between individual behavior and the overall MuRES objective, which makes it challenging to optimize performance through individual mechanism tuning or redesign, limiting the optimization process to predesigned local rules with uncertain global outcomes.

Learning-based approaches have recently emerged as a promising solution to the MuRES problem, as evidenced in [1], [10], [18], and [26]. Algorithms within this category treat MuRES as a multiagent sequential decision-making problem and typically cast it into the Dec-POMDP framework. Several decentralized policy optimization algorithms are proposed, such as MADDPG [26], CE-PG [1], PD-FAC [7]. However, it is important to note that current learning-based algorithms assume that all robots are functioning properly during the search process. In reality, it is not uncommon that some robots might malfunction and leave the team during the task execution. Therefore, this article aims at addressing the issue by proposing a resilient MARL solution which accounts for the possibility of sudden team member failures. Here, we would like to state the respective advantages of model-based algorithms and learning-based algorithms for the multirobot search problem. Model-based methods necessitate a well-defined, explicit model of the target's motion dynamics for planning. In contrast, when only target motion samples are available (without an explicit analytical model), learning-based methods provide a viable alternative. This distinction helps determine the applicability of each approach based on the availability of target motion dynamics.

Finally, game theoretical approaches provide a unique set of tools for addressing the MuRES problem with an "adversarially" moving target. The fact that the target is moving adversarially, i.e., avoids detection by the robots, presents a unique challenge to all three categories of MuRES solutions described earlier, in that the target may adapt to any preplanned multirobot search strategy, thereby decreasing the system's performance. Algorithms within this category typically establish the MuRES problem within the zero-sum game framework and propose various (Nash) equilibrium-seeking algorithms [23], [33], [34] or set constraints on the capabilities of the searching team of robots that are required to succeed in the search task based on the properties of the targets, the environment in which the targets are located and the sensing modalities of the search team [35], as solutions.

C. Multiagent Reinforcement Learning

Since the R-FAC paradigm is essentially targeting the MuRES problem from MARL's perspective, and the subsequently instantiated V2DN is a new cooperative MARL algorithm for the resilient MuRES problem with individual robot failures, in this section, we deliver a brief literature review of cooperative MARL for necessary contexts of R-FAC and V2DN. Interested audiences are referred to [36], [37] for the comprehensive literature review.

As a subfield of reinforcement learning, cooperative MARL focuses on multiple learning agents which interact and *collaborate* with each other in a shared environment. The field has gained increasing attention in recent years, with the development of novel algorithms and successful applications in various domains, such as multirobot cooperation, multivehicle navigation. Prevailing MARL algorithms typically formulate the underlying mathematical problem within the Dec-POMDP framework. They either decompose the system-level central value function into individual ones with the VFF methods while adhering to the IGM principle [38], [39], [40], [41], [42], or directly calculate the (approximated) decentralized policy gradient for each individual agent [43], [44], [45]. For example, Sunehag et al. [39] proposed VDN, which represents the central value function as the summation of individual ones and uses back-propagation to update the parameters of the individual value functions with the goal of minimizing the average TD error of the reconstructed central value function.

However, despite the success of cooperative MARL algorithms in various domains, little attention has been paid to the resiliency aspect of MARL, where some agents within the multiagent system might malfunction and quit the team during task execution, without notifying others. In this article, we focus on resilient MARL with its application to the R-MuRES problem, by proposing 1) the R-FAC paradigm, which outlines the critical requirements for a resilient MARL algorithm, and 2) V2DN as an instantiated R-FAC algorithm for the R-MuRES problem.

III. Problem Formulation and Backgrounds

In this section, we first present R-MuRES's problem setup along the aspects of 1) the operating environment, 2) the target's motion model, 3) the robots' sensing and motion model, and then establish R-MuRES within the Dec-POMDP framework. Subsequently, we introduce the canonical MARL algorithm, namely, VDN, under the CTDE scheme. A list of major notations used throughout this article is presented in the Nomenclature, and we will also deliver brief explanations when they first appear in the main manuscript.

A. R-MuRES's Problem Setup

The problem that we are studying is to coordinate a team of N (possibly faulty) robots to detect and thereby capture one nonadversarial moving target in a discrete environment.

The environment is represented by an undirected and connected "unit cost" graph, $\mathcal{G}(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ ($|\mathcal{N}| = n$) refers to the set of nodes, and $\mathcal{E}$ ($|\mathcal{E}| = m$) represents the set of edges. The term "unit cost" means that the transition time of executing any edge in $\mathcal{G}$ is exactly one time unit. Note that the "unit-cost" assumption is a common one in the MuRES literature for discrete environments, see [1], [7], [15], [26] as examples. In practice, one may simply transform the nonunit-cost topological graph into a unit-cost one by breaking the "long" edges into several connected "unit-cost" subedges.

The target, whose position at time step t is denoted as e_t, moves nonadversarially, i.e., the target's motion is *independent* of the robot team's search strategy. In this article, we represent the target's motion dynamics as a stochastic transition matrix $\mathbf{\Gamma}$, which describes the target's motion model, e.g., $\mathsf{P}[e_{t+1}|e_t] = \mathbf{\Gamma}(e_t, e_{t+1})$. Note that the target's motion model ($\mathbf{\Gamma}$) needs to respect $\mathcal{G}$, i.e., $(e_t, e_{t+1}) \in \mathcal{E}$ or $e_{t+1} = e_t$, and is *unknown* to the robot team.

The robots within the team possess perfect sensors, with the same-node detection ability. For example, when robot i, whose position at time step t is denoted as $p_t^{(i)}$, resides in the same node as the target does, i.e., $p_t^{(i)} = e_t$, the target is detected by robot i, and the R-MuRES task is completed. Denote robot i's decision-making policy as $\pi^{(i)}$, which takes as inputs an ever-growing position sequence, e.g., at time step t, $\pi^{(i)} = \pi^{(i)}\left(\boldsymbol{p}_{\leq t}^{(i)}\right)$, where $\boldsymbol{p}_{\leq t}^{(i)} = \left(p_0^{(i)}, p_1^{(i)}, \ldots, p_t^{(i)}\right)^\top$ indicates robot i's position sequence from time 0 to time t. The decision-making policy ($\pi^{(i)}$) dictates the next step position of robot i, i.e., $p_{t+1}^{(i)} = \pi^{(i)}\left(\boldsymbol{p}_{\leq t}^{(i)}\right)$. Similar to the target's motion model ($\mathbf{\Gamma}$), the decision-making policy must respect $\mathcal{G}$ as well. One unique characteristic of the R-MuRES problem is that during task execution, some of the robots might malfunction and quit the team without notifying others. For example, for robot i, at any time step t, there is a certain probability ($\rho^{(i)}$) that it would malfunction and quit the team, and thus cannot execute $\pi_t^{(i)}$ anymore, without notifying others. The objective of R-MuRES is to reach a resilient and coordinated joint policy $\boldsymbol{\pi} = \left(\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(N)}\right)^\top$, with minimal expected time to detect the nonadversarially moving target despite random individual robot failures during task execution.

As the R-FAC paradigm and the subsequently instantiated V2DN algorithm are treating the R-MuRES problem from MARL's perspective, we establish the R-MuRES problem within the Dec-POMDP framework. A Dec-POMDP can be described as $M = \langle I, S, \{A_i\}, \mathsf{P}, R, \{\Omega_i\}, O, \gamma \rangle$, where I refers to the set of agents, S indicates the set of states, A_i refers to the set of actions for agent i, R is the team reward, Ω_i is the set of observations for agent i, P and O are state transition probabilities and observation probabilities, respectively, γ is the discount factor [46]. In the context of R-MuRES, robot i's current position $p_t^{(i)}$ corresponds to its observation Ω_i, and the executing edge refers to action A_i. The discount factor $\gamma = 1$; the instantiated team reward r_t is set to be 1 at each normal time step and $r_t = r_{\text{cap}}$ if the moving target is captured by any functioning robot, and the whole search process terminates.[2] The objective is to find the joint policy $\boldsymbol{\pi}$, which maximizes the cumulative team reward. Note that one unique characteristic of the established Dec-POMDP is that the set of agents (I) is not constant as the search process evolves, in that some agents might malfunction and withdraw from the team during task execution.

B. Value Decomposition Network

As the R-FAC paradigm is essentially treating the R-MuRES problem from cooperative MARL's perspective, and the instantiated V2DN algorithm under the R-FAC paradigm evolves from the canonical VDN algorithm, we describe the mechanisms of VDN in this section.

VDN is deemed as one of the prevalent cooperative MARL methods along the direction of VFF [39]. The main assumption that VDN makes is that the joint central action-value function can be additively factorized into individual value functions across agents, i.e.,

$$Q^{\text{tot}}(\boldsymbol{s}, \boldsymbol{a}) = \sum_{i=1}^{N} Q^{(i)}\left(s^{(i)}, a^{(i)}; \boldsymbol{\theta^{(i)}}\right) \tag{1}$$

where N is the total number of agents in the system, Q^{tot} is the joint central value function, and $Q^{(i)}$ indicates the individual value function. $\boldsymbol{s}$ and $\boldsymbol{a}$ indicate the joint state and joint action of the system, respectively, while $s^{(i)}$ and $a^{(i)}$ represent the individual state and action, respectively. $\boldsymbol{\theta^{(i)}}$ refers to the parameter vector of the individual value function.

