Self-Adaptive Swarm System

by

Qin Yang

(Under the Direction of Ramviyas Nattanmai Parasuraman)

Abstract

Multi-agent systems (MAS) could play a pivotal role in realizing future intelligent workspaces, especially in building so-called artificial social systems, such as self-driving cars and multi-robot systems (MRS). For example, MAS/MRS cooperates to increase mission performance in many applications, including exploration, surveillance, defense, humanitarian, and emergency missions like urban search and rescue. In such missions, complex environments such as hazardous, dynamic changing, and adversarial surroundings create a significant challenge to the agents in realizing their full potential. Therefore, this thesis addresses some pressing gaps in the literature in realizing an adaptive MAS by proposing a principled MAS cooperation framework, termed the Self-Adaptive Swarm System (SASS), which bridges communication, planning, decision-making and learning in the distributed MAS.

In particular, the core scientific contributions of this thesis are as follows: 1) we define a novel human-inspired agent (robot) needs hierarchy model to consider an agent's motivation and requirements based on the current status and assigned tasks; 2) we present a priority-based distributed negotiation-agreement algorithm for realizing multi-agent tasks assignment problems, effectively avoiding plan conflicts – Here, we decompose the tasks into atomic operations and achieve MAS cooperation through a series of simple sub-tasks; 3) we introduce a new needs-based agent trust and cooperation mechanism to create needs-driven relationships among multiple agents in challenging environments; 4) we build a new hierarchical utility tree to realize game-theoretic solutions for the cooperating MAS in the presence of adversarial opponent agents; 5) we propose a novel Bayesian strategy networks (BSN) applied to deep reinforcement learning by decomposing tasks into multiple sub-level actions and obtaining the optimal agent policies in unknown and challenging environments.

Index words: [Multi-Agent Systems, Multi-Robot Systems, Bayesian Network, Deep Reinforcement Learning]

Self-Adaptive Swarm System

by

Qin Yang

B.S., Harbin Institute of Technology, China, 2004
M.E., Peking University, China, 2011
M.S., Colorado School of Mines, 2018

A Dissertation Submitted to the Graduate Faculty of the University of Georgia in Partial Fulfillment of the Requirements for the Degree.

Doctor of Philosophy

Athens, Georgia

©2022
Qin Yang
All Rights Reserved

Self-Adaptive Swarm System

by

Qin Yang

Major Professor:	Ramviyas Nattanmai Parasuraman
Committee:	Fred Maier Maria Hybinette

Electronic Version Approved:

Ron Walcott
Dean of the Graduate School
The University of Georgia
May 2022

Dedication

To my parents
for endowing me strength and courage
letting me pursue my dream
for so long
so far away from home
&
To my friends
for supporting my
new ideas guiding me to pursue

Acknowledgments

I would like to thank my advisor, Ramviyas Nattanmai Parasuraman, for supporting me over the years, and for giving me so much freedom to explore and discover new areas of multi-agent/robot systems (MAS/MRS). My other committee members have also been very supportive. Fred Maier has long been an inspiration to me. His classes and weekly meetings have proved to be one of my best learning experiences at UGA. Maria Hybinette proved to be a most thorough reviewer, as I expected, and has kept me honest about all the details. I also thank my lab for supporting my research in many ways.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Distributed intelligence is typically the result of *self-organization* in complex system such as distributed coordination Heylighen, 2017. It refers to systems of entities working together to reason, plan, solve problems, think abstractly, comprehend ideas and language, and learn Parker, 2007. In the natural systems, entities are characterized by apparently complex behaviors that emerge as a result of often nonlinear spatiotemporal interactions among a large number of components at different levels of organization Levin, 1998.

In the artificial systems, distributed artificial intelligence (DAI) has developed more than three decades as a subfield of artificial intelligence (AI). It has been divided into two sub-disciplines: Distributed Problem Solving (DPS) focuses on the information management aspects of systems with several branches working together towards a common goal; Multi-Agent Systems (MAS) deals with behavior management in collections of several independent entities, or agents Stone and Veloso, 2000.

As a relatively new area studied from 1980 in computer science, MAS rapid growth has been spurred at least in part by the belief that agents are an appropriate software paradigm through which to exploit the possibilities presented by massive open distributed systems – such as the Internet Wooldridge, 2009. Although MAS plays a pivotal role in the potential of the Internet exploration, there are a lot of research fields related to it, especially building so-called *artificial social systems*, such as self-driving car and Multi-Robot Systems (MRS). In these systems, intelligent entities act flexibly. Their behaviors reactively and deliberatively adapt to various circumstances based on processes such as planning, decision-making, learning, and constraint satisfaction. As autonomous entities, agents have far-reaching control over their behavior within the frame of their objectives, possess decision authority in a wide variety of circumstances, and are able to handle complex and unforeseen situations on their own and without the intervention of humans or other systems Weiss, 1999.

Take urban search and rescue (USAR) missions as an example. Rescue missions can be regarded as life-saving, delivering valuable properties, and tackling necessary facilities in disaster or emergency scenarios, including complex, hazardous, uncertain, unstructured, dynamical changing, and adversarial environments. MRS working in such situations requires rapid response, high adaptation, and strong robustness, reducing the losses in the post-disaster scenarios. Research in MRS-aided USAR aims to increase the mission success rate, improve execution efficiency, and minimize system cost during the rescue missions Q. Yang and Parasuraman, 2020c. In this example, it is essential to understand how to combine a team of mobile robots to achieve a successful search and rescue mission, especially from a heterogeneity point of view and through needs-driven cooperation among robots Q. Yang and Parasuraman, 2020b. Therefore, to address the problem, it is a natural choice to build a MAS distributed architecture organizing individuals' behaviors, and avoiding conflicts in heterogeneous agents' cooperation.

Besides, the relationships between agents play a critical role in MAS cooperation for achieving a specific task. They can help agents build various formations, shapes, and patterns presenting corresponding functions and purposes to adapt to different situations. The relationships would also dynamically change due to the transformation of entities' needs and situations during their interaction. On the other hand, the specific trust level between agents is an essential factor in evaluating their relationships' stability, much as people do. For artificial intelligent agents, especially MRS and Human-Robot Interaction (HRI), trust is also a significant factor that needs to be considered when robots will work as teams or teammates in human-robot teams, used as autonomous agents, or where robots will be used in complex and dangerous situations Khavas et al., 2020.

Generally speaking, the MAS models can be described as three levels: *planning and control*, *decision-making*, and *learning*. For low-level planning and control, it has been well studied in control and optimization theory. Especially in MRS, it allows task-dependent dynamic reconfiguration into a team is among the grand challenges in Robotics G.-Z. Yang et al., 2018, necessitating the research at the intersection of communication, control, and perception. Currently, planning-based approaches combined with star-shaped communication models can not generally scale or handle a large number of agents in a distributed or decentralized manner Desaraju and How, 2012.

Apart from the low-level planning and control in MAS cooperation from automation perspectives, another major area – *decision-making in uncertainty* – involves many interdisciplinary fields, such as game theory, machine learning, and probabilistic graphical models (PGM). More specifically, uncertainty can arise from incomplete information about the environments and practical and theoretical limitations in our ability to predict future events Kochenderfer, 2015. In other case, as in decision under uncertainty, holistic preferences may be represented in terms of utilities for consequences and probabilities for consequences or for "states of the world" Fishburn, 1970. Considering working in adversarial environments, opponents can prevent MAS from achieving global and local tasks, even impair individual or system necessary capabilities or normal functions Jun and D'Andrea, 2003. So the *utility* is critical to determine whether or not an individual can calculate reasonable strategies to help the system balance and optimize the performance between individual and group for MAS cooperation in adversarial environments Q. Yang et al., 2019. Effective MAS cooperative strategies can maximize global system utility and guarantee sustainable development for each group member Shen et al., 2004.

