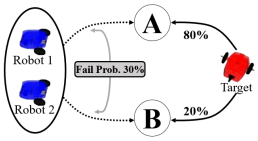
R-FAC: Resilient Value Function Factorization for Multi-Robot Efficient Search with Individual Failure Probabilities

Hongliang Guo[1], Qi Kang[1], Wei-Yun Yau[1], Chee-Meng Chew[2] and Daniela Rus[3]

***Abstract*—This paper investigates the *resilient* multi-robot efficient search problem (R-MuRES), which aims at coordinating multiple robots to detect a 'non-adversarial' moving target with 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 thereby poses new challenges to the research community. 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 (TD) error across various team compositions. R-FAC stipulates that the individual global maximum (IGM) 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 multi-robot 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 multi-robot system in a self-constructed indoor environment as the proof of concept.**

***Index Terms*—R-FAC, resilient multi-robot efficient search (R-MuRES), non-adversarial moving target, individual global maximum (IGM), variational value decomposition network (V2DN).**



(a) R-MuRES Scenario

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

(b) Solution Quality Comparison

Fig. 1: Illustration of an R-MuRES problem. Fig. 1(a) shows a simple multi-robot 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, which disperses the two robots to Node A and Node B, and the optimal R-MuRES solution which conglomerates both robots at Node A for larger target capture probability, considering potential robot failures.

I. Introduction

MULTI-ROBOT efficient search (MuRES) has emerged as a prominent and interdisciplinary research subject, situated at the confluence of fundamental research and industrial applications. From the fundamental research standpoint, MuRES serves as a practical and challenging experimental platform for exploring diverse domains related to artificial intelligence, including multi-agent reinforcement learning (MARL), game theory, and multi-robot coordination and exploration. Additionally, MuRES represents a vital research frontier in numerous industrial applications related to multiple robots, such as search and rescue in hazardous environments [1], collaborative source leakage detection [2], and surveillance and monitoring for regional security [3]. 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 multi-robot 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 *resilient* multi-robot efficient search (R-MuRES) problem. In this problem, some robots might malfunction and leave the team during task execution, while the remaining multi-robot system must continue to function resiliently with a good overall performance. 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 resilient multi-robot efficient search (R-MuRES) algorithm to solve the R-MuRES problem, as canonical MuRES solutions may not be capable of delivering the optimal solution.

This paper proposes the resilient value function factorization (R-FAC) paradigm as a guideline for addressing the R-MuRES

[1]Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore, 138632.
[2]National University of Singapore (NUS), Singapore, 119077.
[3]Computer Science, and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, 02139. rus@csail.mit.edu.

problem. The R-FAC paradigm stipulates three critical requirements: (1) adherence to the individual global maximum (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 temporal difference (TD) error across various functional team configurations. To instantiate the resilient value function factorization (R-FAC) paradigm, we present the variational value decomposition network (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, against state-of-the-art MuRES solutions as well as the canonical value decomposition network (VDN) algorithm, and also validate the applicability of V2DN with a real multi-robot system in a self-constructed indoor environment.

The paper makes the following contributions to the field of multi-robot search: (1) initiation of a new MuRES problem, namely *resilient* multi-robot efficient search, which has been surprisingly overlooked in the field of multi-robot 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 resiliency score, to evaluate and compare the performance of V2DN with several state-of-the-art MuRES solutions in two canonical multi-robot search testing environments, with the results reported on a publicly accessible leader board.

The remainder of the paper is organized as follows. In Section II, we provide a brief literature review of multi-robot search and multi-agent reinforcement learning (MARL). Then, we formulate the R-MuRES problem and introduce related background knowledge of value function factorization 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 resiliency score against several state-of-the-art MuRES solutions in Section V. We then deploy V2DN to a real multi-robot system and demonstrate its effectiveness in Section VI, and conclude the paper with a discussion of future work in Section VII.

II. Literature Review

Multi-robot 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 multi-robot efficient search (MuRES), multi-robot reliable search (MuRRS) and multi-robot guaranteed search (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 pre-defined 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 [4] and [5], respectively. Moreover, we briefly review the recent trends in multi-agent reinforcement learning (MARL) towards the end of this section, in that the R-FAC paradigm is essentially targeting the MuRES problem from MARL's perspective.

A. Multi-Robot 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 [6]–[8], or unbounded and unstructured with stationary obstacles [1], [9]. On the other hand, discrete environments are described by occupancy grid maps [10], [11] or topological graphs [12]–[15].

The target's motion behavior also plays a significant role in designing corresponding multi-robot search strategies. In the MuRES literature, the target may either remain stationary during the search process [10], [16], [17], move in a non-adversarial manner that is independent of the robot team's search strategy [12], [18], [19], or move adversarially in an attempt to evade detection [7], [8], [20].

Finally, the robot team's sensor characteristics, *i.e.*, sensor range, sensor accuracy, are essential factors for categorizing the MuRES literature. In continuous environments, the robot's sensor range can be circular [8], [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 [12], [14], [24] or arbitrary range detection [15], [25]. Additionally, sensors can be perfect or probabilistic, with perfect sensors always accurately detecting the target within their range [14], while probabilistic sensors have a probability of false negative and/or false positive detections [11], [12], [15], [26].

B. Multi-Robot Efficient Search: Methodologies

The last subsection dissects the MuRES literature along various taxonomies, and in this subsection, 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) mathematical optimization, (2) swarm intelligence, (3) multi-agent reinforcement learning and (4) game theory.

Mathematical optimization has been widely deemed as the most canonical method to solve the MuRES problem. Algorithms within this category formulate the MuRES problem within the mathematical programming framework, and utilize existing optimization solvers to find the optimal solution [13]–[15], [27]. 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 mixed integer linear programming (MILP) problem, and the canonical optimization solver, *i.e*., 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. Additionally, the MILP formulation may suffer from scalability issues when the size of the problem or the number of robots increases. Nonetheless, mathematical optimization remains 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 mathematical formulation 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 guidelines. 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 [28]–[31]. For example, Li *et al*. design a three-state finite state machine (FSM), 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 [30]. Generally speaking, swarm intelligence is a useful approach for collective search with advantages such as being scalable and robust. However, current swarm intelligence methods lack a clear relationship between individual behavior and the MuRES objective, which makes it challenging to optimize performance through individual mechanism tuning or redesign, limiting the optimization process to pre-designed local rules with uncertain global outcomes.