VDN follows the CTDE scheme [43], in which the training phase is performed in a centralized and offline manner and the execution phase is implemented in a decentralized and online way. With the team reward at time step t represented as r_t, we can get the TD error w.r.t. Q^{tot} at time step t as

$$\delta_t = r_{t+1} + \gamma Q_{t+1}^{\text{tot}} - Q_t^{\text{tot}} \tag{2}$$

where δ_t denotes the TD error w.r.t. the central value function at time step t. The training objective is to find the optimal values for $\boldsymbol{\theta^{(i)}}$, which minimize the system-level joint TD error across all agents. This ensures that the learned value functions align closely with the system's overall objectives. During the online decentralized execution (decentralized execution) stage, each agent independently makes decisions using its individual value function. Actions are selected following the ϵ-greedy strategy [47], which balances exploration of new actions and exploitation of known optimal actions to maximize long-term rewards.

IV. R-FAC Paradigm and V2DN

This section presents R-FAC as the MARL's solution paradigm to the R-MuRES problem, and then introduces V2DN

[2] Here, r_t is an instantiation of the random variable R.

as one of the instantiated algorithms under the R-FAC paradigm. We analyze that V2DN adheres to the R-FAC paradigm, and presents its pseudocode with the corresponding computational complexity analysis.

A. R-FAC Paradigm

The R-MuRES problem is characterized by the potential occurrence of individual robots' malfunction and withdrawal from the team during task execution. Therefore, it is imperative to design dedicated methodologies that exhibit resilience against such failures. In this context, the R-FAC paradigm sets forth three essential requirements that a resilient methodology for the R-MuRES problem must fulfill. Before delving into these requirements, we introduce the following definitions.

Definition 1 (Variational function): A function is said to be *variational* if it is capable of accepting a *varying* number of input variables. One example of a variational function is the $\max$ operator, which calculates the maximum value of a vector with potentially varying length.

Definition 2 (Central value function): The central value function, denoted as Q^{tot}, represents the expected total reward accumulated by the entire robot team.

Definition 3 (Individual value function): The individual value function, denoted as $Q^{(i)}$ ($i \in \{1, 2, \ldots, N\}$), serves as the key ingredient to construct the central value function. It is a utility function specific to each individual robot, taking as inputs the robot's local observations and actions.

Note that the individual value function, e.g., $Q^{(i)}$, does not indicate the individual robot's expected cumulative reward. Rather, it is merely a utility function for constructing the central value function. In this paper, we denote f as the construction function which recovers the central value function from individual ones, i.e., $Q^{\text{tot}} = f(Q^{(1)}, Q^{(2)}, \ldots, Q^{(N)})$. With the three definitions, we present the R-FAC paradigm, which specifies the following three essential requirements for an algorithm to qualify as an R-MuRES solution:

1) *the central value function construction function (f) must be* variational;
2) *the partial derivative of Q^{tot} w.r.t. $Q^{(i)}$ must exist and is greater than or equal to zero, i.e., $\forall i \in \{1, 2, \ldots, N\}$, $\partial Q^{tot} / \partial Q^{(i)} \geq 0$;*
3) *the mean squared TD error w.r.t. Q^{tot} is minimized.*

The first requirement of the R-FAC paradigm is to ensure that the central value function construction process is capable of accommodating a varying number of inputs, i.e., a varying number of individual value functions; the second requirement mandates that the IGM principle [38] is satisfied for whatever team configurations, and hence any functioning individual robot contributes to the team positively, and the third requirement dictates the objective of the instantiated algorithm under the R-FAC paradigm as the minimization of mean squared TD error w.r.t. the central value function.

It is straightforward to verify that the vanilla VDN algorithm satisfies all the three requirements dictated by the R-FAC paradigm, and hence it can be deemed as one of the instantiated R-FAC algorithms for the R-MuRES problem. Here, it is worth noting that the padding technique in computer vision is usually adopted when the convolution kernel "meets" the corner of the image, with fewer inputs. The "same padding" technique, which adds padding around the input so that the output feature map has the same height and width as the input, is usually applied. Typically, zero values are used for padding, which is then deemed as zero padding. In the multirobot search context, the "same padding" technique can be treated as one type of the "variational" function, by simply treating the nonfunctional robots' Q-values as zeros. In this case, it is essentially the canonical VDN algorithm.

However, the brute-force summation operator within VDN is not well suited for the multirobot search problem, and we will use an illustrative example to highlight the limitations of vanilla VDN, i.e., resulting in the large mean squared TD error, in the multirobot search context in Section V-A. In the next section, we introduce the V2DN method, which employs the $\log$-sum-$\exp$ operation as the central value function construction function, and serves as an improvement over the vanilla VDN algorithm for the R-MuRES problem.

B. Variational Value Decomposition Network

The last section presents the R-FAC paradigm, which outlines three essential requirements for the instantiated algorithm to qualify as a resilient solution to the R-MuRES problem, and points out that VDN satisfies all requirements and thus can be deemed as one of the instantiated R-FAC algorithms. In this section, we introduce the V2DN under the Dec-POMDP framework, as another instantiated R-FAC algorithm for the R-MuRES problem. V2DN follows the CTDE scheme. During the DE stage, each robot simply follows the ϵ-greedy policy w.r.t. its own individual value function ($Q^{(i)}$). On the other hand, during the CT stage, all individual value functions are aggregated together to construct the central value function (Q^{tot}), and the related parameters are optimized to minimize the mean squared TD error w.r.t. the constructed central value function. Fig. 2 depicts V2DN's framework including both the CT module and DE module, as well as the abstract algorithm flow process.

1) Centralized Training for V2DN: There are two main challenges in designing V2DN's CT module. First, the individual value function, denoted as $Q^{(i)}$, takes as input an ever-growing sequence of positions $\boldsymbol{p}^{(i)}_{\leq t}$. The continuously expanding input poses significant difficulties in the parameterization process, requiring efficient methods to handle the dynamic nature of the sequence while maintaining accurate function estimation. Second, the construction of the central value function, Q^{tot}, must satisfy two crucial requirements: it needs to be variational, meaning it can adapt dynamically to varying inputs, and it needs to remain positively correlated with each individual value function $Q^{(i)}$. Meeting these requirements while ensuring seamless integration across agents adds substantial complexity to the design process.

In this article, we employ the sequence encoding function (Ψ) to dynamically compress the position sequence that grows with

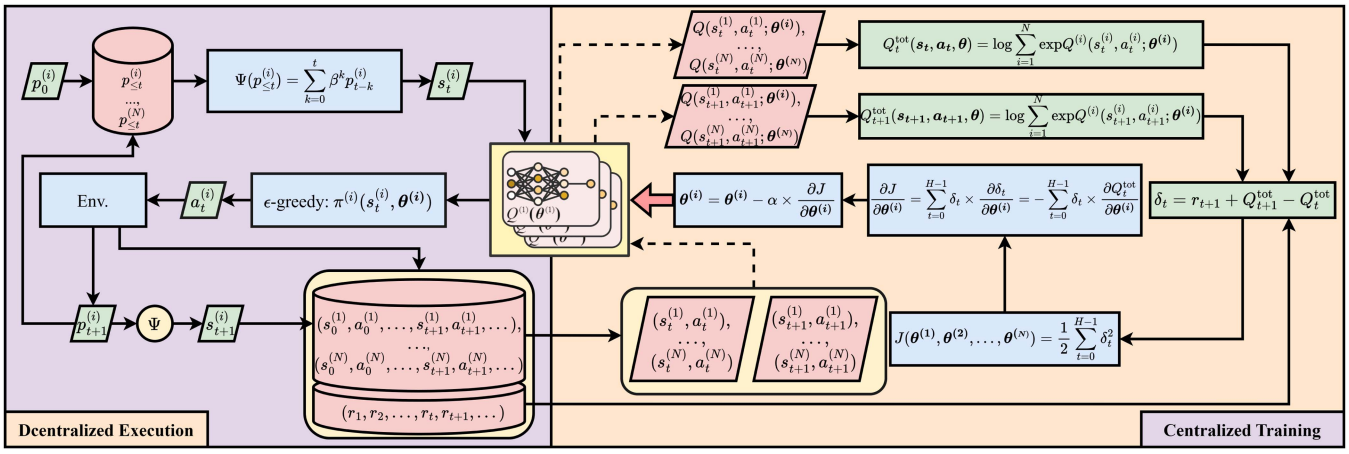


Fig. 2. V2DN's CTDE framework and the abstract algorithm flow process. For the DE module, each robot, e.g., robot i, starts at position $p_0^{(i)}$; calls the sequence encoding function Ψ for state $s_t^{(i)}$; calculates the individual value function $Q^{(i)}$, and decides action $a_t^{(i)}$ with ϵ-greedy policy. The robot executes $a_t^{(i)}$, reaches $p_{t+1}^{(i)}$, and goes to a new round of execution process while saving related data into database. The CT module collects all robots' trajectories and team reward sequence, and updates each individual value function's parameter vector ($\boldsymbol{\theta^{(i)}}$), with the objective of minimizing the mean squared TD error w.r.t. Q^{tot}.

time t into a fixed length feature vector, i.e.,

$$\boldsymbol{s}_t^{(i)} = \Psi\left(\boldsymbol{p}_{\le t}^{(i)}\right) \tag{3}$$

where $\boldsymbol{s}_t^{(i)}$ is the embedded fixed-length state feature vector, and Ψ is the sequence encoding function. In V2DN, we simply use the time-discounted encoding scheme [48] for state feature embedding.[3] Given a position sequence, e.g., $\boldsymbol{p}_{\le t}^{(i)}$, and each position element has been represented with the one-hot embedding scheme, the discounted encoding function embeds the feature vector as follows:

$$\Psi(\boldsymbol{p}_{\le t}^{(i)}) = \sum_{k=0}^{t} \beta^k \boldsymbol{p}_{t-k}^{(i)} \tag{4}$$

where $0 < \beta < 1$ is the time-discounted factor, and $p_{t-k}^{(i)} \in \mathcal{R}^n$ is robot i's position at time step $(t-k)$. With the embedded state vector $\boldsymbol{s}_t^{(i)}$, we represent the individual value function $Q^{(i)}$ as a MLP and denote the related parameters as $\boldsymbol{\theta^{(i)}} \in \mathcal{R}^{k_l}$, where k_l refers to the total number of parameters of the MLP. In this way, we abstractly represent the parameterized individual value function at time step t as $Q_t^{(i)}(\boldsymbol{s}_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta^{(i)}})$ or $Q_t^{(i)}$ for succinct representation.