Furthermore, the highest-level needs – *learning* – for MAS, individuals upgrading by interaction based on their learned experiences would lead to the self-evolution of the whole system. More specifically, organized individual actions can help the system presenting more complicated and robust structures, functions, and behaviors to adapt to the uncertain environment. By adapting learning based on the agent's needs and interactive experience, individuals will orchestrate behaviors, strategies, resources, and learning activities between agent-agent and agent-environment to effectively optimize or sub-optimize the group utilities. It involves building appropriate inference and predictive models, designing appropriate utility functions, estimating reasonable parameters fitting the specific scenario, and so forth.

1.1 Challenges

Currently, designing distributed MAS and MRS cooperation solutions suffers from the following key challenges.

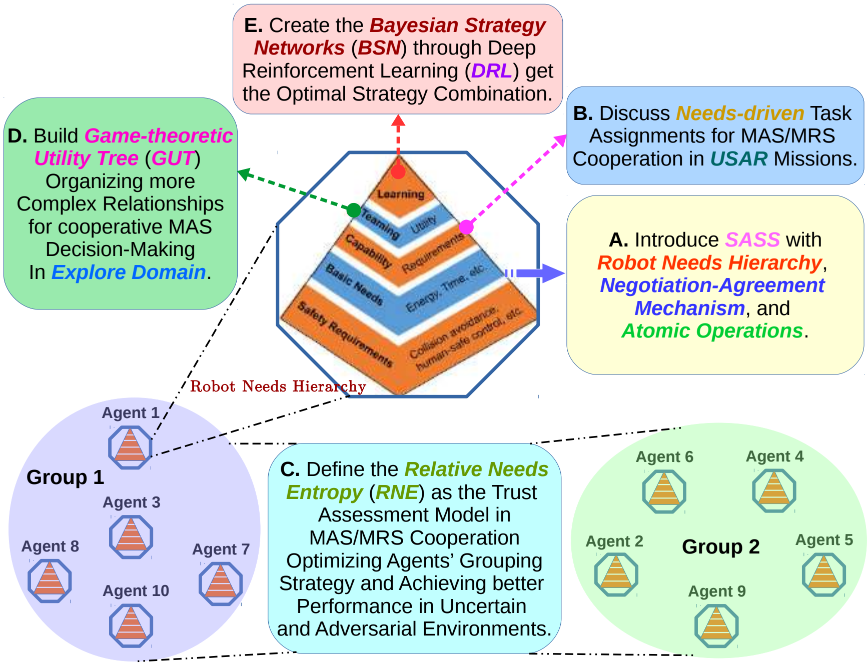
1. *Planning and Control*: For low-level MAS planning and control, Y.Rizk Rizk et al., 2019 groups the system based on the cooperative tasks' complexity as four levels of automation. In the first level (least automated), only task execution is automated, while the second level also automates either task allocation or coalition formation but not both. The third level automates both coalition formation and task allocation but does

not automate task decomposition. The fourth level automates the entire system. Currently, no references were found that automated the entire process (fourth level of automation).

2. *Decision-Making*: For high-level MAS decision-making, recent studies mainly concern partial cooperation and do not consider deeper cooperative relationships among agents representing more complex team strategies. There are still some of the remaining challenges: 1) Scalability; 2) Computational Complexity; 3) Dynamic Environments; 4) System Heterogeneity; 5) Big Data; 6) Evaluation Standards; 7) Task Complexity Rizk et al., 2018.

4. *Agent Trust Model*: Although modeling trust has been studied from various perspectives and involves many different factors, it is still challenging to develop a conclusive trust model that incorporates all of these factors. Therefore, future research into trust modeling needs to be more focused on developing general trust models based on measures other than the countless factors affecting trust Khavas et al., 2020.

5. *Learning*: For the highest level learning, it is fundamentally more difficult than the single-agent case due to the presence of multi-agent pathologies, e.g., the moving target problem (non-stationarity), curse of dimensionality, multi-agent credit assignment, global exploration, and relative overgeneralization Hernandez-Leal et al., 2019. However, MAS learning a complex task from scratch is impractical due to the huge sample complexity of RL algorithms. A potential use of *Curriculum* learning in MAS could be, for example, in adversarial environments, training how to play a game against simulated opponents who become progressively more competent, and then reuse the gathered knowledge to face a real-world highly specialized opponent Da Silva and Costa, 2019. Combining the information from perceiving the environments and inferring the corresponding strategies and the world's conditional (or even causal) relations in those scenarios, *Bayesian deep learning* has emerged as a unified probabilistic framework to tightly integrate deep learning and Bayesian models H. Wang and Yeung, 2020. Nevertheless, we still have challenges designing the efficient Bayesian formulation of neural networks with reasonable time complexity and efficient and effective information exchange between the perception component and the task-specific component.



E. Create the ***Bayesian Strategy Networks*** (***BSN***) through Deep Reinforcement Learning (***DRL***) get the Optimal Strategy Combination.

D. Build ***Game-theoretic Utility Tree*** (***GUT***) Organizing more Complex Relationships for cooperative MAS Decision-Making In ***Explore Domain***.

B. Discuss ***Needs-driven*** Task Assignments for MAS/MRS Cooperation in ***USAR*** Missions.

A. Introduce ***SASS*** with ***Robot Needs Hierarchy***, ***Negotiation-Agreement Mechanism***, and ***Atomic Operations***.

C. Define the ***Relative Needs Entropy*** (***RNE***) as the Trust Assessment Model in MAS/MRS Cooperation Optimizing Agents' Grouping Strategy and Achieving better Performance in Uncertain and Adversarial Environments.

Figure 1.1: SASS Dissertation Workflow

1.2 Contributions

In this thesis, we first propose a principled MAS cooperation framework, *Self-Adaptive Swarm System (SASS)* Q. Yang, 2021 to bridge the fourth level automation gap between perception, communication, planning, execution, decision-making, and learning Rizk et al., 2019. Following the general framework, we propose *Agent (robot)*[1] *Needs Hierarchy* to model the intelligent agent's motivation and offer a priority queue in a distributed *Negotiation-Agreement Mechanism* avoiding plan conflicts effectively. Then, we provide several *Atomic Operations* to decompose the complex tasks into a series of simple sub-tasks. Furthermore, we introduce a network model called *Game-theoretic Utility Tree (GUT)* mimicking the intelligent agent decision-making process for MAS cooperation in uncertain environments, especially in adversarial scenarios. Besides, to analyze the reliability and stability of relationships between agents or groups in MAS cooperation, we define the needs-based agent trust model – *Relative Needs Entropy (RNE)* – to improve the decision-making and learning performance in MAS cooperation. Moreover, we provide the *Bayesian Strategy Networks (BSN)*

[1] In this thesis, we use the terms agent and robot interchangeably.

based Deep Reinforcement Learning (DRL) to organize agents' behaviors and optimize their strategies efficiently and balance the utility between individual and group reasonably to fulfill agents' the highest level needs learning helping *SASS* self-evolution. Figure 1.1 describes the workflow of this dissertation as corresponding modules. To evaluate our models, we propose the *Explore Domain* and a new application in urban search and rescue (USAR) missions. Specifically, we briefly introduce these contributions as follow:

- **Self-Adaptive Swarm System (SASS)**: The *SASS* framework Q. Yang and Parasuraman, 2020b for cooperation heterogeneous MAS in dynamic task assignments and automated planning combines artificial intelligent agent perception, communication, planning, execution, decision-making, and learning in cooperation. It considers individual agents' needs and action plans and emphasizes the complex relationships created through communication between the agents. Specifically, we proposed *Agent (robot) Needs Hierarchy* to model the agent's motivation and offer a priority queue in a distributed *Negotiation-Agreement Mechanism* avoiding plan conflicts effectively. Furthermore, we provide several *Atomic Operations* to decompose the complex tasks into a series of simple sub-tasks. The details are outlined as follow:

  – *Agent (robot) Needs Hierarchy*: To model an individual robot's *motivation* and *needs* in the negotiation process, we introduce the prioritization technique inspired by Maslow's hierarchy of human psychological needs Maslow, 1943 and solve the conflicts associated with the sub-tasks/elements in task planning. We define the *Robot Need Hierarchy* at five different levels: safety needs (collision avoidance, human-safety control, etc.); basic needs (energy, time constraints, etc.); capability (heterogeneity, hardware differences, communication, etc.); team cooperation (global utility, team performance, cooperation, and global behaviors); and self-upgrade (learning).