Learning-based approaches have recently emerged as a promising solution to the MuRES problem, as evidenced in [8], [12], [18], [26]. Algorithms within this category treat MuRES as a multi-agent sequential decision-making problem and typically cast it into the decentralized partially observable Markov decision process (Dec-POMDP) framework. Several decentralized policy optimization algorithms are proposed, such as multi-agent deep deterministic policy gradient (MADDPG) [26], cross entropy regularized policy gradient (CEPG) [12], probability density factorization (PD-FAC) [5]. 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 paper aims at addressing the issue by proposing a resilient MARL solution which accounts for the possibility of sudden team member failures.

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 pre-planned multi-robot 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 as solutions, see [23], [32], [33] as examples.

C. Multi-Agent Reinforcement Learning (MARL)

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 subsection, we deliver a brief literature review of cooperative MARL for necessary contexts of R-FAC and V2DN. Interested audiences are referred to [34], [35] for the comprehensive literature review.

As a sub-field 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 multi-robot cooperation, multi-vehicle 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 value function factorization (VFF) methods while adhering to the individual global maximum (IGM) principle [36]–[40], or directly calculate the (approximated) decentralized policy gradient for each individual agent [41]–[43]. For example, Sunehag *et al*. propose value decomposition network (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 temporal difference (TD) error of the reconstructed central value function [37].

However, despite the success of cooperative MARL algorithms in various domains, little attention has been paid to the resiliency aspect of multi-agent reinforcement learning, where some agents within the multi-agent system might malfunction and quit the team during task execution, without notifying others. In this paper, 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

TABLE I: List of Major Notations Used in the Paper

Notations	Descriptions
N	number of robots in the pre-deployment phase
$\mathcal{G}(\mathcal{V},\mathcal{E})$	the 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

model, and then establish R-MuRES within the decentralized partially observable Markov decision process (Dec-POMDP) framework. Subsequently, we introduce the canonical MARL algorithm, namely value decomposition network (VDN), under the centralized training and decentralized execution (CTDE) scheme. A list of major notations used throughout the paper is presented in Table I, 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 non-adversarial 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 $\mathcal{N}$ ($\lvert\mathcal{N}\rvert = n$) refers to the set of nodes, and $\mathcal{E}$ ($\lvert\mathcal{E}\rvert = m$) 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 [5], [12], [15], [26] as examples. In practice, one may simply transform the non-unit-cost topological graph into a unit-cost one by breaking the 'long' edges into several connected 'unit-cost' sub-edges.

The target, whose position at time step t is denoted as e_t, moves non-adversarially, *i.e.*, the target's motion is *independent* of the robot team's search strategy. In this paper, 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, $\pi^{(i)} = \pi^{(i)}\big(\boldsymbol{p}_{\leq t}^{(i)}\big)$, where $\boldsymbol{p}_{\leq t}^{(i)} = (p_0^{(i)}, p_1^{(i)}, \ldots, p_t^{(i)})^\top$ indicates robot i's position sequence from time 0 to time t. The decision-making policy $\big(\pi^{(i)}\big)$ dictates the next step position of robot i, *i.e.*, $p_{t+1}^{(i)} = \pi^{(i)}\big(\boldsymbol{p}_{\leq t}^{(i)}\big)$. 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 $\big(\rho^{(i)}\big)$ 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 $\boldsymbol{\pi} = \big(\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(N)}\big)^\top$, with minimal expected time to detect the non-adversarially 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 resilient multi-robot efficient search 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 [44]. 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 team reward $r_t = -1$ at each normal time step and $r_t = r_{\text{cap}}$ if the moving target is captured by any functioning robot, and the whole process terminates. 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 process evolves, in that some agents might malfunction and withdraw from the team during task execution.

B. Value Decomposition Network (VDN)

As the R-FAC paradigm is essentially treating the R-MuRES problem from cooperative multi-agent reinforcement learning's perspective, and the instantiated variational value decomposition network (V2DN) algorithm under the R-FAC paradigm evolves from the canonical VDN algorithm, we describe the mechanisms of VDN in this subsection.

VDN is deemed as one of the prevalent cooperative MARL methods along the direction of value function factorization (VFF) [37]. 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.*,

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

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 centralized training and decentralized execution (CTDE) scheme, 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. The training objective is to the find the best $\boldsymbol{\theta_i}$ values which minimize the system-level joint TD error. While during the online decentralized execution stage, each agent simply makes decisions based on the individual value function with the ϵ-greedy action-selection strategy [45].

IV. The R-FAC Paradigm and V2DN

This section presents resilient value function factorization (R-FAC) as the MARL's solution paradigm to the R-MuRES problem, and then introduces variational value decomposition network (V2DN) as one of the instantiated algorithms under the R-FAC paradigm.