In V2DN, the central value function, Q^{tot} is constructed from the individual value functions with the log-sum-exp operator

$$Q_t^{\text{tot}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}) = \log \sum_i \exp Q^{(i)}\left(\boldsymbol{s}_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta^{(i)}}\right) \tag{5}$$

where $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ are the system-level joint (embedded) state feature vector and joint action at time step t, respectively, and $\boldsymbol{\theta}$ is the joint parameter vector. Note that (1) $\boldsymbol{s}_t$, $\boldsymbol{a}_t$ and $\boldsymbol{\theta}$ are merely for symbolic representation convenience, and the actual calculation of Q^{tot} is relayed to the right-hand side of (5); and (2) the summation operation in (5) is to aggregate all the remaining functional robots' individual value functions, and the total number is not necessarily equal to N, which is the number of robots at the predeployment phase. Denoting the TD error at time step t as δ_t, we have

$$\delta_t = r_{t+1} + \gamma Q_{t+1}^{\text{tot}} - Q_t^{\text{tot}}. \tag{6}$$

Note that $\gamma = 1$ and thus it can be omitted in (6). The objective of V2DN's CT module is to minimize

$$J\left(\boldsymbol{\theta^{(1)}}, \boldsymbol{\theta^{(2)}}, \ldots, \boldsymbol{\theta^{(N)}}\right) = \frac{1}{2} \sum_{t=0}^{H-1} \delta_t^2 \tag{7}$$

where H is the length of the collected multirobot search trajectory. One may use the chain rule of multivariate calculus to derive $\partial J / \partial \boldsymbol{\theta^{(i)}}$ as follows:

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{\theta^{(i)}}} &= \sum_{t=0}^{H-1} \delta_t \times \frac{\partial \delta_t}{\partial \boldsymbol{\theta^{(i)}}} \\ &= -\sum_{t=0}^{H-1} \delta_t \times \frac{\partial Q_t^{\text{tot}}}{\partial \boldsymbol{\theta^{(i)}}} \\ &= -\sum_{t=0}^{H-1} \delta_t \times \frac{\exp Q_t^{(i)}}{\sum_i \exp Q_t^{(i)}} \times \frac{\partial Q_t^{(i)}}{\partial \boldsymbol{\theta^{(i)}}} \end{aligned} \tag{8}$$

where $\partial Q_t^{(i)} / \partial \boldsymbol{\theta^{(i)}}$ can be derived with MLP's backpropagation mechanism. Note that since the CT module's training objective is to minimize the mean squared TD error rather than the residual error w.r.t. the central value function, we treat the parameters of Q_{t+1}^{tot} as constant variables during the derivative derivation process, and only focus on calculating the derivative w.r.t. Q_t^{tot}. This is the reason that Line 2 of (8) has only one partial derivative term. Treating a part of parameters as constant variables is a common scheme used in TD-related learning algorithms, such as DQN [51].

[3] Note that one may use more complex schemes such as gated recurrent unit (GRU) [49] or transformer [50], for the state feature embedding.

With $\partial J/\partial \boldsymbol{\theta}^{(i)}$, one may update $\boldsymbol{\theta}^{(i)}$ with stochastic gradient descent, i.e.,

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i)} - \alpha \times \frac{\partial J}{\partial \boldsymbol{\theta}^{(i)}} \tag{9}$$

where α is the learning rate.

2) Decentralized Execution for V2DN: With the calculated parameters for the individual value function, each robot can update its decision-making strategy (policy) and execute it by employing the ϵ-greedy policy framework [47]. This policy balances exploration and exploitation. For instance, when robot i encounters a specific (compressed) state denoted as $\boldsymbol{s}^{(i)}$, it evaluates its current set of possible actions. The robot follows the ϵ-greedy approach to select an action a based on the following rules.

1) *Exploitation*: With a probability of $1-\epsilon$, the robot selects the action that has the highest predicted value, as determined by its learned value function $Q^{(i)}$. Mathematically, this is expressed as

$$a = \arg\max_{a} Q^{(i)}\left(\boldsymbol{s}^{(i)}, a; \boldsymbol{\theta}^{(i)}\right) \tag{10}$$

where $Q^{(i)}$ is the value function for robot i, parameterized by $\boldsymbol{\theta}^{(i)}$.

2) *Exploration*: With a probability of ϵ, the robot selects an action randomly from the set of all available actions at the given state $\boldsymbol{s}^{(i)}$. Each alternative action is chosen with equal likelihood, computed as: $\epsilon/|A(\boldsymbol{s}^{(i)})|$, where $|A(\boldsymbol{s}^{(i)})|$ represents the total number of available actions at state $\boldsymbol{s}^{(i)}$.

This approach ensures that the robot occasionally explores other actions to discover potentially better strategies while predominantly exploiting its current knowledge to maximize the accumulated reward.

C. Analysis of V2DN

The last section introduces V2DN, which comprises the CT module and the DE module. In the CT module, we first utilize the time-discounted embedding scheme [48] to compress each individual robot's ever-growing position sequence into a fixed length feature vector, and then employ MLP to represent the corresponding individual value function. The central value function is constructed through the log-sum-exp operator, and the MLP parameters are updated through stochastic gradient descent to minimize the mean squared TD error w.r.t. the central value function. In the DE module, each robot simply executes the ϵ-greedy strategy w.r.t. the pretrained individual value function. In this section, we first validate that V2DN adheres to the R-FAC paradigm, and then displays the pseudocode of V2DN's centralized training and decentralized execution process, followed by the corresponding computational complexity analysis.

1) V2DN's R-FAC Paradigm Validation: We validate that V2DN conforms to the R-FAC paradigm by checking the three stipulated requirements in Section IV-A. First, it is straightforward to verify that the central value function construction function, i.e., log-sum-exp, is *variational* and thus capable of accommodating to a varying number of inputs. Second, we prove that the partial derivative of Q^{tot} w.r.t. $Q^{(i)}$ is greater than or equal to zero by introducing and proving the following theorem.

Theorem 1 (Variational IGM theorem): The constructed central value function has a strict positive relationship w.r.t. the individual value function, regardless of the remaining functional team composure, i.e., $\forall i \in \{1, 2, \ldots, N\}$, we have $\partial Q^{\text{tot}}/\partial Q^{(i)} > 0$.

Proof: Calculating the derivative of Q^{tot} w.r.t. $Q^{(i)}$ based on (5), one can get

$$\frac{\partial Q^{\text{tot}}}{\partial Q^{(i)}} = \frac{\exp Q^{(i)}}{\sum_i \exp Q^{(i)}} > 0. \tag{11}$$

From (11), we can see that for whatever team configuration, the partial derivative of Q^{tot} w.r.t. $Q^{(i)}$ is always strictly greater than zero, and in the extreme case of only one remaining functional robot, which leads to $Q^{\text{tot}} = Q^{(i)}$, we still have $\partial Q^{\text{tot}}/\partial Q^{(i)} = 1$, which still satisfies the positive relationship requirement dictated by the R-FAC paradigm. □

Finally, V2DN's parameter update scheme is to minimize the mean squared TD error w.r.t. the constructed central value function, which satisfies the third stipulated requirement by the R-FAC paradigm. Therefore, V2DN qualifies as an instantiated algorithm under the R-FAC paradigm for the R-MuRES solution.

2) Pseudocode and Computational Complexity Analysis: Armed with the understandings of V2DN's variational construction function for Q^{tot}, and the MLP's parameter update objective, we present the pseudocode of V2DN's centralized training process in Algorithm 1, and put the pseudocode for V2DN's decentralized execution process in the publicly available github website for paper's overall flow smoothness.

With the understanding of V2DN's variational construction function for Q^{tot} and the MLP's parameter update objective, we present the pseudocode for V2DN's centralized training process in Algorithm 1. For better readability and maintaining the overall flow of the paper, the pseudocode for V2DN's decentralized execution process is provided in the publicly available github website. Note that 1) Algorithm 1 takes, as inputs, the precollected robot team's trajectory data and the team reward sequence from the decentralized execution process of V2DN; 2) Algorithm 1 depicts V2DN's centralized training process for only one epoch, and the DE module will then collect a new round of robots' trajectory information and team reward sequence with the updated parameterized individual value functions for a new centralized training epoch.

Next, we use the big $\mathcal{O}$ notation to analyze the computational complexity of V2DN's centralized training process. Note that V2DN's decentralized execution process is to simply let each robot follow the ϵ-greedy strategy w.r.t. the pretrained individual value function, and thus we skip the corresponding computational complexity analysis.

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 [52]. To evaluate an algorithm's computational complexity, we count the total

Algorithm 1: V2DN's Centralized Training Process.