  – *Negotiation-Agreement Mechanism*: We propose a distributed Negotiation Agreement mechanism for selection (task assignment), formation (shape control), and routing (path planning) through automated planning of state-action sequences, helping MAS solve the conflicts in cooperation efficiently.

  – *Atomic Operations*: By decomposing the complex tasks into a series of simple sub-tasks, we can recursively achieve those sub-tasks until MAS completes the high-level task. We provide several *Atomic*

Operations for the swarm behaviors: *Selection*, *Formation*, and *Routing*, which allows MAS decomposing a particular plan as flocking, pattern formation, and route planning under the same framework.

- **Relative Needs Entropy (RNE)**: From artificial intelligent agents (like robots) needs perspective, trust can be defined as the difference or distance of needs distribution between agents in a specific scenario for an individual or groups. From a statistical perspective, it can be regarded as calculating the similarity of high-dimensional sample sets – *Relative Entropy* Cover and Thomas, 1991 – we called *Relative Needs Entropy (RNE)*, which represents the reliability and stability of the relationships between agents in MAS cooperation Q. Yang and Parasuraman, 2021.

- **Game-theoretic Utility Tree (GUT)**: *GUT* Q. Yang and Parasuraman, 2020a is a new network model for artificial intelligent agent decision-making or MAS achieving cooperative decisions in uncertain environments, especially in adversarial scenarios. It builds a hierarchical relationship between individual behaviors and interactive entities and helps agents represent more complex behaviors to adapt to various scenarios. *GUT* consists of *Game-theoretic Utility Computation Units* distributed in multiple levels by decomposing strategies, thereby significantly lowering the game-theoretic operations in strategy space dimension. It combines the core principles of *Bayesian Networks* Koller and Friedman, 2009, *Game Theory* Myerson, 2013, and *Utility Theory* Fishburn, 1970; Kochenderfer, 2015. Moreover, the payoff (utility) values in the *Game-theoretic Utility Computation Units* are calculated through the agent needs expectations organized hierarchically following the *Agent (robot) Needs Hierarchy*. As a comprehensive artificial intelligent architecture, *GUT* not only considers the data from perceiving the environments with different "senses" (e.g., vision and hearing) but also infer the world's conditional (or even causal) relations and corresponding uncertainty with the organized hierarchical network.

- **Bayesian Strategy Networks (BSN) in Deep Reinforcement Learning (DRL)**:

  Adopting reasonable strategies is crucial for an intelligent agent with limited resources working in hazardous, unstructured, and dynamic changing environments to improve the system utility, decrease the overall cost, and increase mission success probability. Especially in the application domains such as exploration, disaster rescue, and emergency scenarios, the agent needs to dynamically switch its strategies based on the priority of its needs and adapt to various situations. Organizing the agent's behaviors and actions to represent complex strategies is challenging to learn policies through interaction in *Deep Reinforcement Learning (DRL)*. By decomposing an intricate policy into several simple sub-policies and organizing their relationships through *Bayesian Strategy Networks (BSN)*, it can form a joint policy and build a corresponding DRL model training agents to adapt to various scenarios.

- **Applicaiton domain used in this thesis**:

  - *Explore Domain*: From the realistic and practical perspective, we design a new game domain called *Explore Domain* using which we can analyze how to organize more complex relationships and behaviors in MAS cooperation, achieving given tasks with higher success probability and lower costs in uncertain environments. In this domain, there are multiple explorer agents that need to cooperatively execute assigned tasks while overcoming the adversarial opponent agents in the environment
  - *Urban Search And Rescue (USAR)*: Through organizing heterogeneous agents' capabilities and needs for MAS cooperation in USAR, we consider the *Group's Utility* (teaming needs) as the number of lives (victims) or valuable properties saved and rescued as much as possible in a limited time. In the entire process, agents need to consider exploring the uncertain areas, tackling the *unintentional adversaries* like obstacles, wind, rain, and repairing necessary facilities, treating injurers, carrying victims and properties to a safe place.

Agent (robot) Needs Hierarchy, *Negotiation-Agreement Mechanism*, *Atomic Operations*, *GUT*, *RNE*, and *BSN Deep Reinforcement Learning* can be regarded as different modules focusing on the corresponding perspective integrating with *SASS*. The *Explore Domain* and *urban search and rescue (USAR)* mission provide the new evaluation approach and application area for *SASS* correspondingly. *Agent (robot) Needs Hierarchy* is the foundation in *SASS*, which surveys the system's utility from the perspective of individual needs. Balancing the rewards between agents and groups for MAS through interaction and adaptation in cooperation optimizes the global system's utility and guarantees sustainable development for each group member, much like human society does.

From modeling the MAS perspective, the planning and control govern the individual low-level *safety* and *basic* needs; *capability* and *teaming* needs are the preconditions and requirements of MAS cooperation in decision-making; for the highest level needs learning, individuals fulfill and upgrade themselves from interaction and adaptation helping *SASS* self-evolution.

1.3 Organization

The rest of the thesis is organized as follows: Chapter 2 introduces background on *cooperative multi-agent/robot systems* (MAS/MRS), *utility theory*, *game theory*, *information theory*, *probabilistic graphical models*, and *deep reinforcement learning* (DRL). Following the background discussion, Chapter 3 presents the proposed *SASS* framework based on the *Agent (robot) Needs Hierarchy* integrating perception, communication, planning and control, decision-making, and learning. Then, Chapter 4 discusses the hierarchical needs-driven model and its application in robot-aided urban search and rescue (USAR) missions. To achieve better cooperation in MAS/MRS, the Chapter 5 proposed a general agent trust model based on *relative needs entropy (RNE)* to measure and analyze the trust levels between agents and groups. Considering MAS/MRS cooperative working in adversarial or dangerous environments, we introduce a new *game-theoretic utility tree (GUT)* for multi-agent decision-making in Chapter 6. Moreover, Chapter 7 discusses the hierarchical needs-driven systems learning and proposes a novel agent strategy composition approach termed *Bayesian strategy network (BSN)* for achieving efficient deep reinforcement learning (DRL) methods. Finally, the Chapter 8 concludes the thesis and proposes some future research directions.

1.4 List of Publications

This thesis has resulted in the following publications:

Chapter 3

• **Qin Yang**, Zhiwei Luo, Wenzhan Song, and R. Parasuraman. "Self-Reactive Planning of Multi-Robots with Dynamic Task Assignments". in IEEE International Symposium on Multi-Robot and Multi-Agent Systems (MRS) 2019.