A. The 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 $Q^{(i)}$ $\left(i \in \{1, 2, \ldots, N\}\right)$, 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.*, $Q^{\text{tot}} = f\left(Q^{(1)}, Q^{(2)}, \ldots, Q^{(N)}\right)$. 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} with respect to $Q^{(i)}$ must exist and is greater than or equal to zero, *i.e.*, $\forall i \in \{1, 2, \ldots, N\}$, $\partial Q^{\text{tot}} / \partial Q^{(i)} \geq 0$;
3) the mean squared temporal difference (TD) error with respect to 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 individual global maximum (IGM) principle [36] 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 with respect to 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. However, the brute-force summation operator within VDN is not well suited for the multi-robot 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 multi-robot search context in Section V-A. In the next subsection, we introduce the variational value decomposition network (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 (V2DN)

The last subsection 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 the requirements and thus can be deemed as one of the instantiated R-FAC algorithms. In this subsection, we introduce the variational value decomposition network (V2DN) under the Dec-POMDP framework, as another instantiated R-FAC algorithm for the R-MuRES problem. V2DN follows the centralized training and decentralized execution (CTDE) scheme. During the decentralized execution (DE) stage, each robot simply follows the ϵ-greedy policy with respect to its own individual value function ($Q^{(i)}$). On the other hand, during the centralized training (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 with respect to 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: (1) the individual value function $\left(Q^{(i)}\right)$ takes as inputs an ever-growing position

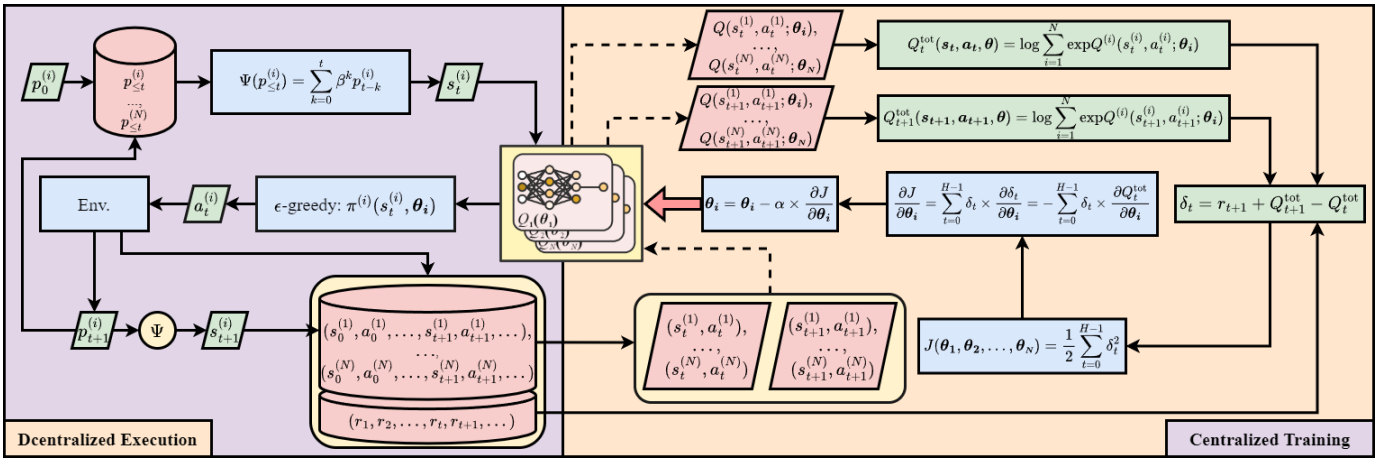


Fig. 2: V2DN's CTDE framework and the abstract algorithm flow process. For the DE module, each robot, *e.g.*, robot i, starts at position $p_0^{(i)}$; calls the sequence encoding function Ψ for state $s_t^{(i)}$; calculates the individual value function $Q^{(i)}$, and decides action $a_t^{(i)}$ with 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}.

sequence ($\boldsymbol{p}_{\leq t}^{(i)}$), which poses difficulties in the parameterization process; (2) the central value function $\left(Q^{\text{tot}}\right)$ construction function needs to be variational and positively correlated with each individual value function.

In this paper, we employ the sequence encoding function (Ψ) to dynamically compress the every-growing position sequence into a fixed length feature vector, *i.e.*,

$$\boldsymbol{s}_t^{(i)} = \Psi(\boldsymbol{p}_{\leq t}^{(i)}), \tag{2}$$

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 [46] for state feature embedding[1]. Given a position sequence, *e.g.*, $\boldsymbol{p}_{\leq t}^{(i)}$, and each position element has been represented with the one-hot embedding scheme, the discounted encoding function embeds the feature vector as follows:

$$\Psi(\boldsymbol{p}_{\leq t}^{(i)}) = \sum_{k=0}^{t} \beta^k \boldsymbol{p}_{t-k}^{(i)}, \tag{3}$$

where $0 < \beta < 1$ is the time-discounted factor, and $p_{t-k}^{(i)}$ 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 multi-layer perceptron (MLP) and denote the related parameters as $\boldsymbol{\theta_i}$. 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:

$$Q_t^{\text{tot}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}) = \log \sum_i \exp Q^{(i)}(\boldsymbol{s}_t^{(i)}, a_t^{(i)}; \boldsymbol{\theta_i}), \tag{4}$$

[1]Note that one may use more complex schemes such as gated recurrent unit (GRU) [47] or transformer [48], for the state feature embedding.

where $\boldsymbol{s}_t$, $\boldsymbol{a}_t$ 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 (RHS) of Eq. (4); and (2) the summation operation in Eq. (4) 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 pre-deployment phase.

With the team reward at time step t represented as r_t, we can get the TD error with respect to Q^{tot} at time step t as:

$$\delta_t = r_{t+1} + Q_{t+1}^{\text{tot}} - Q_t^{\text{tot}}, \tag{5}$$

where δ_t denotes the TD error with respect to the central value function at time step t. Note that $\gamma = 1$ and thus it is omitted in Eq. (5). The objective of V2DN's CT module is to minimize $J(\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \ldots, \boldsymbol{\theta_N}) = \frac{1}{2}\sum_{t=0}^{H-1} \delta_t^2$, where H is the length of the collected multi-robot search trajectory. One may use the chain rule of multi-variate calculus to derive $\partial J / \partial \boldsymbol{\theta_i}$ as follows:

$$\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{\theta_i}} &= \sum_{t=0}^{H-1} \delta_t \times \frac{\partial \delta_t}{\partial \boldsymbol{\theta_i}} \\
&= -\sum_{t=0}^{H-1} \delta_t \times \frac{\partial Q_t^{\text{tot}}}{\partial \boldsymbol{\theta_i}} \\
&= -\sum_{t=0}^{H-1} \delta_t \times \frac{\exp Q_t^{(i)}}{\sum_i \exp Q_t^{(i)}} \times \frac{\partial Q_t^{(i)}}{\partial \boldsymbol{\theta_i}},
\end{aligned} \tag{6}$$

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 temporal difference error rather than the residual error with respect to central value function, we '**freeze**' the parameters of Q_{t+1}^{tot} during the derivative derivation process, and only focus on calculating

the derivative with respect to Q_t^{tot}. This is the reason that Line 2 of Eq. (6) has only one partial derivative term. Freezing parameters is a common scheme used in TD-related learning algorithms, such as deep Q-network (DQN) [49].