Input: Number of robots N; Learning rate α; Time-discounted factor β for Ψ; Pre-collected robot team's trajectory data, *e.g.*, $(p_0^{(i)}, a_0^{(i)}, p_1^{(i)}, a_1^{(i)} \ldots, p_{H_i}^{(i)})$ and team reward sequence $(r_1, r_2, \ldots, r_H)$, with each robot following ϵ-greedy policy w.r.t. $Q^{(i)}(\boldsymbol{\theta^{(i)}})$;

Output: Updated parameter $\boldsymbol{\theta^{(i)}}$ $(i \in \{1, 2, \ldots, N\})$;

Init: $t \leftarrow 0$; $\partial J / \partial \boldsymbol{\theta^{(i)}} \leftarrow 0$; $H \leftarrow \max\{H_i\}$; $\boldsymbol{p}_{\leq t}^{(i)} \leftarrow p_0^{(i)}$;

1 **while** $t \leq H - 1$ **do**
2 **foreach** $i \in \{1, 2, \ldots, N\}$ **do**
3 **if** $t \leq H_i$ **then**
  `/* Robot is functional at t */`
4 $\boldsymbol{p}_{\leq t+1}^{(i)} = \boldsymbol{p}_{\leq t}^{(i)} \cup p_{t+1}^{(i)}$;
5 Calculate $Q_t^{(i)}$ and $Q_{t+1}^{(i)}$ with MLP, *i.e.*, $Q_t^{(i)} = Q^{(i)}\big(\Psi(\boldsymbol{p}_{\leq t}^{(i)}), a_t^{(i)}; \boldsymbol{\theta^{(i)}}\big)$; $Q_{t+1}^{(i)} = Q^{(i)}\big(\Psi(\boldsymbol{p}_{\leq t+1}^{(i)}), a_{t+1}^{(i)}; \boldsymbol{\theta^{(i)}}\big)$;
6 Calculate Q_t^{tot} and Q_{t+1}^{tot} with Eq. (5);
7 Calculate TD-error: $\delta_t = r_t + Q_{t+1}^{\text{tot}} - Q_t^{\text{tot}}$;
8 Store δ_t and $Q_t^{(i)}$ into database;
9 $t \leftarrow t + 1$;
10 **foreach** $i \in \{1, 2, \ldots, N\}$ **do**
11 Calculate $\partial J / \partial \boldsymbol{\theta^{(i)}}$ with Eq. (8), *i.e.*, $\frac{\partial J}{\partial \boldsymbol{\theta^{(i)}}} = -\sum_{t=0}^{H_i - 1} \delta_t \times \frac{\exp Q_t^{(i)}}{\sum_i \exp Q_t^{(i)}} \times \frac{\partial Q_t^{(i)}}{\partial \boldsymbol{\theta^{(i)}}}$;
12 Update $\boldsymbol{\theta^{(i)}}$ with Eq. (9), *i.e.*, $\boldsymbol{\theta^{(i)}} \leftarrow \boldsymbol{\theta^{(i)}} - \alpha \times \frac{\partial J}{\partial \boldsymbol{\theta^{(i)}}}$;
13 Final.

number of flops; express it as a function (usually a polynomial) of the dimensions of involved matrices and vectors, and simplify the expression by ignoring all terms except for the leading ones. Here, we wish to note that the use of flops to analyze an algorithm's computational complexity is unconventional according to [53], and we treat it as an informal discussion of the algorithm's operational cost.

Examining Algorithm 1, we find that the core computational load happens between Lines 2 and 12, which consists of two loops. The first loop contains a nested loop from Lines 3 to 5. Line 4 involves joining two n-dimensional vectors resulting in a computational complexity of $\mathcal{O}(n)$, and Line 5 calculates $Q_t^{(i)}$ and $Q_{t+1}^{(i)}$ by embedding the ever-growing position sequence and utilizing an MLP to calculate the related value. The embedding function has a worst-case computational complexity of $\mathcal{O}(Hn)$. The MLP is configured as a $2n \times h \times 1$ network, where n is the number of nodes in $\mathcal{G}$, and d is the number of hidden nodes in the MLP. In this case, calculating $Q^{(i)}$ has a computational complexity of $\mathcal{O}(2nd)$. Lines 3 to 5 loop for N times for the nested loop, resulting in a computational complexity of $\mathcal{O}(N \times (n + 2(Hn + 2nd)))$, and we can express it as $\mathcal{O}(Nn(H + d))$ when dropping the nonleading terms. Similar analysis can be applied to Lines 6 and 7, which yield the computational complexity at $\mathcal{O}(2N)$ and $\mathcal{O}(3)$, respectively. Therefore, the first loop in Algorithm 1 has a computational complexity of $\mathcal{O}(H(2N + 3 + Nn(H + d))$, and can be represented as $\mathcal{O}(HNn(H + d))$.

The second loop (Line 11 to Line 12) involves the back-propagation of the MLP, i.e., calculating $\partial Q_t^{(i)} / \partial \boldsymbol{\theta^{(i)}}$, which results in a computational complexity of $\mathcal{O}(2nd)$. Based on this, we calculate the computational complexity for Lines 11 and 12 as $\mathcal{O}(2ndNH)$ and $\mathcal{O}(2n)$, respectively. The second loop runs for N times, and thus has a computational complexity of $\mathcal{O}((2ndNH + 2n) \times N)$, which can be simplified as $\mathcal{O}(ndN^2H)$. Therefore, the total computational complexity of the CT module for one epoch is $\mathcal{O}(HNn(H + d) + ndN^2H)$, which can be expressed as $\mathcal{O}(H^2Nn + HN^2dn)$. Therefore, we conclude that the computational complexity of V2DN's centralized training module is polynomial w.r.t. the related parameters. Here, N represents the total number of robots; H is the maximum trajectory length of all robots (accounting for potential robot failures that may shorten certain trajectories); n is the number of nodes in the environment graph $\mathcal{G}$; and d is the number of hidden nodes in the neural network for Q-value approximation.

D. V2DN's Application to Multitarget Search

So far, we have presented the V2DN algorithm as one solution to the multirobot resilient search problem. In practice, detecting only one target is quite limiting. In this section, we discuss the application of V2DN to the multitarget search problem. First, extending the application of V2DN to the multitarget settings is straightforward, one may design state and action for each robot in the same way as for the single-target use case. However, for the team reward setting, r_t is set to be -1 for each team action that results in no target detection, and set to be R_{cap} for each action that *any* robot detects an target. The Dec-POMDP process will not terminate until the maximal allowed time is reached or all robots become faulty. We will evaluate the performance of V2DN for the multitarget search problem in Section V-E.

V. Simulation Results and Analysis

In this section, we evaluate and compare V2DN under the R-FAC paradigm with state-of-the-art MuRES solutions as well as the vanilla VDN algorithm, in two canonical multirobot search environments: MUSEUM and OFFICE[2]. For state-of-the-art MuRES solutions, we select 1) CE-PG [1]; 2) PD-FAC [7]; 3) DRL-Searcher [54]; 4) FHPE [2]; 5) MILP [15], and 6) mSAC [55] twisted for the MuRES problem, referred to as mSAC-Searcher. Note that 1) all selected baseline algorithms are most recently proposed MuRES algorithms, and are resilient in the sense that when one or several robots malfunction and quit the team, the remaining robots are still functioning to search for the moving target; 2) we do not include swarm intelligence-based algorithms as baselines in that they typically apply to *continuous* environments and the unique interagent communication and coordination mechanism prevents us from adapting related algorithms to the *topological* environments used in this

TABLE I
ALGORITHMS' META-PARAMETER CONFIGURATION

Param	Description	Algorithm	Value
$T_{\max}$	max. # of training epochs	All	3×10^4
α	learning rate	All	0.01
h	planning horizon	FHPE, MILP	5
γ	discount factor	FHPE, MILP	0.98
β_i	balance parameter for robot i	CE-PG, mSAC-Searcher	0.5
M	# of bins for value distribution	PD-FAC	51
$V_{\min}$,$V_{\max}$	min./max. of value distribution	PD-FAC	1, $2n$
n	# of nodes for OFFICE/MUSEUM	All	60/70
ϵ	exploration parameter	VDN,V2DN, PD-FAC	0.1
β	time-discounted factor	V2DN	0.9
r_{cap}	capture reward	V2DN	100

article; 3) FHPE and MILP are two canonical model-based baselines for the MuRES problem. However, neither FHPE nor MILP considers the possibility of individual robot failure, during the model-based planning stage, and we simply assume that all robots are functioning at their respective residing positions. When evaluating the performance of the respective algorithms, we randomly select a subset of the robots according to the preset failure probability (the ρ value), and let them stay in the original position without executing the plan.

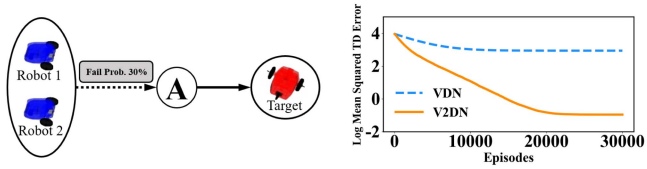
The meta-parameter configurations for selected algorithms as well as V2DN are presented in Table I. All algorithms are (re-)implemented in Python 3.8 with publicly available source code,[4] and the experiments are conducted on a server eqipped with an 8 core 16 threads 3.3 GHz CPU, 16 GB RAM, NVIDIA RTX3080 and the 64-bit Ubuntu system.

In the following several sections, we first present a simple yet illustrative scenario to demonstrate the superior suitability of the log-sum-exp operator in V2DN for addressing the *resilient* multirobot search problem, as opposed to the brute-force summation employed in vanilla VDN. Subsequently, we evaluate and compare the performance of V2DN with vanilla VDN as well as CE-PG [1] in the example scenario depicted in Fig. 1(a) for different failure probability values. Finally, we introduce a novel resiliency performance metric specific to the R-MuRES problem, namely, RS, and then utilize it as the performance metric to benchmark the resiliency performance of V2DN against vanilla VDN and state-of-the-art MuRES solutions in the two canonical multirobot search environments, namely, MUSEUM and OFFICE.