• **Qin Yang** and R. Parasuraman. "Hierarchical needs based self-adaptive framework for cooperative multi-robot system". in IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2020.

• **Qin Yang**. Self-adaptive swarm system (sass). The Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-2021, Doctoral Consortium.

Chapter 4

• **Qin Yang** and R. Parasuraman. "Needs-driven heterogeneous multi-robot cooperation in rescue missions". in IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2020.

Chapter 5

• **Qin Yang** and R. Parasuraman. "How can robots trust each other for better cooperation? a relative needs entropy based robot-robot trust assessment model." in IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2021.

Chapter 6

• **Qin Yang** and R. Parasuraman. "Game-theoretic Utility Tree for Multi-Robot Cooperative Pursuit Strategy." in the 54^{th} International Symposium on Robotics (Europe 2022).

• **Qin Yang** and R. Parasuraman. "A Hierarchical Game-Theoretic Decision-Making for Cooperative Multi-agent Systems Under the Presence of Adversarial Agents." submitted 2022.

Chapter 7

• **Qin Yang** and R. Parasuraman. "BSAC – Bayesian Strategy Network Based Soft Actor Critic in Deep Reinforcement Learning." submitted 2022.

Chapter 2

Background

In this chapter, we will briefly review the background on cooperative multi-agent/robot systems (MAS/MRS), utility theory, game theory, information theory, probabilistic graphical models (PGM), and deep reinforcement learning (DRL).

2.1 Cooperative MAS and MRS

ALLIANCE Parker, 1994a and ACTRESS Asama et al., 1989 are among the earliest heterogeneous multi-agent/robot systems (MAS/MRS) developed by researchers, bringing the benefits from information sharing among agents, data fusion, distribution of tasks, time, and energy consumption Ismail et al., 2018. Therefore, robots of various shapes, sizes, and capabilities such as unmanned aerial vehicles (UAVs), unmanned ground vehicles (UGVs), humanoids, and others have been designed to cooperate with each other and humans to successfully accomplish complex tasks, which will significantly increase the spectrum of automated tasks Rizk et al., 2019.

However, building a robust and intelligent cooperative MAS/MRS is challenging. It involves designing the low-level planning and control framework, the high-level decision-making and learning system, and the efficient decentralized or distributed communication architecture. Moreover, the most reflected and affected key elements and current issues in cooperative MAS/MRS are listed as follow:

2.1.1 Types of agents: homogeneous and heterogeneous

Based on the physical structure and capabilities of the AI agents (robots), we can divide the MAS/MRS into two categories: *homogeneous* and *heterogeneous*.

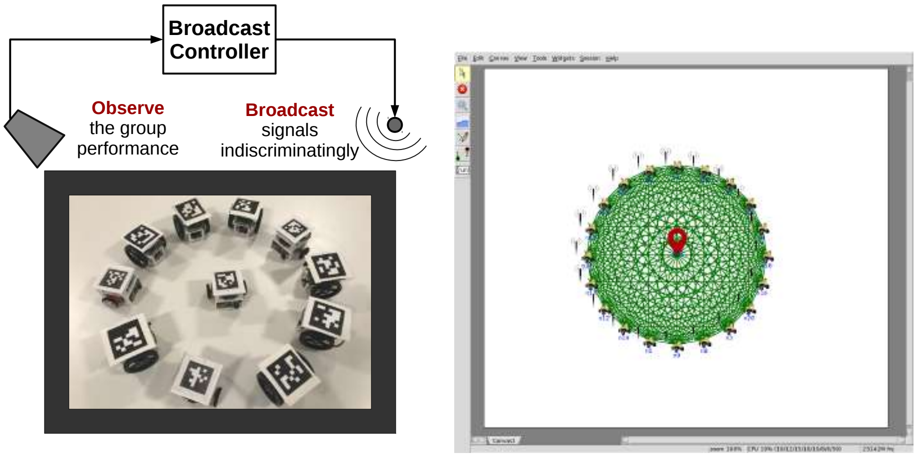


Figure 2.1: Homogeneous Agents Control with Broadcast Ismail et al., 2018 and Simulate with CORE

Homogeneous agents have identical physical structures and capabilities (Figure 2.1 [2]) like *Swarm Robotics* Hamann, 2018. Nevertheless, for heterogeneous agents, their capabilities are not identical, and they are different among robots, where the individual has its specialization or a specific task to fulfill Yan et al., 2013. Also, their physical structures are not identical among group members (Figure 2.2).

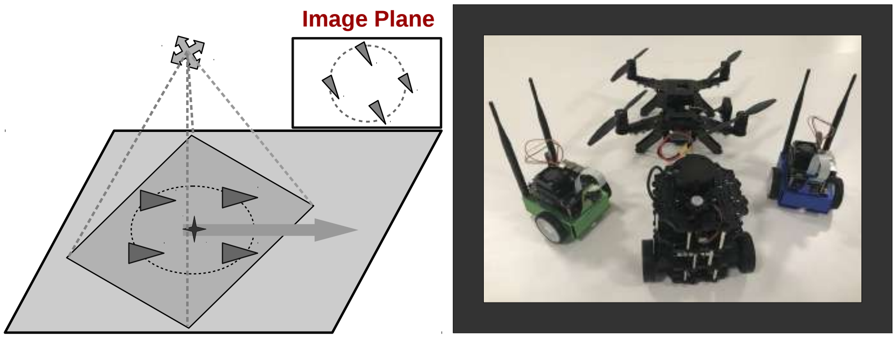


Figure 2.2: The Heterogeneous Team includes a Single UAV (Pelican Quadrotor) Controlling Many UGV Ismail et al., 2018

Although homogeneous agents can perform the task efficiently Sugawara and Sano, 1997 and maximize the task coverage Parker, 1994b, the heterogeneous agents present higher adaptability and more applications J. Li and Li, 2015.

[2] **CORE** Ahrenholz, 2010 (Common Open Research Emulator) is a tool for building virtual networks. As an emulator, CORE builds a representation of a real computer network that runs in real time, as opposed to simulation, where abstract models are used. The live-running emulation can be connected to physical networks and routers. It provides an environment for running real applications and protocols, taking advantage of tools provided by the Linux operating system.

Generally speaking, considering the factors of efficiency, adaptability, robustness, stability, interference, and consensus, the selection of homogeneous or heterogeneous agents depends on the specific research application of the cooperative MAS/MRS.

2.1.2 Control Models: reactive, deliberative, and hybrid

According to the roles and capabilities of each agent in the group and the working style of the entire system, the cooperative MAS/MRS can be classified into three categories as follow:

The *reactive model* can be regarded as decentralized control, which does not rely on high-level communication between group members. In other words, the cooperation between agents depends on direct perception, signal broadcast, or indirect communication via environmental changes. Without the interference of the central controller, each agent can select any task to execute based on its capabilities independently. For example, Glorennec, 1997 coordinated the robots to avoid obstacles and other group members in MRS through fuzzy logic techniques. In terms of local information context, Lopez-Franco et al., 2015 introduces decentralized control to stabilize the nonlinear MAS through neural inverse optimization. Y. Chen and Sun, 2016 introduces a new distributed control law helping the MAS/MRS optimize the achieving consensus to avoid collisions. However, how to balance the efficiency of the centralized systems and the adaptability of the distributed systems is still a challenging problem in MAS/MRS studies.

On the other hand, the *deliberative model* focuses on the high-level communication, rich sensor, and complete environment information representation among agents interactions, also known as the centralized approach Ismail et al., 2018. A global map is usually shared among agents to drive them fulfilling the tasks efficiently in the cooperative MAS/MRS Atınç et al., 2014; Oleiwi et al., 2015.