With $\partial J/\partial \boldsymbol{\theta_i}$, one may update $\boldsymbol{\theta_i}$ with stochastic gradient descent, *i.e.*,

$$\boldsymbol{\theta_i} = \boldsymbol{\theta_i} - \alpha \times \frac{\partial J}{\partial \boldsymbol{\theta_i}}, \tag{7}$$

where α is the learning rate.

2) Decentralized Execution for V2DN: With the updated parameters for individual value function, each individual robot's policy can be updated and also executed by simply following the ϵ-greedy rule. For instance, when facing a (compressed) state $\boldsymbol{s}^{(i)}$, robot i chooses $a = \arg\max Q^{(i)}\big(\boldsymbol{s}^{(i)}, a; \boldsymbol{\theta_i}\big)$ with probability $1-\epsilon$, and chooses any other available action with equal probabilities, *e.g.*, with the probability $\epsilon/|A(\boldsymbol{s}^{(i)})|$, where $|A(\boldsymbol{s}^{(i)})|$ refers to the total number of available actions at state $\boldsymbol{s}^{(i)}$.

C. Analysis of V2DN

The last subsection introduces V2DN, which comprises the CT module and the DE module. In the CT module, we first utilize the time-discounted embedding scheme [46] 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 with respect to the central value function. In the DE module, each robot simply executes the ϵ-greedy strategy with respect to the pre-trained individual value function. In this subsection, we first validate that V2DN adheres to the R-FAC paradigm, and then displays the pseudo code 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. Secondly, we prove that the partial derivative of Q^{tot} with respect to $Q^{(i)}$ is greater than or equal to zero by introducing and proving the following theorem.

Theorem 1 (The Variational IGM Theorem)**.** The constructed central value function has a strict positive relationship with respect to 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} with respect to $Q^{(i)}$ based on Eq. (4), one can get:

$$\frac{\partial Q^{\text{tot}}}{\partial Q^{(i)}} = \frac{\exp Q^{(i)}}{\sum_i \exp Q^{(i)}} > 0. \tag{8}$$

From Eq. (8), we can see that for whatever team configuration, the partial derivative of Q^{tot} with respect to $Q^{(i)}$ 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 with respect to 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) Pseudo Code 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 pseudo code of V2DN's centralized training process in Algorithm 1, and put the pseudo code for V2DN's decentralized execution process in Algorithm 2 in the Appendix for paper's overall flow smoothness. Note that (1) Algorithm 1 takes, as inputs, the pre-collected 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 with respect to the pre-trained individual value function, and thus we skip the corresponding computational complexity analysis.

In this paper, 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 [50]. To evaluate the computational complexity of an algorithm, we count the total number of flops; express it as a function (usually a polynomial) of the dimensions of the involved matrices and vectors, and simplify the expression by ignoring all terms except for the leading ones. Examining Algorithm 1, we find that the core computational load happens between Line 2 and Line 12, which consists of two loops.

The first loop contains a nested loop from Line 3 to Line 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 $\mathcal{O}(2nd)$. Line 3 to Line 5 loop for N times for the nested loop, resulting in a computational complexity of $\mathcal{O}\big(N \times (n + 2(Hn + 2nd))\big)$, and we can express it as $\mathcal{O}\big(Nn(H+d)\big)$ when dropping the non-leading terms. Similar analysis can be applied to Line 6 and Line 7, which yield the computational complexity at $\mathcal{O}(2N)$

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. (4);
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. (6), *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. (7), *i.e.*, $\boldsymbol{\theta_i} \leftarrow \boldsymbol{\theta_i} - \alpha \times \frac{\partial J}{\partial \boldsymbol{\theta_i}}$;
13 Final.

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 $\mathcal{O}\big(HNn(H+d)\big)$.

The second loop (Line 11 to Line 12) involves the backpropagation 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 Line 11 and Line 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 with respect to the related parameters.

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 multi-robot search environments: MUSEUM and OFFICE. For state-of-the-art MuRES solutions, we select (1) cross entropy regularized policy gradient (CE-PG) [12]; (2) probability density factorization (PD-FAC) [5]; (3) distributional reinforcement learning based searcher (DRL-Searcher) [51]; (4) deep Q-network for multi-robot cooperative search [11], referred to as DQN-Searcher; (5) multi-agent deep deterministic policy gradient (MADDPG) for multi-robot cooperative search [26], referred to as MADDPG-Searcher, and (6) multi-agent soft actor critic (mSAC) [52] twisted for the multi-robot efficient search problem, referred to as mSAC-Searcher. Note that (1) all selected baseline algorithms are most recently proposed multi-robot efficient search 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 inter-agent communication and coordination mechanism prevents us from adapting related algorithms to the *topological* environments used in this paper; (3) optimization-based algorithms, such as finite horizon path enumeration (FHPE) [14] and mixed integer linear programming (MILP) [15] require the full knowledge of target's motion dynamics as well as all robots' position information, which make them vulnerable (non-resilient) to individual failures during task execution. Therefore, we did not select them as the comparative baseline algorithms.

The meta-parameter configurations for selected algorithms as well as V2DN are presented in Table II. All algorithms are (re-)implemented in Python 3.8 with publicly available source code[2], and the experiments are conducted on a server eqipped with an 8 core 16 threads 3.3 GHz CPU, 16GB RAM, NVIDIA RTX3080 and the 64-bit Ubuntu system.