A. Central Value Function Construction Scheme Comparison

As mentioned toward the end of Section IV-A, the vanilla VDN algorithm satisfies all three requirements dictated by the R-FAC paradigm, and thus it also qualifies as an R-MuRES solution. The main difference between VDN and the subsequently proposed V2DN algorithm is that, for the central value function construction, VDN uses the brute-force summation, while V2DN employs the log-sum-exp mechanism. We argue that specifically for the resilient multirobot search problem, since the number of functioning robots is varying over time, it is better to use the log-sum-exp mechanism, which makes the constructed central value function stabler when facing individual

[4]Source code is available at https://github.com/kevinkang1125/r-fac.



(a) (b)

Fig. 3. R-MuRES example and the log of mean squared TD error's convergence process comparison. Capture reward: $r_{\text{cap}} = 100$; learning rate: $\alpha = 0.01$; initialization: $Q^{(1)} = Q^{(2)} = 0$. (a) R-MuRES example 1. (b) TD-error comparison.

failures, and thereby leads to a smaller mean squared TD error. Next, we use a simple yet illustrative example to demonstrate the superiority of log-sum-exp mechanism over brute-force summation in constructing the central value function for the R-MuRES problem.

Fig. 3(a) shows a simple resilient multirobot search use case, where two (possibly faulty) robots residing in the robot depot collaborate to search for one target, which resides in the target depot. The robots need to navigate through Node A to the target depot and thereby detect the target. The unique characteristic of the referred R-MuRES problem is that each of the two robots has a 30% failure probability of quitting the team when navigating to Node A.

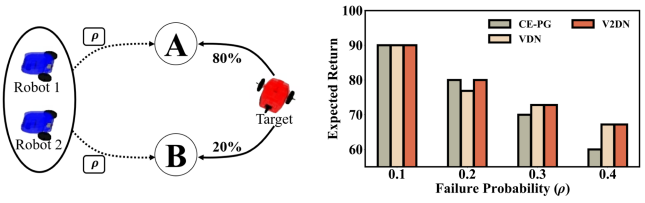
For the use case, VDN will represent the central value function at the robot depot (Q^{tot}) as

$$Q^{\text{tot}} = \begin{cases} Q^{(1)} + Q^{(2)} & \text{w.p. } 49\% \\ Q^{(1)} & \text{w.p. } 21\% \\ Q^{(2)} & \text{w.p. } 21\% \\ 0 & \text{w.p. } 9\% \end{cases} \tag{12}$$

where $Q^{(1)}$ and $Q^{(2)}$ are individual value functions at Node A, for Robot 1 and Robot 2, respectively. On the other hand, V2DN represents Q^{tot} as

$$Q^{\text{tot}} = \begin{cases} \log\left(\exp Q^{(1)} + \exp Q^{(2)}\right) & \text{w.p. } 49\% \\ \log\left(\exp Q^{(1)}\right) = Q^{(1)} & \text{w.p. } 21\% \\ \log\left(\exp Q^{(2)}\right) = Q^{(2)} & \text{w.p. } 21\% \\ 0 & \text{w.p. } 9\%. \end{cases} \tag{13}$$

Fig. 3(b) compares the convergence process of the mean-squared TD error (represented on the logarithmic scale) between VDN (the direct application of the padding technique to multirobot search) and V2DN. From the figure, we observe that both VDN and V2DN are capable of reducing the mean-squared TD error during the initial learning stage. However, when considering the final converged mean-squared TD error, V2DN achieves a notably smaller value compared to VDN (three orders of magnitude smaller). The underlying reason is that the brute-force summation operator is not well suited for the use case of varying number of inputs, and thus the constructed central value function changes drastically as one robot quits the team, resulting in a substantial increase in the TD error. On the other hand, the log-sum-exp mechanism essentially functions as a soft maximum



(a) (b)

Fig. 4. Performance Comparison for different individual failure probabilities. Capture reward: $r_{\text{cap}} = 100$, learning rate: $\alpha = 0.01$; initialization: $Q^{(1)}(A) = Q^{(1)}(B) = Q^{(2)}(A) = Q^{(2)}(B) = 0$; $\epsilon = 0.1$. (a) R-MuRES example 2. (b) Performance comparison.

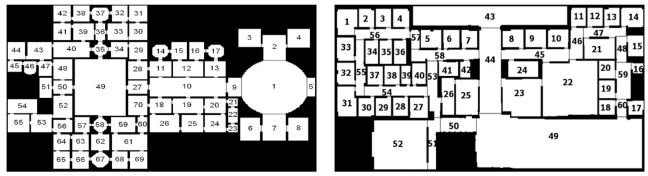
operator. When a robot quits the team, the constructed central value function remains relatively stable, which contributes to a more accurate estimation of the central value function, and hence achieves a smaller mean squared TD error.

B. Performance Comparison in a Simple Scenario

In the last section, we use a simple yet illustrative example to demonstrate that 1) the brute-force summation within VDN leads to a large mean squared TD error when facing individual robot's failures; 2) the log-sum-exp mechanism within V2DN behaves quite well in estimating the central value function considering individual failures, and yields a much smaller mean squared TD error. This subsection presents another simple use case to showcase that V2DN outperforms state-of-the-art MuRES solutions like VDN and CE-PG [1], in the *resilient* MuRES problem with individual failure probabilities.

Fig. 4(a) depicts the scenario where two robots coordinate to search for one target, which has an 80% probability of being in Node A and a 20% probability of being in Node B. Both robots can choose Action A (leading to Node A) or Action B (leading to Node B). However, both actions have a common failure probability denoted as ρ, which causes the robot to malfunction and withdraw from the team. Upon careful calculation, we can conclude that for $\rho \leq 0.25$, separating the two robots to Node A and Node B is the better strategy, while for $0.25 \leq \rho \leq 1$, it is better to conglomerate both robots to Node A for the larger successful target capture probability.

Fig. 4(b) shows the performance comparison among V2DN, VDN, and CE-PG for different ρ values ($\rho \in \{0.1, 0.2, 0.3, 0.4\}$). In the figure, we can see that 1) CE-PG tends to disperse the robots to distinct nodes, due to the cross entropy regularization term. Therefore, it performs well when ρ is small but struggles to find optimal solutions for larger ρ values, e.g., $\rho = 0.3$, $\rho = 0.4$; 2) VDN, on the other hand, tends to aggregate robots towards the high-rewarding search paths. It performs well for larger ρ values but fails to find the optimal solution for $\rho = 0.2$; 3) V2DN leverages the strengths of both VDN and CE-PG, by separating the robots to distinctive nodes for small ρ values (e.g., $\rho = 0.1$, $\rho = 0.2$), while aggregating robots to Node A for large ρ values (e.g., $\rho = 0.3$, $\rho = 0.4$).



(a) (b)

Fig. 5. Canonical multirobot search testing environments from [2], each room is associated with the corresponding node number. Left: MUSEUM; Right: OFFICE. (a) MUSEUM. (b) OFFICE.

C. Resiliency Performance Benchmark and Comparison in Canonical Multirobot Search Environments

So far, we have used two simple use cases to demonstrate that 1) V2DN achieves a much smaller mean-squared TD-error when compared with the vanilla VDN algorithm, and 2) V2DN outperforms or performs as well as VDN and CE-PG for a range of individual failure probabilities. In this section, we further evaluate and compare the resiliency performance of V2DN with state-of-the-art MuRES solutions as well as VDN in two canonical multirobot search environments, namely, MUSEUM and OFFICE, whose layouts are displayed in Fig. 5.

We set up the R-MuRES problem in two multirobot search testing environments as follows: 1) the robots are initialized at the robot depot, i.e., Node 1 for MUSEUM and Node 43 for OFFICE; 2) the target is initialized at a random node other than the robot depot or its adjacent nodes and moves randomly at each time step, choosing between the adjacent nodes or staying in the current node with equal probabilities; 3) the maximal allowed target capture time ($t_{\max}$) is set to be $2n$ (n refers to the number of nodes in Graph $\mathcal{G}$), which means that we terminate the simulation at $2n$ time steps if all robots within the team malfunction or the evaluation algorithm fails to detect the target within the given time limit; and 4) for each configuration, the evaluating algorithm is tested for 1000 independent times, and the corresponding algorithm's expected target capture time (μ) is estimated by the averaged target capture time over 1000 independent runs.

In addition, since R-MuRES is a newly proposed multirobot search problem, there exists no specific performance metric in the literature, to indicate the resiliency performance of a multirobot system for moving target search in the face of individual failures. Therefore, we introduce the following concept (Resiliency Score) to gauge how "resilient" a multirobot search system is against individual robot failures.

Definition 4 (Resiliency score): The resiliency score (η) of an R-MuRES algorithm is defined as the percentage of the ego algorithm's expected target capture time w.r.t. the maximum allowed target capture time.

Suppose that for the R-MuRES problem, the expected target capture time of an algorithm is μ, and the maximum allowed target capture time is $t_{\max}$, we can calculate the resiliency score[5]

[5]The resiliency score ($0 < \eta \leq 1$) of an algorithm is the **smaller**the better.