From the computation complexity perspective, the deliberative control calculates a plan involving the global information; in contrast, the reactive control is more towards computing a local plan through partial observation. The deliberative control shows advantages in computation efficiency and energy effectiveness on a small scale. However, when the number of agents expands, it will lead to various issues, such as adaptability, flexibility, robustness, stability, and scalability. Distributed systems are potential solutions that have been proven to satisfy those requirements in static and dynamic environments Ren and Cao, 2011. Nevertheless, the main issue is that agents can not effectively predict and control group behaviors with limited local information. Furthermore, other issues in distributed systems such as consensus, formation, containment [3], task allocation, optimization, and intelligence can not be ignored.

The *hybrid architecture* integrates reactive and deliberative control, which offers the most widespread solution in controlling MAS/MRS Posadas et al., 2008. Especially in the real world, agents need to adapt to various dynamic and uncertain environments, and hybrid control provides a robust and flexible model to organize agents' collaborative behaviors efficiently. Furthermore, considering the technical and economic factors, we still need to balance them based on different task requirements and scenarios in our system design.

2.1.3 Communications: implicit and explicit

Communication plays a crucial role in the cooperative MAS/MRS. It is a mode of interaction within agents and helps them build various relationships through sharing and exchanging information. In other words, systems can rapidly get consensus and adapt to various scenarios with an efficient communication architecture. Especially in the era of IoT, plenty of information is extracted from the environment through sensor fusion and distributed sensing; how to combine IoT and MAS into a unified framework, termed Internet of robotic things Ray, 2016, to form so-called seamless cooperation between the various "things" becomes a trend. Furthermore, communication affects the perception of each agent and influences the group decision-making, which directly determines the success in the cooperation of the MAS/MRS.

According to the studies in Y. U. Cao et al., 1997, from mobile robots' communication angle, communication structures can be classified into two groups: interaction via sensing – *implicit* and interaction via communications – *explicit*. Implicit communication refers to the local interactions between agents, and agents sense each other through various sensors. Table 2.1 lists the typical studies in MRS with implicit communication.

Table 2.1: Typical Studies in Implicit Communication

Tasks	**Sensors**
avoid other robots, remove obstacles and pass objects Kuniyoshi et al., 1994	CCD cameras
follow the leader avoiding obstacles Al-Jarrah et al., 2015	ultrasound sensors
cooperatively track the target and push the box Pham et al., 2006	side sensors
UAV allocate UGV Rosa et al., 2015	UAV: video camera and IMU; UGV: onboard laser range finder sensor
cooperatively collects the pucks Sugawara and Sano, 1997	photo and IR sensors

[3] **Containment problem** refers to introducing more than one leader among the agents to ensure the groups are not ventured by the hazardous environment. If the agents are faced with this situation, they will move the robots to the safe region spanned by a group of leaders. The agents can either be homogeneous agents that have identical dynamics or heterogeneous agents that have different dynamics Ismail et al., 2018

In contrast, explicit communication usually requires onboard communication modules and involves the network topologies and communication protocol design when directly exchanging information between agents or via broadcasting messages Ismail et al., 2018. Table 2.2 presents an example of explicit communication implemented in MRS.

Table 2.2: Typical Studies in Explicit Communication

Tasks	Communication Styles	Interaction
the control architecture for chain micro robots Brunete et al., 2012	I^2C command exchange protocol	one to many modules for global; agent to agents for local.
cooperatively push paralyzed robots to a specific point Simonin and Grunder, 2009	infrared serial communications interface/transceiver	one to all agents (broadcast)
cooperatively MRS using Hello-Call communication Ichikawa et al., 1993	"hello-call" protocol	agent to agent
swarm robots using wireless networks control robots W. Li and Shen, 2011	Wifi	one to many agents (broadcast)
using local communication grouping Yoshida et al., 1994	information diffusion	agent to groups of agents

2.1.4 Cooperative MAS Decision-Making and Learning

Although cooperative MAS decision-making used to be studied in many separate communities, such as evolutionary computation, complex systems, game theory, graph theory, and control theory (Figure 2.3 [4]) Rizk et al., 2018, these problems are either be episodic or sequential Russell and Norvig, 2002. Agents' actions or behavior are usually generated from a sequence of actions or policies, and the decision-making algorithms are evaluated based on policy optimality, search completeness, time complexity, and space complexity [5].

Considering decision-making in the context of cooperative MAS, the learning process can be centralized or decentralized. Panait and Luke, 2005 divides it into two categories: *team learning* and *concurrent learning*.

In **team learning**, only one learner involves the learning process and represents a set of behaviors for the group of agents, which is a simple approach for cooperative MAS learning since the standard single-agent machine learning techniques can handle it. On the other hand, from the difference of the individual behaviors, we can also classify them as *homogeneous* and *heterogeneous*.

For the *homogeneous* team learning, all agents present identical behaviors without considering their physical divergence, which drastically reduces the search space of the learning process. However, the system hardly achieves high performance due to less specialization among agents. Especially facing vast numbers of agents like the *swarm* [6], the search space is simply too large to implement heterogeneous learning. For example, Haynes et al., 1996; Haynes and

[4] **Note**: Partially observable stochastic games (POSG) Emery-Montemerlo et al., 2005, interactive dynamic IDs (I-DID) Zeng et al., 2007, network of IDs (NID) Gal and Pfeffer, 2008, multiagent IDs (MAID) Koller and Milch, 2003, interactive partially observable MDP (I-POMDP) Gmytrasiewicz and Doshi, 2004, Multiagent POMDP (M-POMDP) Amato, Oliehoek, et al., 2013, multi-agent MARKOV decision process (M-MDP) Shoham and Leyton-Brown, 2008, Time-dependent MDP (TMDP) Boyan and Littman, 2000, decentralized MDP (Dec-MDP) Shoham and Leyton-Brown, 2008, decentralized POMDP (Dec-POMDP) Amato, Chowdhary, et al., 2013, ant colony optimization (ACO) Bonabeau et al., 1999, bee colony optimization (BCO) Teodorovic and Dell'Orco, 2005, particle swarm optimization (PSO) Kennedy and Eberhart, 1995, Pigeon Duan and Qiao, 2014, genetic algorithms (GA) Mitchell, 1998

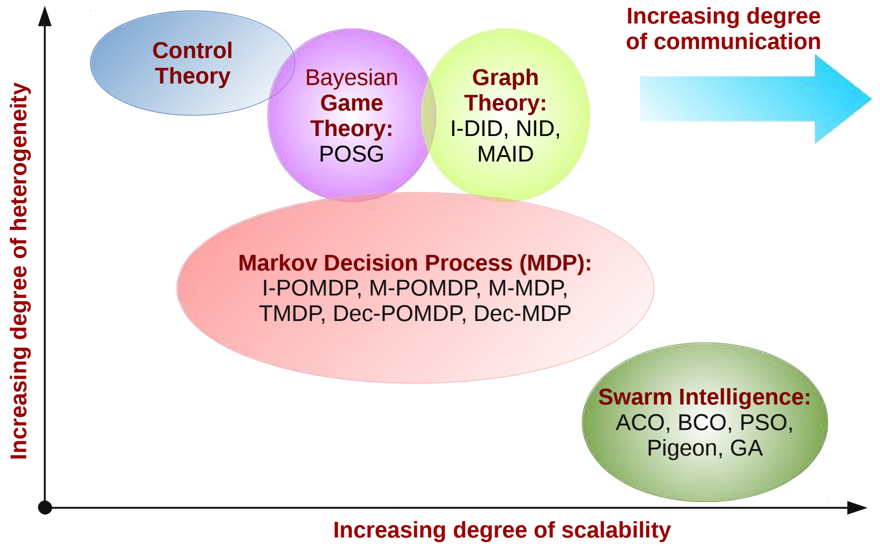

Increasing degree of heterogeneity
Control Theory
Bayesian Game Theory: POSG
Graph Theory: I-DID, NID, MAID
Increasing degree of communication
Markov Decision Process (MDP): I-POMDP, M-POMDP, M-MDP, TMDP, Dec-POMDP, Dec-MDP
Swarm Intelligence: ACO, BCO, PSO, Pigeon, GA
Increasing degree of scalability