TABLE II: Algorithms' Meta-Parameter Configuration

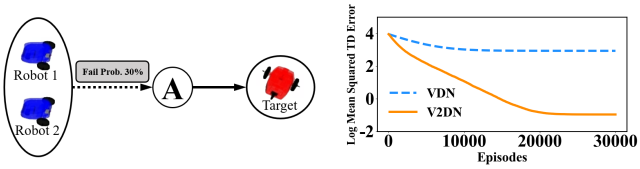
Param	Description	Algorithm	Value
$T_{\max}$	max. # of training epochs	All	3×10^4
α	learning rate	All	0.01
β_i	balance parameter for robot i	CE-PG, mSAC-Searcher, DRL-Searcher	0.5
M	# of bins for value distribution	PD-FAC, DRL-Searcher	51
$V_{\min}$,$V_{\max}$	min./max. of value distribution	PD-FAC,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

In the following several subsections, 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* multi-robot 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 [12] 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 resiliency score (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 multi-robot search environments, namely MUSEUM and OFFICE.

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

A. Central Value Function Construction Scheme Comparison

As mentioned towards 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 multi-robot 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 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.



(a) R-MuRES Example 1 (b) TD-Error Comparison

Fig. 3: The 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: $Q^{(1)} = Q^{(2)} = 0$.

Fig. 3(a) shows a simple resilient multi-robot 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:

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

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:

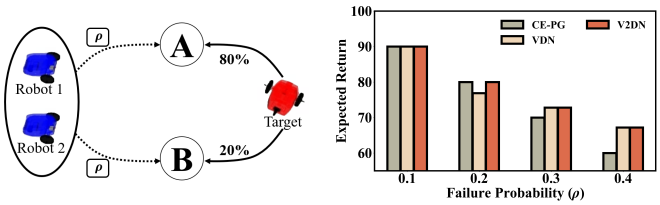
$$Q^{\text{tot}} = \begin{cases} \log\left(\exp Q^{(1)} + \exp Q^{(2)}\right) & \text{w.p.} \quad 49\% \\ \log\left(\exp Q^{(1)}\right) = Q^{(1)} & \text{w.p.} \quad 21\% \\ \log\left(\exp Q^{(2)}\right) = Q^{(2)} & \text{w.p.} \quad 21\% \\ 0 & \text{w.p.} \quad 9\%. \end{cases} \tag{10}$$

Fig. 3(b) compares the convergence process of the mean-squared TD error (represented on a logarithmic scale) between VDN 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 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 subsection, 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 [12], in the *resilient* multi-robot efficient search 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 better performance.



(a) R-MuRES Example 2 (b) Performance Comparison

Fig. 4: Performance Comparison for Different Individual Failure Probabilities. Capture reward: $r_{\text{cap}} = 100$, learning rate: $\alpha = 0.01$; initialization: $Q^{(1)}(A) = Q^{(1)}(B) = Q^{(2)}(A) = Q^{(2)}(B) = 0$; $\epsilon = 0.1$.

Fig. 4(b) shows the performance comparison among V2DN, VDN and CE-PG for different ρ values $\left(\rho \in \{0.1, 0.2, 0.3, 0.4\}\right)$. In the figure, we can see that (1) CE-PG tends to disperse the robots to distinct nodes, due to the

TABLE III: Pre-Deployment resiliency score comparison between V2DN and state of the arts in MUSEUM and OFFICE.

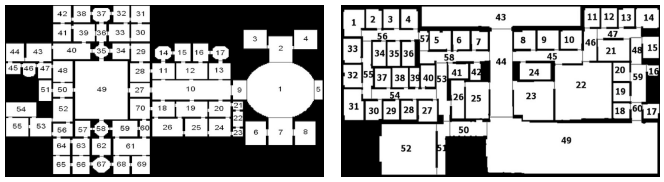
	N	ρ	V2DN	VDN	CE-PG	PD-FAC	DRL-Searcher	DQN-Searcher	MADDPG-Searcher	mSAC-Searcher
MUSEUM	3	0.1	**0.169**	0.240	0.241	0.268	0.237	0.225	0.235	0.282
		0.2	**0.190**	0.218	0.402	0.309	0.318	0.303	0.347	0.394
		0.3	**0.216**	0.328	0.495	0.499	0.406	0.350	0.477	0.527
	4	0.1	**0.147**	0.206	0.201	0.190	0.176	0.183	0.195	0.221
		0.2	**0.173**	0.265	0.343	0.229	0.253	0.257	0.281	0.306
		0.3	**0.227**	0.313	0.390	0.374	0.314	0.304	0.413	0.390
	5	0.1	0.130	0.191	**0.117**	0.146	0.172	0.133	0.141	0.177
		0.2	**0.174**	0.246	0.224	0.182	0.205	0.188	0.212	0.216
		0.3	**0.198**	0.280	0.266	0.279	0.226	0.217	0.246	0.252
OFFICE	3	0.1	**0.202**	0.264	0.336	0.298	0.271	0.214	0.250	0.263
		0.2	**0.239**	0.311	0.445	0.341	0.334	0.329	0.322	0.334
		0.3	**0.280**	0.353	0.505	0.397	0.440	0.376	0.408	0.461
	4	0.1	**0.158**	0.221	0.220	0.246	0.184	0.223	0.239	0.215
		0.2	**0.189**	0.270	0.355	0.299	0.265	0.245	0.256	0.332
		0.3	**0.249**	0.351	0.446	0.383	0.296	0.288	0.319	0.426
	5	0.1	0.103	0.192	**0.097**	0.120	0.144	0.139	0.161	0.162
		0.2	**0.160**	0.230	0.222	0.198	0.195	0.193	0.229	0.231
		0.3	**0.188**	0.253	0.273	0.315	0.229	0.254	0.263	0.290

† **Bold** numbers indicate the best performance for the configuration, and underlined numbers indicate the second best performance.