TABLE II
PREDEPLOYMENT RESILIENCY SCORE COMPARISON BETWEEN V2DN AND STATE OF THE ARTS IN MUSEUM AND OFFICE

	N	ρ	V2DN	VDN	CE-PG	PD-FAC	DRL	MILP	FHPE	mSAC-Searcher
MUSEUM	3	0.1	**0.169**	0.240	0.241	0.268	0.237	0.473	0.352	0.282
		0.2	**0.190**	0.218	0.402	0.309	0.318	0.521	0.484	0.394
		0.3	**0.216**	0.328	0.495	0.499	0.406	0.565	0.538	0.527
	4	0.1	**0.147**	0.206	0.201	0.190	0.176	0.397	0.282	0.221
		0.2	**0.173**	0.265	0.343	0.229	0.253	0.494	0.431	0.306
		0.3	**0.227**	0.313	0.390	0.374	0.314	0.523	0.487	0.390
	5	0.1	0.130	0.191	**0.117**	0.146	0.172	0.332	0.239	0.177
		0.2	**0.174**	0.246	0.224	0.182	0.205	0.411	0.388	0.216
		0.3	**0.198**	0.280	0.266	0.279	0.226	0.465	0.433	0.252
OFFICE	3	0.1	**0.202**	0.264	0.336	0.298	0.271	0.420	0.523	0.263
		0.2	**0.239**	0.311	0.445	0.341	0.334	0.435	0.545	0.334
		0.3	**0.280**	0.353	0.505	0.397	0.440	0.490	0.581	0.461
	4	0.1	**0.158**	0.221	0.220	0.246	0.184	0.413	0.484	0.215
		0.2	**0.189**	0.270	0.355	0.299	0.265	0.415	0.429	0.332
		0.3	**0.249**	0.351	0.446	0.383	0.296	0.444	0.562	0.426
	5	0.1	0.103	0.192	**0.097**	0.120	0.144	0.382	0.327	0.162
		0.2	**0.160**	0.230	0.222	0.198	0.195	0.403	0.437	0.231
		0.3	**0.188**	0.253	0.273	0.315	0.229	0.431	0.496	0.290

Bold numbers indicate the best performance for the configuration, and underlined numbers indicate the second best performance. Note that ρ refers to the individual robot's failure probability before deployment and N refers to the total number of robots. The resiliency score can be referred to (14).

of the algorithm for the R-MuRES problem as

$$\eta = \mu / t_{\max} \times 100\% \tag{14}$$

where the maximum allowed target capture time $t_{\max}$ is set to be $2n$. In the following, we investigate two use cases of resilience for the R-MuRES problem, namely *predeployment resilience*, and *in-execution resilience*. For predeployment resilience, all robots in the team are subject to a certain failure probability (ρ) only in the predeployment stage, and the remaining functioning robots in the execution stage are always operational until the search task is completed. For in-execution resilience, besides the predeployment failure probability, all robots are also subject to a certain failure probability at each execution step.

1) Predeployment Resiliency Score Evaluation and Comparison: Table II presents the predeployment resiliency score comparison amongst R-MuRES algorithms for different number of robots (N) and different failure probabilities (ρ). Note that each robot is subject to a failure probability ρ before deployment, in the extreme case that all robots are faulty, the target's capture time is set to be the maximum allowed target capture time, i.e., $t_{\max}$. The value range of N is selected according to the guideline presented in [1], which states that 5 functional robots are enough to perform the multirobot coordinated search task in both testing environments, and more robots would bring the navigation deadlock problem, which cannot be overlooked in the two environments.

In the table, we observe that 1) V2DN achieves the best resiliency score in most of the experiment configurations barring the two use cases where $\rho = 0.1$ and $N = 5$ in both testing environments, and V2DN ranks the second. For the two specific use cases, we conjecture that with the relatively small ρ value ($\rho = 0.1$) and large number of robots ($N = 5$), one may, to some extent, ignore the faulty probability and simply disperse the robots into the environment for normal search, which is what CE-PG does. 2) With large ρ values, e.g., $\rho = 0.3$, V2DN dominates other algorithms in terms of the resiliency score in both testing environments, in that no other algorithm is specifically designed to consider individual robots' failure.

2) In-Execution Resiliency Score Evaluation and Comparison: Table III displays the in-execution resiliency score comparison for different configuration parameters, i.e., the number of robots (N), the failure probability at each step (ρ). Note that for the in-execution resiliency score evaluation, since at each execution step, all robots are subject to a failure probability ρ, we cannot set the value to be too high. For example, when we set $\rho = 0.1$, and there are $N = 4$ robots, it means that after 10 executions steps, the expected number of malfunctioning robots is already 4. Therefore, when evaluating the *in-execution* resiliency score, we set relatively smaller ρ values, i.e., $\rho \in \{0.04, 0.06, 0.08\}$, which makes the robots endure a longer task execution duration.

In the table, several noteworthy observations can be made: 1) not surprisingly, V2DN ranks the first in terms of the RS for most experimental configurations, in that it explicitly incorporates the individual robot's failure into decision-making process, while all other algorithms simply ignore it; 2) for the same N value, when we increase ρ, RS increases for all algorithms; 3) for the same ρ value, when we increase N, RS decreases. The later two observations are rational, in that it conforms to the intuition that, in general, more faulty robots would increase the search time in expectation; on the other hand, more robots would decrease the expected search time.

Furthermore, from Tables II and III, we can see that although in theory, model-based methods would excel via online replanning when individual robots fail or leave the team, provided that there are no further robot failures in the future and all

TABLE III
IN-EXECUTION RESILIENCY SCORE COMPARISON BETWEEN V2DN AND STATE OF THE ARTS IN MUSEUM AND OFFICE

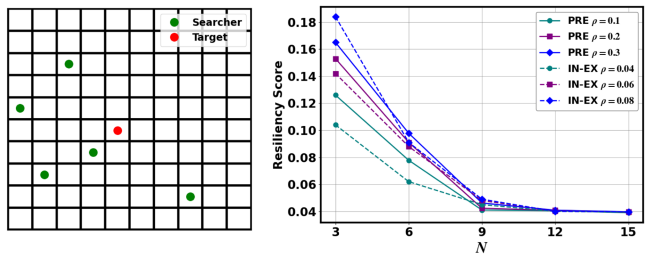
	N	ρ	V2DN	VDN	CE-PG	PD-FAC	DRL	MILP	FHPE	mSAC-Searcher
MUSEUM	3	0.04	**0.214**	0.322	0.299	0.321	0.312	0.404	0.543	0.356
		0.06	**0.289**	0.437	0.413	0.543	0.411	0.554	0.639	0.538
		0.08	**0.335**	0.521	0.529	0.667	0.580	0.606	0.684	0.665
	4	0.04	**0.195**	0.311	0.270	0.282	0.286	0.367	0.459	0.305
		0.06	**0.269**	0.373	0.387	0.409	0.382	0.418	0.560	0.481
		0.08	**0.303**	0.547	0.489	0.519	0.473	0.545	0.651	0.608
	5	0.04	**0.178**	0.288	0.245	0.194	0.248	0.341	0.318	0.257
		0.06	**0.230**	0.294	0.348	0.297	0.329	0.401	0.416	0.395
		0.08	**0.285**	0.475	0.423	0.472	0.373	0.475	0.621	0.545
OFFICE	3	0.04	0.233	0.328	**0.229**	0.306	0.333	0.457	0.584	0.378
		0.06	**0.284**	0.443	0.471	0.513	0.454	0.508	0.676	0.536
		0.08	**0.364**	0.566	0.591	0.662	0.544	0.638	0.746	0.723
	4	0.04	**0.217**	0.283	0.218	0.274	0.305	0.410	0.430	0.307
		0.06	**0.257**	0.336	0.430	0.458	0.417	0.474	0.656	0.477
		0.08	**0.315**	0.459	0.542	0.579	0.511	0.488	0.675	0.695
	5	0.04	0.183	0.225	**0.163**	0.215	0.196	0.398	0.344	0.256
		0.06	**0.216**	0.302	0.309	0.364	0.325	0.422	0.626	0.445
		0.08	**0.264**	0.361	0.362	0.473	0.352	0.450	0.644	0.615

Bold numbers indicate the best performance for the configuration, and underlined numbers indicate the second best performance. Note that ρ refers to the individual robot's failure probability during execution and N refers to the total number of robots. The resiliency score can be referred to (14).

information can be communicated in real time to a central server for the centralized replanning process. However, there are use cases that do not strictly adhere to the two aforementioned conditions, and hence will degrade the performance of the model-based re-planning strategy. Use case 1: the remaining team member might encounter probabilistic failure again, which makes the strategy of simply replanning not an optimal one, Fig. 1(a) presents such an illustrative example, where model-based planning or replanning strategies produce a suboptimal strategy via simply dispersing the robots into the environment. Use case 2: the replanning process is essentially a *centralized* strategy, which needs to gather all remaining robots' status information. In practice, when one or several robots fail and quit the team, other robots might not know the information in real time, and they are still following the previously calculated search plan. In this case, the centralized replanning strategy is unrealistic, and one needs a resilient multirobot search strategy, which plans for the use case of individual robot failures beforehand.

D. V2DN's Scalability Evaluation in a 10×10 Grid World

The original two testing environments, namely MUSEUM and OFFICE, are too "narrow," in a sense that 5 robots already achieve a good overall search performance. Simply increasing the number of robots would not decrease the expected target captured time significantly. Therefore, in this section, we create a 10×10 grid world as the multirobot search testing environment, where each robot and the moving target are situated at one of the grids, as shown in Fig. 6(a). The target is deemed as captured when any robot occupies in the same grid as the moving target does. Fig. 4(b) shows the scalability test of V2DN for different number of robots, different ρ values and different resilient use cases, i.e., predeployment resiliency and in-execution resiliency.



(a) (b)

Fig. 6. V2DN's scalability evaluation. (a) 10×10 grid world. (b) Performance comparison.

Note that 1) $t_{\max} = n = 100$, and we set the maximum number of robots as 15 in that when the number of robots exceeds that number, the collision avoidance process between robots become a significant issue.

E. V2DN's Application to the Multitarget Search Problem

We evaluate the performance of V2DN for multitarget detection in the MUSEUM and OFFICE environments, respectively, for different problem setups, e.g.,, different number of robots, and different ρ values. We set the total number of targets as **20**, and the maximal allowed time is $2n$, where n refers to the total number of nodes in the environment.