Figure 2.3: Cooperative MAS decision-making frameworks comparison Rizk et al., 2018

Sen, 1995 implementing homogeneous team learning evolved behaviors in the predator-prey pursuit domain and Salustowicz et al., 1997; Sałustowicz et al., 1998 compare different machine learning approaches with homogeneous learning. The other typical example is the *cellular automation* (CA) Mitchell et al., 1996, which is an oft-overlooked paradigm for homogeneous team learning. It can interact and communicate with a neighborhood agent and update its own internal state based on the rule applied to all the agents synchronously.

On the other hand, *heterogeneous* team learning considers agents with different behaviors in the group, and learners can develop a unique behavior for each agent. Although it increases the search space, this approach provides more diversity and higher adaptability in the team. There are lots of studies in heterogeneous team learning concerned itself with the requirement for or the emergence of specialists. Luke and Spector, 1996 investigate possible alternatives for evolving teams of agents and Andre and Teller, 1998 implement genetic programming developing the RoboCup simulator for MAS playing soccer. Furthermore, Angeline and Pollack, 1993; Quinn, 2001 use so-called "one-population" coevolution [7] in heterogeneous team learning. Typically, each agent updates the behaviors through its unique learning process.

[5] A policy is optimal if it has the highest utility. A search algorithm is complete if it guarantees to return an optimal policy in finite time, when it exists. Time complexity quantifies the amount of time needed to search for a solution while space complexity quantifies the amount of computational memory needed Rizk et al., 2018

[6] **A swarm** is a large group of identical organisms, natural or artificial, each possessing limited intelligence, that work together to achieve a higher goal beyond their individual capabilities Kennedy, 2006

Considering multiple learning processes improve the team's performance, **concurrent learning** is the most common approach in cooperative MAS. Since concurrent learning projects the large joint team search space onto separate, smaller subset search spaces, Jansen and Wiegand, 2003 argue that it is suitable for the domains that can be decomposed into independent problems and benefit from them. Especially for individual behaviors that are relatively disjoint, it can dramatically reduce search space and computation complexity. Furthermore, breaking the learning process into smaller chunks provides more flexibility for the individual learning process using computational resources.

However, Panait and Luke, 2005 argue that the central challenge for concurrent learning is that each learner is adapting its behaviors in the context of other co-adapting learners over which it has no control, and there are three main thrusts in the area of concurrent learning: *credit assignment*, *the dynamics of learning*, and *teammate modeling*.

The *credit assignment* problem focuses on appropriately apportioning the group rewards to the individual learner. The most typical solution is to separate the team rewards for each learner equally or the reward changing trend of individual learners are the same. The approach divvying up among each learner and receiving rewards through joint actions or strategies is usually termed *global rewards*. Wolpert and Tumer, 2002 argue that global reward does not scale well to increasingly difficult problems because the learners do not have sufficient feedback tailored to their own specific actions.

In contrast, if we do not divide the team rewards equally, evaluating each agent's performance is based on individual behaviors, which means agents do not have the motivation to cooperate and tend to greedy behaviors; these methods are the so-called *local reward*. Through studies different credit assignment policies, Balch et al., 1997; Balch, 1999 argue that local reward leads to faster learning rates, but not necessarily to better results than global reward. Because local rewards increase the homogeneity of the learning group, he suggests that credit assignment selection relies on the desired degree of specialization. Furthermore, Mataric, 1994 argues that individual learning processes can be improved by combining separate local reinforcement with types of social reinforcement.

The dynamics of learning consider the impact of co-adaptation on the learning processes. Assuming agents work in dynamic changing and unstructured environments, they need to constantly track the shifting optimal behavior or strategy adapting to various situations, especially considering the agents may change other group members' learning environments. Evolutionary game theory provides a common framework for cooperative MAS learning. Ficici and Pollack, 2000; Wiegand, 2004 studied the properties of cooperative coevolu-

[7] A single population is evolved using an ordinary evolutionary algorithm, but agents are tested by teaming them up with partners chosen at random from the same population, to form a heterogeneous team Panait and Luke, 2005

tion and Panait et al., 2004 visualize *basins of attraction* [8] to Nash equilibria for cooperative coevolution.

Lots of research in concurrent learning involve game theory to investigate MAS problems Fudenberg et al., 1998; Shoham and Leyton-Brown, 2008. Especially for Nash equilibria, it provides a joint strategy for each group member, which lets individuals not have the motivation to shift their strategy or action in the current situation in terms of a better reward. In the *full-cooperative* [9] scenario with global reward, the reward affects the benefits of each agent and only needs to consider the globally optimal Nash equilibrium. However, the relationships among agents are not clear for more general situations, which combine the cooperative and non-cooperative scenarios. In other words, individual rewards do not directly relate to the team reward, so the *general-sum game* Shoham and Leyton-Brown, 2008 is the cooperative learning paradigm.

Furthermore, to get more accurate information from other group members and cooperate with them efficiently, individuals need to estimate other agents' current status, and *teammate modeling* plays a crucial role. Boutilier, 2013; Chalkiadakis and Boutilier, 2003 implement a Bayesian learning method to update other agents' states, and Suryadi and Gmytrasiewicz, 1999 proposes the same model through learning the beliefs, capabilities, and agents' preferences. Nowak and Sigmund, 1998 introduces the concept of *reciprocity* [10] to analyze an agent's reputation, which is a necessary condition for cooperation with agents.

Generally speaking, cooperative MAS learning involves lots of research areas such as reinforcement learning and control theory, evolutionary computation and stochastic optimization, cellular automata and swarm intelligence models, robotics, and game theory, which holds the promise of widespread applicability.

2.1.5 Levels of Automation

Considering that building a cooperative MAS/MRS fulfills the complex tasks, Kiener and Von Stryk, 2010 proposed four main design modules: task composition, coalition formation, task allocation, and task execution (planning and control). Figure 2.4 presents a workflow of their arguments involving a human expert decomposing complex tasks into simple sub-tasks based on robots' capabilities to form coalitions in MAS/MRS.

Furthermore, according to the complexity of the cooperative tasks, Rizk et al., 2019 groups various heterogeneous MAS/MRS as four different levels of automation. The first level is the least automated, which only executes the lowest planning and control. In addition to achieving the first automation, the second automation can also consider one of the functions of task allocation and coalition formation. Except for not task decomposition, the third level

[8] In dynamical systems research, a **basin of attraction** is the set of all the starting points – usually close to one another – that arrive at the same final state as the system evolves through time Santa Fe, 2021

[9] A **fully cooperative** scenario employs a global reward scheme to divvy reinforcement equally among all agents. Panait and Luke, 2005

[10] They provide two types of **reciprocity**: *direct* (agent A helps agent B and expects its future rewards) and *indirect* (agent A helps agent B and expects the future rewards from others).