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.*, $\rho = 0.3$, $\rho = 0.4$).

C. Resiliency Performance Benchmark and Comparison in Canonical Multi-Robot 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 VDN and CE-PG for a range of individual failure probabilities. In this subsection, we further evaluate and compare the resiliency performance of V2DN with state-of-the-art MuRES solutions as well as VDN in two canonical multi-robot search environments, namely MUSEUM and OFFICE, whose layouts are displayed in Fig. 5.



(a) MUSEUM (b) OFFICE

Fig. 5: Canonical multi-robot search testing environments from [14], each room is associated with the corresponding node number. Left: MUSEUM; Right: OFFICE.

We set up the R-MuRES problem in two multi-robot 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 averaged target capture time is deemed as the corresponding algorithm's expected target capture time (μ).

Additionally, since R-MuRES is a newly proposed multi-robot search problem, there exists no specific performance metric in the literature, to indicate the resiliency performance of a multi-robot system for moving target search in the face of individual failures. Therefore, we introduce the following concept (Resiliency Score) to gauge how 'resilient' a multi-robot 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 with respect to 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[3] of the algorithm for the R-MuRES problem as:

$$\eta = \mu / t_{\max} \times 100\%, \tag{11}$$

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 *pre-deployment resilience*, and *in-execution resilience*. For pre-deployment resilience, all robots in the team are subject to a certain failure probability (ρ) only in the pre-deployment stage, and the remaining functioning robots in the execution stage are always

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

TABLE IV: In-Execution resiliency score comparison between V2DN and state of the arts in MUSEUM and OFFICE. **Bold** numbers indicate the best performance for the configuration, and underlined numbers indicate the second best performance.

	N	ρ	V2DN	VDN	CE-PG	PD-FAC	DRL-Searcher	DQN-Searcher	MADDPG-Searcher	mSAC-Searcher
MUSEUM	3	0.04	**0.214**	0.322	0.299	0.321	0.312	0.350	0.303	0.356
		0.06	**0.289**	0.437	0.413	0.543	0.411	0.484	0.389	0.538
		0.08	**0.335**	0.521	0.529	0.667	0.580	0.613	0.493	0.665
	4	0.04	**0.195**	0.311	0.270	0.282	0.286	0.280	0.299	0.305
		0.06	**0.269**	0.373	0.387	0.409	0.382	0.419	0.349	0.481
		0.08	**0.303**	0.547	0.489	0.519	0.473	0.501	0.458	0.608
	5	0.04	**0.178**	0.288	0.245	0.194	0.248	0.255	0.249	0.257
		0.06	**0.230**	0.294	0.348	0.297	0.329	0.318	0.303	0.395
		0.08	**0.285**	0.475	0.423	0.472	0.373	0.426	0.415	0.545
OFFICE	3	0.04	0.233	0.328	**0.229**	0.306	0.333	0.357	0.299	0.378
		0.06	**0.284**	0.443	0.471	0.513	0.454	0.459	0.401	0.536
		0.08	**0.364**	0.566	0.591	0.662	0.544	0.541	0.581	0.723
	4	0.04	**0.217**	0.283	0.218	0.274	0.305	0.302	0.286	0.307
		0.06	**0.257**	0.336	0.430	0.458	0.417	0.373	0.319	0.477
		0.08	**0.315**	0.459	0.542	0.579	0.511	0.527	0.562	0.695
	5	0.04	0.183	0.225	**0.163**	0.215	0.196	0.245	0.203	0.256
		0.06	**0.216**	0.302	0.309	0.364	0.325	0.311	0.284	0.445
		0.08	**0.264**	0.361	0.362	0.473	0.352	0.457	0.376	0.615

operational until the search task is completed. For in-execution resilience, besides the pre-deployment failure probability, all robots are also subject to a certain failure probability at each execution step.

1) Pre-Deployment Resiliency Score Evaluation and Comparison: Table III presents the pre-deployment 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 [12], which states that 5 functional robots are enough to perform the multi-robot 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.

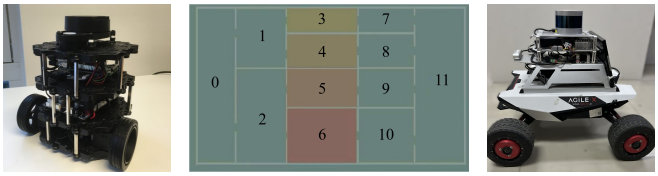
2) In-Execution Resiliency Score Evaluation and Comparison: Table IV displays the in-execution resiliency score comparison for different configuration parameters, *i.e.*, the number of robots (N), the failure probability at each step (ρ). Note that for the in-execution resiliency score evaluation, since 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 incorporates the individual robot's failure into consideration, 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, as 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.

VI. System Integration and Experimental Results

The last section evaluates and benchmarks V2DN's performance against state-of-the-art multi-robot search algorithms in two canonical simulation environments. In this section, we deploy V2DN to a real multi-robot 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. 6(c), is a four-wheel differential drive SCOUT MINI pro robot, equipped with Velodyne's Puck LiDAR sensor (VLP-16) and Intel RealSense camera for object detection. We integrate our previously developed mapping and localization module, namely E-LOAM [53], as well as the canonical motion planning and obstacle avoidance modules into the robot testbed. On the other hand, the non-adversarially moving target is Turtlebot3 Burger, as shown in Fig. 6(a). The self-constructed indoor environment, as shown in Fig. 6(b), mimics a 12-room housing environment, with four SCOUT MINI pro 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 ending up in different rooms, with



(a) Turtlebot3 (b) Testing Environment (c) SCOUT MINI

Fig. 6: The experiment setup: Fig. 6(a) is a Turtlebot3 Burger, which acts as the moving target; Fig. 6(c) is a SCOUT MINI pro, used as the robot testbed, and Fig. 6(b) shows the multi-robot search testing environment, which is a 12-room housing environment. The target starts at Room 0, and the robot team are initialized at Room 11.

the yellower the lower probability and the redder the higher probability.