Table IV presents the final average number of captured targets for different problem setups, e.g.,, different ρ values, different number of robots, predeployment resiliency or in-execution resiliency use case. For each evaluation, the final trained algorithm is run for 1000 independent times, and the averaged number of captured targets is reported in the Table. In Table IV, we can see that as we increase ρ, the average number of captured targets decreases. The underlying reason is that the individual robots

TABLE IV
V2DN'S PERFORMANCE EVALUATION FOR THE MULTITARGET SEARCH PROBLEM

N		3			4			5		
Predeployment	ρ	0.1	0.2	0.3	0.1	0.2	0.3	0.1	0.2	0.3
	MUSEUM	16.47	15.68	14.17	16.45	16.16	14.76	17.98	16.40	15.52
	OFFICE	17.13	15.29	14.38	17.25	16.80	14.63	18.12	16.96	14.85
In-Execution	ρ	0.04	0.06	0.08	0.04	0.06	0.08	0.04	0.06	0.08
	MUSEUM	16.88	13.35	12.14	17.14	15.34	11.38	17.55	14.66	12.92
	OFFICE	15.71	14.82	14.65	16.85	15.23	14.39	17.10	15.79	15.25

Note that ρ refers to the individual robt's failure probability during execution and N refers to the total number of robots.

TABLE V
ADVERSARIAL MOVING TARGET EXPERIMENT FOR V2DN (RESILIENCY SCORE)

N		3			4			5		
Pre-Deployment	ρ	0.1	0.2	0.3	0.1	0.2	0.3	0.1	0.2	0.3
	MUSEUM	0.507	0.549	0.568	0.494	0.547	0.554	0.497	0.528	0.553
	OFFICE	0.685	0.717	0.783	0.667	0.671	0.744	0.575	0.603	0.646
In-Execution	ρ	0.04	0.06	0.08	0.04	0.06	0.08	0.04	0.06	0.08
	MUSEUM	0.427	0.567	0.610	0.410	0.517	0.607	0.396	0.532	0.602
	OFFICE	0.545	0.631	0.779	0.530	0.618	0.770	0.489	0.694	0.768

Note that ρ refers to the individual robt's failure probability during execution and N refers to the total number of robots.

are more likely to become faulty and thereby quit the team with larger ρ values.

F. V2DN's Application to Adversarial Moving Target Search

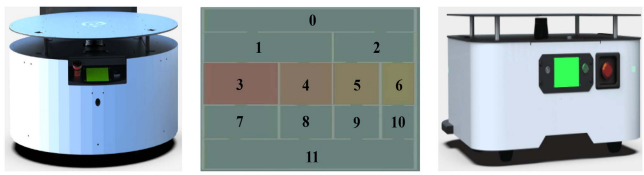
In this section, we evaluate V2DN's performance when applied to the *adversarial* moving target search problem. First, the V2DN algorithm can be directly applied to detect the *simple* adversarial moving target. Here, the term "simple" refers to the fact the the moving target just acts reactively to avoid being detected. The target's motion dynamics is to simply go to the neighbor node which is the farthest from the search team, i.e., the node whose distance to the nearest searcher is the largest. In this case, we evaluate V2DN's performance in both MUSEUM and OFFICE environments for different ρ values, and report the results in Table V. In the table, we can see that with increase ρ values, the RS value increases, which means the robots spent more time to detect the adversarial moving target.

However, for the superior adversarial moving target, e.g., the target's motion dynamics adapts with the robot team's search strategy. The overall problem becomes a zero-sum game, and we need to employ game theory to let V2DN reach certain equilibrium, e.g., Nash Equilibrium. We put it as one of the future work directions.

G. Discussions

So far, we have evaluated V2DN's application to the R-MuRES problem, and also tested its performance for cases of scalability, multitarget search and adversarial moving target search. Here, we wish to articulate that learning-based methods are not the only preferred solution scheme to the R-MuRES problem proposed in this article.

Both model-based and learning-based approaches have their respective advantages. Model-based algorithms are powerful and widely used for solving the MuRES problem, particularly when the environment is well-characterized and the assumptions



(a) (b) (c)

Fig. 7. Experiment setup: Fig. 7(a) is a C30 robot, which acts as the moving target; Fig. 7(c) is a S30 robot with embedded LiDAR and camera, used as the robot testbed, and Fig. 7(b) shows the design map of the multirobot search environment, which mimics a 12-room house. The target starts at Room 0, and the robot team are initialized at Room 11. (a) C30 (Target). (b) Design map. (c) S30 (Searcher).

in the modeling process hold. On the other hand, learning-based methods excel in adapting robots' behavior during interactions with the environment, making them well-suited for search tasks where the target's motion dynamics and sensor characteristics are not well modeled.

In summary, both model-based and learning-based approaches are capable of offering viable solutions to the R-MuRES problem, and the choice between them should be based on the availability of model parameters and training data samples.

VI. SYSTEM INTEGRATION AND EXPERIMENTAL RESULTS

The last section evaluates and benchmarks V2DN's performance against state-of-the-art multirobot search algorithms in two canonical simulation environments. In this section, we deploy V2DN to a real multirobot system, and test its resilient search functionality in a self-constructed indoor environment for the R-MuRES problem as a proof of concept.

The robot testbed, as shown in Fig. 7(c), is a four-wheel differential drive S30 robot, equipped with Velodyne's Puck LiDAR sensor and Intel RealSense camera for object detection. We integrate our previously developed mapping and localization

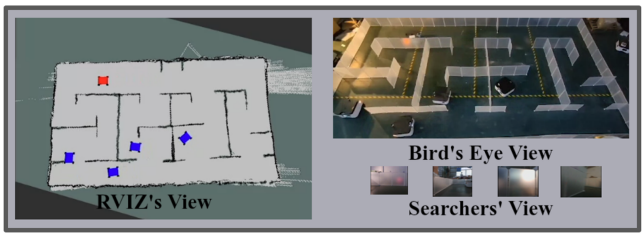
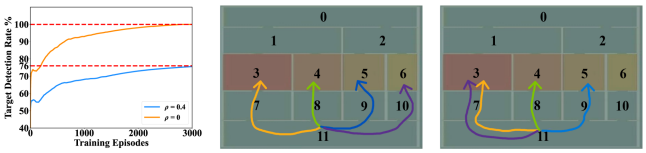


Fig. 8. Snapshot of the resilient multirobot search process.



(a) (b) (c)

Fig. 9. V2DN's learning curve comparison and final multirobot search paths illustration for different predeployment faulty probabilities (ρ). (a) Learning curve. (b) $\rho = 0.0$. (c) $\rho = 0.4$.

module, namely, E-LOAM [56], as well as the canonical motion planning and obstacle avoidance modules into the robot testbed. On the other hand, the nonadversarially moving target is a differential drive C30 robot, as shown in Fig. 7(a). Both robots and the moving target are controlled by ROS. The self-constructed indoor environment, whose design map is shown in Fig. 7(b), mimics a 12-room housing environment, with four S30 robots initiated at Room 11, and the moving target starting at Room 0. The target moves probabilistically to Rooms 3 and 4 via Room 1, and to Rooms 5 and 6 via Room 2. Different colors indicate different probabilities of the target residing in different rooms in the final time step, with the yellower the lower probability and the redder the higher probability.

While a demonstrative video has been uploaded together with the manuscript,[6] Fig. 8 shows one snapshot of the integrated multirobot search process, where the left half figure illustrates the multirobot search system's overall status via RViz; the top right figure displays the bird's-eye-view of the whole operational system, and the four bottom right figures show all individual searching robots' local vision information.

We evaluate two use cases of the predeployment resiliency, i.e., $\rho = 0.0$, and $\rho = 0.4$, with $\rho = 0.0$ indicating the normal MuRES problem, and $\rho = 0.4$ corresponding to the R-MuRES problem. Fig. 9 shows the mean learning curve comparison as well as the correspondingly sketched final multirobot search paths for the two use cases. In the figure, we can see that (1) when $\rho = 0.0$ [see Fig. 9(b)], V2DN is able to disperse the robots in the environment for efficient moving target search and the target's successful detection rate reaches 100% [as indicated in Fig. 9(a)]; (2) when $\rho = 0.4$ [see Fig. 9(c)], since the robots are subject to a high faulty probability, V2DN is able to conglomerate two of the robots to Room 3, which has the highest target residing probability, to compensate the individual robots' malfunction possibility for more efficient target search, and reach a 76% target detection rate [also shown in Fig. 9(a)]. Both quantitative results are evaluated with 100 independent trials, and also reported in the uploaded demonstrative video.

[6]https://github.com/kevinkang1125/r-fac/tree/main/results.

VII. Conclusion and Future Work

This article introduces a novel multirobot search problem, namely, R-MuRES, whose unique characteristic is that the individual robots within the team might malfunction and quit the team during task execution. We then propose the R-FAC paradigm, which outlines three essential requirements for a value function factorized MARL algorithm to qualify as an R-MuRES solution. Subsequently, an instantiated R-MuRES algorithm, namely, V2DN is proposed and validated with both numerical simulations in canonical multirobot search environments and physical experiments on a real multirobot system.

In the future, we would like to extend/improve V2DN by considering the following use cases: 1) when the individual robot malfunctions, instead of quitting the team, it remains at the current node and acts as a *stationary* sensor with the target detection ability; 2) each robot is equipped with a probabilistic sensor which introduces false negative and/or false positive target detection probabilities, deviating from the current assumption of a perfect sensor; 3) the robot team is deployed to search for the *adversarial* moving target, where the target adapts its motion dynamics based on the robot team's search strategy; and 4) the application of V2DN to heterogeneous robots and the faster moving target. Moreover, we are also keen on analyzing the theoretical bounds of the multirobot resilient search problem when given the environment structure, target's motion dynamics and each robot's failure probability.