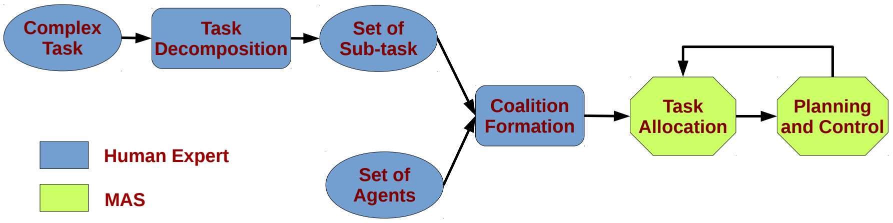


Figure 2.4: Cooperative MAS Workflow involving a human expert Rizk et al., 2019

automation can automate task allocation and coalition formation and automate execution. The fourth level of automation can automate planning and control, task allocation, coalition formation, and task decomposition without human interference. Unfortunately, based on Rizk et al., 2019 studies, no references are currently found to achieve the fourth level of automation.

Table 2.3: Comparison Levels of Automation in Typical MAS/MRS Frameworks

Approach	**Ref.**	**HET**	**COOP**	**COM**	**NEGO**	**DEC**	**Learning**	**DIST**	**Scenario**	**Scale**	**Problems**			
											Patrol	**Coverage**	**Formation**	**Navi.**
ABBA	Jung and Zelinsky, 1999	√	√	√			√	√	dynamic	≤ 10				√
CHARON	Fierro et al., 2002		√					√	static	≤ 10			√	
Token Passing	Farinelli et al., 2006		√	√				√	dynamic	≤ 10				√
Teamcore	Tambe et al., 2000	√	√	√		√		√	dynamic	≤ 10				√
ALLIANCE	Parker, 1998	√	√	√	√			√	dynamic	≤ 20				√
ASyMTRe	Parker and Tang, 2006	√	√	√	√			√	dynamic	≤ 20				√
BITE	Kaminka and Frenkel, 2005	√	√	√		√		√	dynamic	≤ 10				√
DIST Layered	Goldberg et al., 2002		√	√	√			√	dynamic	≤ 10				√
STEAM	Tambe, 1997		√	√		√		√	dynamic	≤ 100				√
Market-Based	Dias and Stentz, 2000		√	√	√				dynamic	≤ 20				√
Hierarchy-Based	Ma et al., 2017		√	√		√			static	≤ 20		√	√	

Table 2.3 [11] presents a comparison among common MAS/MRS frameworks that focus on task allocation and planning at different automation levels.

[11] **Here**, HET: Heterogeneous, COOP: Cooperation, COM: Communication, NEGO: Negotiation, DEC: Decision Making, DIST: Distributed Agents, Navi: Navigation.

2.2 Utility theory

The dominant approach to modeling an agent's interests or needs is *utility theory*. This theoretical approach aims to quantify an agent's degree of preference across a set of available alternatives and understand how these preferences change when an agent faces uncertainty about which alternative he will receive Shoham and Leyton-Brown, 2008.

To describe the interactions between multiple utility-theoretic agents, we use the specific *utility function* to analyze their preference and rational action. The utility function is a mapping from states of the world to real numbers, which are interpreted as measures of an agent's level of happiness (needs) in the given states. If the agent is uncertain about its current state, the utility is defined as the *expected value* of its utility function for the appropriate probability distribution over states Shoham and Leyton-Brown, 2008. From the perspective of the connection between a decision maker and its preference, a decision maker would rather implement a more preferred alternative (act, course of action, strategy) than one that is less preferred Fishburn, 1970.

Furthermore, *expected utility theory* states that the decision maker choose between risky or uncertain prospects by comparing their expected utility values, that is, the weighted sums obtained by adding the utility values of outcomes multiplied by their respective probabilities Mongin, 1998. There are two kinds of theories based on the standard distinction between risk and uncertainty: *Subjective expected utility theory* (SEUT) and *Von Neumann–Morgenstern theory* (VNMT).

2.2.1 Subjective expected utility theory

The theory of subjective expected utility describes the attractiveness of an economic opportunity as perceived by a decision-maker in the presence of risk, which combines two subjective concepts: a personal utility function and a personal probability distribution (usually based on Bayesian probability utility theory) Karni, 2017; Savage, 1972.

In SEUT, if the decision maker adheres to axioms of rationality, believing an uncertain event has possible outcomes $\{x_i\}$ each with utility $u(x_i)$, then the person's choices can be explained as arising from this utility function combined with the subjective belief that there is a probability of each outcome, $P(x_i)$. And the subjective expected utility is Eq. (2.1).

$$\mathbb{E}\left[u(X)\right] = \sum_i u(x_i) \cdot P(x_i) \tag{2.1}$$

The decision relies on the higher subjective expected utility, which means agents may make different decisions because of their various utility functions or beliefs about the probabilities of different outcomes. Moreover, Mongin and d'Aspremont, 1998 further discussed the connection between ethical and utility theory describing the complex social behaviors in human.

2.2.2 Von Neumann–Morgenstern utility theory

In decision theory, the *Von Neumann–Morgenstern utility theory* (VNMT) is also known as the *expected utility hypothesis*, which shows that, under certain axioms [12] of rational behavior, a decision-maker faced with risky (probabilistic) outcomes of different choices will behave as if the agent is maximizing the expected value of some function defined over the potential outcomes at some specified point in the future Von Neumann and Morgenstern, 2007.

2.2.3 Maslow's hierarchy of human needs:

From the psychological perspective, *Maslow's hierarchy of needs* [13] Maslow, 1943 is used to study how humans intrinsically partake in behavioral motivation. Maslow used the terms *physiological*, *safety*, *belonging and love*, *social needs or esteem*, and *self-actualization* to describe the pattern through which human motivations generally move and the conditional dependent relationships between each level's needs. In other words, if the agent's motivation wants to arise at the next stage, the current level's *needs (utilities)* must satisfy a certain amount. Furthermore, for more advanced intelligent agents like human beings [14], many different motivations from various levels of Maslow's needs hierarchy can occur simultaneously. Instead of focusing on a certain need at any time, Maslow, 1981 stated that a specific need might dominate the decision at a certain time, and the likelihood of the dominant need will follow the order presented in the needs hierarchy.

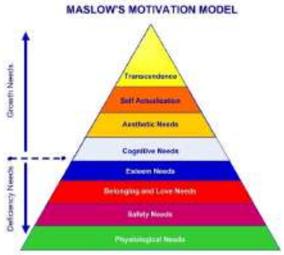
Generally speaking, if we can build a model describing the agent's hierarchy of needs based on the *expected utility theory*, we can develop a more intelligent single agent and MAS representing more complex behaviors to adapt to various dynamic and complex environments.

2.3 Game theory

Game theory can be defined as the study of mathematical models of conflict and cooperation between intelligent rational decision-makers; it provides general mathematical techniques for analyzing situations in which two or more

[12] The four axioms of VNM-rationality are *completeness*, *transitivity*, *continuity*, and *independence*. Von Neumann and Morgenstern, 2007

[13] **Maslow's hierarchy of needs** is often portrayed in the shape of a pyramid, with the largest, most fundamental needs at the bottom, and the need for self-actualization and transcendence at the top. In other words, the idea is that individuals' most basic needs must be met before they become motivated to achieve higher-level needs.



[14] The human brain is a complex system and has parallel processes running at the same time.

individuals make decisions that will influence one another's welfare Myerson, 2013.

2.3.1 Game Types: cooperative and non-cooperative

Game theory has two main branches: *cooperative game theory* (CGT) and *non-cooperative game theory* (NCGT). *Cooperative game theory* focuses on predicting which coalitions will form, the joint actions that groups take, and the resulting collective payoffs Elkind and Rothe, 2016. It is a game between coalitions of players rather than between individuals, and it questions how groups form and how they allocate the payoff among players Von Neumann and Morgenstern, 2007.