While a demonstrative video has been uploaded together with the manuscript, Fig. 7 shows one snapshot of the integrated multi-robot search process, where the top left figure visualizes the status information from Robot Operating System (ROS) perspective; four bottom left figures show all individual robots' local vision information and the right half figure displays the bird's-eye-view of the whole operational system.

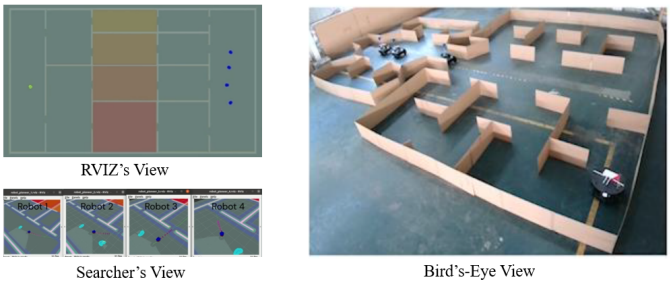
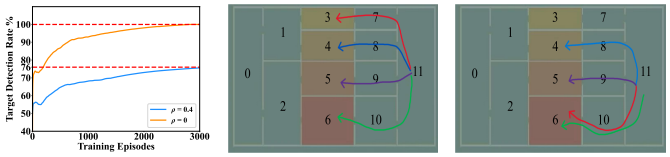


Fig. 7: Snapshot of the resilient multi-robot search process

We evaluate two use cases of the pre-deployment 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. 8 shows the mean learning curve comparison as well as the correspondingly sketched final multi-robot search paths for the two use cases. In the figure, we can see that (1) when $\rho = 0.0$ (Fig. 8(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. 8(a)); (2) when $\rho = 0.4$ (Fig. 8(c)), since the robots are subject to a high faulty probability, V2DN is able to conglomerate two of the robots to Room 6, 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. 8(a)). Both quantitative results are evaluated with 100 independent trials, and also reported in the uploaded demonstrative video.

VII. Conclusion and Future Work

This paper introduces a novel multi-robot search problem, namely *resilient* multi-robot efficient search (R-MuRES),



(a) Learning Curve (b) $\rho = 0.0$ (c) $\rho = 0.4$

Fig. 8: V2DN's learning curve comparison and final multi-robot search paths illustration for different pre-deployment faulty probabilities (ρ).

whose unique characteristic is that the individual robots within the team might malfunction and quit the team during task execution. We then propose the resilient value function factorization (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 variational value decomposition network (V2DN) is proposed and validated with both numerical simulations in canonical multi-robot search environments and physical experiments on a real multi-robot 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) different robots have different sensing ranges and/or different motion capabilities, *i.e.*, the heterogeneous coordination between drones and ground robots for advanced moving target search.

References

[1] 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*. Springer, 2020, pp. 133–146.

[2] L.-G. Pan, Q. Lu, X. Xie, J. Wang, and J. Wang, "A probability distribution based cooperative search approach for stochastic source localization," in *IEEE International Symposium on Industrial Electronics (ISIE)*, 2018, pp. 585–590.

[3] 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 *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7393–7399.

[4] G. Hollinger, A. Kehagias, and S. Singh, "GSST: anytime guaranteed search," *Autonomous Robots*, vol. 29, no. 1, pp. 99 – 118, July 2010.

[5] 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 Robotics and Automation Letters (RA-L)*, vol. 7, no. 4, pp. 8869–8876, 2022.

[6] R. Hu, N. Tan, and F. Ni, "A new scheme for cooperative hunting tasks with multiple targets in dynamic environments," in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2021, pp. 1816–1822.

[7] A. Asgharnia, H. M. Schwartz, and M. Atia, "Multi-invader multi-defender differential game using reinforcement learning," in *2022 IEEE International Conference on Fuzzy Systems*, 2022, pp. 1–8.

[8] C. de Souza, R. Newbury, A. Cosgun, P. Castillo, B. Vidolov, and D. Kulić, "Decentralized multi-agent pursuit using deep reinforcement learning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 3, pp. 4552–4559, 2021.

[9] A. Goldhoorn, A. Garrell, R. Alquézar, and A. Sanfeliu, "Searching and tracking people with cooperative mobile robots," *Autonomous Robots (AR)*, vol. 42, no. 4, p. 739–759, 2018.

[10] F. Yan, K. Di, J. Jiang, Y. Jiang, and H. Fan, "Efficient decision-making for multiagent target searching and occupancy in an unknown environment," *Robotics and Autonomous Systems (RAS)*, vol. 114, pp. 41–56, 2019.

[11] X. Qin, X. Li, Y. Liu, R. Zhou, and J. Xie, "Multi-agent cooperative target search based on reinforcement learning," *Journal of Physics: Conference Series*, vol. 1549, no. 2, p. 022104, 2020.

[12] H. Guo, Z. Liu, R. Shi, W.-Y. Yau, and D. Rus, "Cross-Entropy regularized policy gradient for multirobot nonadversarial moving target search," *IEEE Transactions on Robotics (T-RO)*, pp. 1–16, 2023.

[13] H. Lau, S. Huang, and G. Dissanayake, "Probabilistic search for a moving target in an indoor environment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 3393–3398.

[14] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias, "Efficient multi-robot search for a moving target," *The International Journal of Robotics Research (IJRR)*, vol. 28, no. 2, pp. 201–219, 2009.

[15] B. A. Asfora, J. Banfi, and M. Campbell, "Mixed-integer linear programming models for multi-robot non-adversarial search," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 4, pp. 6805–6812, 2020.

[16] J. Berger, N. Lo, and M. Barkaoui, "Static target search path planning optimization with heterogeneous agents," *Annals of Operations Research*, 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," *Journal of Natural Gas Science and Engineering*, vol. 84, p. 103636, 2020.