References

[1] H. Guo, Z. Liu, R. Shi, W.-Y. Yau, and D. Rus, "Cross-entropy regularized policy gradient for multirobot nonadversarial moving target search," *IEEE Trans. Robot.*, vol. 39, no. 4, pp. 2569–2584, Aug. 2023.

[2] 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.

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

[4] L.-G. Pan, Q. Lu, X. Xie, J. Wang, and J. Wang, "A probability distribution based cooperative search approach for stochastic source localization," in *Proc. IEEE Int. Symp. Ind. Electron.*, 2018, pp. 585–590.

[5] L. Collins, P. Ghassemi, E. T. Esfahani, D. Doermann, K. Dantu, and S. Chowdhury, "Scalable coverage path planning of multi-robot teams for monitoring non-convex areas," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 7393–7399.

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

[7] 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. Automat. Lett.*, vol. 7, no. 4, pp. 8869–8876, Oct. 2022.

[8] R. Hu, N. Tan, and F. Ni, "A new scheme for cooperative hunting tasks with multiple targets in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Biomimetics* 2021, pp. 1816–1822.

[9] A. Asgharnia, H. M. Schwartz, and M. Atia, "Multi-invader multi-defender differential game using reinforcement learning," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2022, pp. 1–8.

[10] C. de Souza, R. Newbury, A. Cosgun, P. Castillo, B. Vidolov, and D. Kulić, "Decentralized multi-agent pursuit using deep reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4552–4559, Jul. 2021.

[11] A. Goldhoorn, A. Garrell, R. Alquézar, and A. Sanfeliu, "Searching and tracking people with cooperative mobile robots," *Auton. Robots*, vol. 42, no. 4, pp. 739–759, 2018.

[12] F. Yan, K. Di, J. Jiang, Y. Jiang, and H. Fan, "Efficient decision-making for multiagent target searching and occupancy in an unknown environment," *Robot. Auton. Syst.*, vol. 114, pp. 41–56, 2019.

[13] 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, no. 2, 2020, Art. no. 022104.

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

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

[16] J. Berger, N. Lo, and M. Barkaoui, "Static target search path planning optimization with heterogeneous agents," *Ann. Operations Res.*, vol. 244, pp. 295–312, 2016.

[17] 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.

[18] W. Yue, X. Guan, and L. Wang, "A novel searching method using reinforcement learning scheme for multi-UAVs in unknown environments," *Appl. Sci.*, vol. 9, no. 22, 2019, Art. no. 4964.

[19] R. Ravichandran, D. Ghose, and K. Das, "UAV based survivor search during floods," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2019, pp. 1407–1415.

[20] J. Szklarski, L. Białek, and A. Szałs, "Paraconsistent reasoning in cops and robber game with uncertain information: A simulation-based analysis," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 27, no. 03, pp. 429–455, 2019.

[21] Z. Zhang, X. Wang, Q. Zhang, and T. Hu, "Multi-robot cooperative pursuit via potential field-enhanced reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 8808–8814.

[22] I. Vandermeulen, R. Groß, and A. Kolling, "Sampling-based search for a semi-cooperative target," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 4774–4781.

[23] J. Cui, D. Li, P. Liu, J. Qin, Y. Ma, and Z. Lu, "Game-model prediction hybrid path planning algorithm for multiple mobile robots in pursuit evasion game," in *Proc. IEEE Int. Conf. Unmanned Syst.*, 2021, pp. 925–930.

[24] S. Yi, C. Nam, and K. Sycara, "Indoor pursuit-evasion with hybrid hierarchical partially observable Markov decision processes for multi-robot systems," in *Distributed Autonomous Robotic Systems*, N. Correll, M. Schwager, and M. Otte, Eds. Berlin, Germany: Springer International Publishing, 2019, pp. 251–264.

[25] L. Gregorin, E. Freire, E. Carvalho, L. Molina, and S. Givigi, "Evolutionary robotics applied to the multi-robot worst-case pursuit-evasion problem," in *Proc. IEEE Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf.*, 2016, pp. 1–7.

[26] 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.

[27] V. Shree, B. Asfora, R. Zheng, S. Hong, J. Banfi, and M. Campbell, "Exploiting natural language for efficient risk-aware multi-robot SaR planning," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3152–3159, Apr. 2021.

[28] H. Osooli, P. Robinette, K. Jerath, and R. Ahmadzadeh, "A multi-robot task assignment framework for search and rescue with heterogeneous teams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Adv. Multi-Agent Learn.-Coordination, Percep., Control Workshop*, 2023, pp. 1–6.

[29] Z. Ismail and M. Hamami, "Systematic literature review of swarm robotics strategies applied to target search problem with environment constraints," *Appl. Sci. (Switzerland)*, vol. 11, no. 5, 2021, Art. no. 2383.

[30] J. T. 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.

[31] J. Li and Y. Tan, "A probabilistic finite state machine based strategy for multi-target search using swarm robotics," *Appl. Soft Comput.*, vol. 77, pp. 467–483, 2019.

[32] M. Dadgar, M. S. Couceiro, and A. Hamzeh, "RbRDPSO: Repulsion-based RDPSO for robotic target searching," *Iranian J. Sci. Technol., Trans. Elect. Eng.*, vol. 44, pp. 551–563, 2020.

[33] Q. Yang and R. Parasuraman, "Game-theoretic utility tree for multi-robot cooperative pursuit strategy," in *Proc. 54th Int. Symp. Robot.*, ISR Europe 2022, 2022, pp. 278–284.

[34] M. Casini and A. Garulli, "A two-pursuer one-evader game with equal speed and finite capture radius," *J. Intell. Robot. Syst.*, vol. 106, no. 4, 2022, Art. no. 77.

[35] R. M. Francos and A. M. Bruckstein, "Search for smart evaders with swarms of sweeping agents," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1080–1100, Apr. 2022.

[36] A. Oroojlooy and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *Appl. Intell.*, vol. 53, no. 11, pp. 13677–13722, 2023.

[37] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: A survey," *Artif. Intell. Rev.*, vol. 55, no. 2, pp. 895–943, 2022.

[38] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 4295–4304.

[39] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. Int. Conf. Auton. Agents MultiAgent Syst.*, 2018, pp. 2085–2087.

[40] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," *Adv. Neural Inf. Process. Syst.*, vol. 21, no. 178, pp. 1–51, 2020.

[41] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "QPLEX: Duplex dueling multi-agent Q-learning," in *Proc. Int. Conf. Learn. Representations*, 2021.

[42] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, vol. 97, pp. 5887–5896.

[43] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6380–6391.

[44] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2974–2982.

[45] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 24611–24624.

[46] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*, 1st ed. Berlin, Germany: Springer Publishing Company, Incorporated, 2016.

[47] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: The MIT Press, 2018.

[48] M. R. Roesch, A. R. Taylor, and S. Geoffrey, "Encoding of time-discounted rewards in orbitofrontal cortex is independent of value representation," *Neuron*, vol. 51, no. 4, pp. 509–520, 2006.

[49] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS 2014 Workshop Deep Learn.*, 2014.

[50] A. Vaswani et al., "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[51] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

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

[53] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman, 1979.

[54] H. Guo, Q. Peng, Z. Cao, and Y. Jin, "DRL-Searcher: A unified approach to multirobot efficient search for a moving target," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 3, pp. 3215–3228, Mar. 2024.

[55] 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.*, vol. 34, no. 12, pp. 10980–10992, Dec. 2023.

[56] H. Guo, J. Zhu, and Y. Chen, "E-LOAM: LiDAR odometry and mapping with expanded local structural information," *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1911–1921, Feb. 2023.

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

He has been working as a Postdoc Researchers with Almende, Nanyang Technological University and Singapore MIT Alliance for Research and Technology, Singapore, respectively. In 2021, as a Scientist he joins the Institute for Infocomm Research (I2R) in A∗STAR, Singapore. His research interests include reliable path planning and learning under uncertainties, and multi-robot efficient search.

Qi Kang received the bachelor's degree in mechanical engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2021, and master's degree in mechanical engineering from the National University of Singapore (NUS), Singapore, in 2023. He is currently working toward the Ph.D degree in transportation with Northeastern University (Boston). His research interest includes safe reinforcement learning, autonomous fleet management and multi-agent system.

Wei-Yun Yau (Senior Member, IEEE) received the B.Eng. degree electrical engineering from the National University of Singapore, Singapore, in 1992, and the M.Eng. degree 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, A∗STAR, Singapore, as the Head of Robotics and Autonomous Systems Department. Since 2019, he has been a Deputy Director (Technology) with the Rehabilitation Research Institute of Singapore, Singapore. H has authored or coauthored 13 patents granted and more than 150 publications. His research interest includes intelligent robots, biometrics, robotic perception and navigation and human-robot interaction.

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

Chee-Meng Chew (Senior Member, IEEE) received the B.Eng. and M.Eng. degree in mechanical engineering from the National University of Singapore (NUS), Singapore, in 1991, and the S.M. and Ph.D. degree in mechanical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA in 1998 and 2000, respectively.

After completing his Ph.D. degree, he joined NUS as an Assistant Professor with the Department of Mechanical Engineering, where he is currently an Associate Professor. His main research interests include robot learning, autonomous systems, bioinspired and biomimetic systems, novel actuation and mechanism, assistive device. He is currently a Senior Member of IEEE.

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

She is the Andrew (1956) and Erna Viterbi Professor of electrical engineering and computer science and Director of the Computer Science and Artificial Intelligence Laboratory, MIT, 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 ACM and AAAI, and a Member of the National Academy of Engineering, and the American Academy for Arts and Science.