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

[19] R. Ravichandran, D. Ghose, and K. Das, "UAV based survivor search during floods," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 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," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 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 *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8808–8814.

[22] I. Vandermeulen, R. Groß, and A. Kolling, "Sampling-based search for a semi-cooperative target," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 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 *Proceedings of IEEE International Conference on Unmanned Systems (ICUS)*, 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. 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 *IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 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] H. Osooli, P. Robinette, K. Jerath, and R. Ahmadzadeh, "A multi-robot task assignment framework for search and rescue with heterogeneous teams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Advances in Multi-Agent Learning-Coordination, Perception, and Control Workshop*, 2023, pp. 1–6.

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

[29] J. T. Ebert, F. Berlinger, B. Haghighat, and R. Nagpal, "A hybrid PSO algorithm for multi-robot target search and decision awareness," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 11 520–11 527.

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

[31] M. Dadgar, M. S. Couceiro, and A. Hamzeh, "RbRDPSO: Repulsion-based RDPSO for robotic target searching," *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, vol. 44, pp. 551–563, 2020.

[32] Q. Yang and R. Parasuraman, "Game-theoretic utility tree for multi-robot cooperative pursuit strategy," in *54th International Symposium on Robotics, ISR Europe 2022*, 2022, pp. 278–284.

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

[34] A. Oroojlooy and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *Applied Intelligence*, 2022.

[35] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 895–943, 2022.

[36] 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 *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 4295–4304.

[37] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018, pp. 2085–2087.

[38] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[39] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "QPLEX: Duplex dueling multi-agent Q-learning," in *International Conference on Learning Representations (ICLR)*, 2021.

[40] 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 *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, 2019, pp. 5887–5896.

[41] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 2017-December, 2017, pp. 6380–6391.

[42] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *32nd AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2974–2982, 2018.

[43] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. WU, "The surprising effectiveness of PPO in cooperative multi-agent games," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 24 611–24 624.

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

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

[46] 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.

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

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of International Conference on Neural Information Processing Systems (NeurIPS)*. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

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

[50] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[51] H. Guo, Q. Peng, Z. Cao, and Y. Jin, "DRL-Searcher: A unified approach to multirobot efficient search for a moving target," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, pp. 1–14, 2023.

[52] Z. He, L. Dong, C. Song, and C. Sun, "Multiagent soft actor-critic based hybrid motion planner for mobile robots," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, pp. 1–13, 2022.

[53] H. Guo, J. Zhu, and Y. Chen, "E-LOAM: LiDAR odometry and mapping with expanded local structural information," *IEEE Transactions on Intelligent Vehicles (T-IV)*, vol. 8, no. 2, pp. 1911–1921, 2023.

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
FSM	finite state machine
GRU	gated recurrent unit
IGM	individual global maximum
MADDPG	multi-agent deep deterministic policy gradient
MARL	multi-agent reinforcement learning
MILP	mixed integer linear programming
MLP	multi-layer perceptron
mSAC	multi-agent soft actor critic
MuRES	multi-robot efficient search
MuRGS	multi-robot guaranteed search
MuRRS	multi-robot reliable search
PD-FAC	probability density factorization
R-FAC	resilient value function factorization
R-MuRES	resilient multi-robot efficient search
ROS	Robot Operating System
RS	resiliency score
TD	temporal difference
V2DN	variational value decomposition network
VDN	value decomposition network
VFF	value function factorization
w.p.	with probability
w.r.t.	with respect to

Appendix

Algorithm 2: V2DN's Decentralized Execution

Input: Graph $\mathcal{G}$; Number of Pre-Deployed robots N; Individual value function network $Q^{(i)}(\boldsymbol{\theta_i})$; Per-step failure rate: $\rho^{(i)}$; max. search time step $T_{\max}$; Target motion model $\mathbf{\Gamma}$; Sequence encoding function Ψ;

Output: 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 $(r_1, r_2, \ldots, r_H)$

Init: Each robot initial position: $p_0^{(i)}$; Each robot's malfunction indicator, $\text{flag}_i = \text{False}$; Target initial position e_0; $t \leftarrow 0$; $\boldsymbol{p}_{\leq t}^{(i)} \leftarrow p_0^{(i)}$;

while $t \leq T_{\max} - 1$ **do**
  foreach $i \in \{1, 2, \ldots, N\}$ **do**
    if *flag*$_i$ = *False* **then**
      Calculate state vector with embedding function: $s_t^{(i)} = \Psi(\boldsymbol{p}_{\leq t}^{(i)})$;
      Choose action $a_t^{(i)}$ based on ϵ-greedy policy w.r.t. $Q^{(i)}(\boldsymbol{\theta_i})$;
      Execute action $a_t^{(i)}$ and get $p_{t+1}^{(i)}$;
      Update Robot i's functional status:
      $$\text{flag}_i = \begin{cases} \text{True} & \text{with probability } \rho^{(i)} \\ \text{False} & \text{otherwise} \end{cases}$$
    else
      $p_{t+1}^{(i)} = \varnothing$ /* Robot i malfunctions and quits the team. */
  $e_{t+1} = \mathbf{\Gamma}(e_t)$; $r_{t+1} = R(\boldsymbol{p}_{t+1}, e_{t+1})$;
  $\forall i$, store $(p_t^{(i)}, a_t^{(i)})$ into database;
  store r_{t+1} into database;
  foreach $i \in \{1, 2, \ldots, N\}$ **do**
    if $(p_{t+1}^{(i)} = e_{t+1}$ & *flag*$_i$ = *False*$)$ **then**
      /* Target is captured. */
      return;
    else if $\forall i \in \{1, 2, \ldots, N\}$, *flag*$_i$ = *True* **then**
      /* All robots malfunction. */
      return;
    else
      $\boldsymbol{p}_{\leq t+1}^{(i)} \leftarrow \boldsymbol{p}_{\leq t}^{(i)} \cup p_{t+1}^{(i)}$;
      $t \leftarrow t + 1$;
Final.