Non-cooperative game theory deals with how rational economic agents deal with each other to achieve their own goals and analyzes cases built on the assumption that each player maximizes its own utility function while treating the expected strategic responses of other players as constraints Von Neumann and Morgenstern, 2007. The most common non-cooperative game is the strategic game, in which only the available strategies and the outcomes that result from a combination of choices are listed. A simplistic example of a real-world non-cooperative game is rock-paper-scissors Hayes, 2022.

In a word, the essential between the two branches is that in NCGT, the basic modeling unit is the individual (including its beliefs, preferences, and possible actions), while in CGT, the basic modeling unit is the group Shoham and Leyton-Brown, 2008.

2.3.2 Games in Normal Form

The *normal form*, also known as the strategic form, is the most familiar representation of strategic interactions in game theory. A game written in this way amounts to a representation of every player's utility for every state of the world, in the special case where states of the world depend only on the players' combined actions. Because most other representations of interest can be reduced to it, the normal-form representation is arguably the most fundamental in the game theory Shoham and Leyton-Brown, 2008.

Definition 1 (Normal-form Game)**.** *A (finite, n-person) normal-form game is a tuple (N, A, u), where:*

- *N is a finite set of n players, indexed by i;*
- $A = A_1 \times \cdots \times A_n$*, where* A_i *is a finite set of actions available to player i. Each vector* $a = (a_1, \cdots, a_n) \in A$ *is called an action profile;*

- $u = (u_1, \cdots, u_n)$, *where* u_i*:* $A \mapsto \mathbb{R}$ *is a real-valued utility (or payoff) function for player i.*

There are some typically restricted classes in normal-form games. The *common-payoff games* are so-called *pure coordination games* or *team games*, in which the agents have no conflicting interests and try to coordinate on an action maximizing their global benefits.

Definition 2 (Common-payoff Game)**.** *A common-payoff game is a game in which for all action profiles* $a \in A_1 \times \cdots \times A_n$ *and any pair of agent* i*,* j*, it is the case that* $u_i(a) = u_j(a)$*.*

The other is *constant-sum games*, which are meaningful primarily in the context of two-player games.

Definition 3 (Constant-sum Game)**.** *A two-player normal-form game is constant-sum if there exists a constant* c *such that for each strategy profile* $a \in A_1 \times A_2$ *it is the case that* $u_1(a) + u_2(a) = c$*.*

Particularly, if the constant $c = 0$, it is a *zero-sum* game. Considering *common-payoff games* are pure coordination, *zero-sum games* will be pure competition, and one player's gain is equal to the other player's expense.

2.3.3 Nash Equilibrium

From the standpoint of specific strategies, the *pure strategy* describes a plan determining a player's movement for any situation he might face from his observations. The *mixed strategy* assigns a probability to each *pure strategy*, which randomizes over the set of available actions according to some probability distribution Shoham and Leyton-Brown, 2008.

Definition 4 (Mixed strategy)**.** *Let* (N, A, u) *be a normal-form game, and for any set* X *let* $\Pi(X)$ *be the set of all probability distributions over* X*. Then the set of mixed strategies for player* i *is* $S_i = \Pi(A_i)$*.*

Definition 5 (Mixed-strategy profile)**.** *The set of mixed-strategy profiles is simple the Cartesian product of the individual mixed-strategy sets,* $S_1 \times \cdots \times S_n$*.*

$s_i(a_i)$ presents the probability of an action a_i played under mixed strategy s_i. The subset of actions that are assigned positive probability by the mixed strategy s_i is called the *support* of s_i.

Definition 6 (Support)**.** *The support of a mixed strategy* s_i *for a player* i *is the set of pure strategies* $\{a_i | s_i(a_i) > 0\}$*.* [15]

[15] Note that a **pure strategy** is a special case of a mixed strategy, in which the support is a single action.

According to the *expected utility theory* Eq. (2.1), we can define the *expected utility of a mixed strategy* as Eq. (2.2).

Definition 7 (Expected utility of a mixed strategy). *Given a normal-form game* (N, A, u)*, the expected utility* u_i *for player* i *of the mixed-strategy profile* $s = (s_1, \cdots, s_n)$ *is define as:*

$$\mathbb{E}\left[u_i(s)\right] = \sum_{a \in A} u_i(a) \prod_{j=1}^{n} s_j(a_j) \tag{2.2}$$

Then, we can introduce the most influential concept in game theory, the *Nash equilibrium*. Formally, we define $s_{-i} = (s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_n)$ as a strategy profile s without agent i's strategy and write as $s = (s_i, s_{-i})$. If the agent other than i (whom we denote $-i$) were to commit to play s_{-i}, a utility-maximizing agent i would face the problem of determining its *best response*.

Definition 8 (Best response). *Player* i*'s best response to the strategy profile* s_{-i} *is a mixed strategy* $s_i^* \in S_i$ *such that* $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ *for all strategies* $s \in S_i$.

The best response is not necessarily unique, and except in the extreme case in which a unique best response is a *pure strategy*, the number of best responses is always infinite Shoham and Leyton-Brown, 2008. Next, we can define the most central notion –*Nash equilibrium* in non-cooperative game theory based on the idea of the *best response*.

Definition 9 (Nash equilibrium). *A strategy profile* $s = (s_1, \cdots, s_n)$ *is a Nash equilibrium if, for all agents* i*,* s_i *is a best response to* s_{-i}.

Nash equilibrium describes a stable strategy profile that no agent wants to change if it knows the strategies other agents were following.

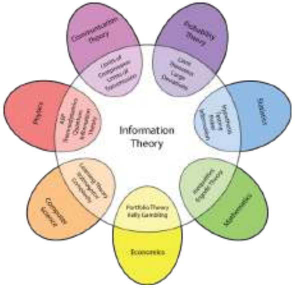
Furthermore, J. Nash, 1951; J. F. Nash et al., 1950 had proved that it has at least one *Nash equilibrium* for every game.

Theorem 1 (The existence of Nash equilibrium). *Every game with a finite number of players and action profiles has at least one Nash equilibrium.*

2.4 Information theory

Information theory intersects physics (statistical mechanics), mathematics (probability theory), electrical engineering (communication theory), and computer science (algorithmic complexity) Thomas and Joy, 2006. The initial questions treated by *information theory* focus on data compression and transmission, but it has fundamental contributions to various research areas [16].

[16] Especially, *information theory* has currently become one of the dry topics that marginally touches Machine learning (ML)Vanam, 2017. And the below picture presents the relationship of information theory to other fields:



2.4.1 Entropy

A key measure in *information theory* is *entropy*, and the concept of entropy in information theory is related to the concept of entropy in statistical mechanics. It is a measure of the uncertainty of a random variable. Formally, let X be a discrete random variable with $\mathcal{X}$ and probability mass function $p(x) = Pr\{X = x\}, x \in \mathcal{X}$. The *entropy* can be defined by Eq. (2.3) Thomas and Joy, 2006.

Definition 10 (Entropy). *The entropy $H(X)$ of a discrete random variable X is defined by*

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \tag{2.3}$$

Note that *entropy* is a functional of the distribution of X, and does not depend on the actual values taken by the random variable X, but only on the probabilities. Furthermore, we introduce the *joint entropy*, *conditional entropy*, and the corresponding *chain rule* as follow:

Definition 11 (Joint entropy). *The joint entropy $H(X,Y)$ of a pair of a discrete random variable (X,Y) with a joint distribution* p(x,y) *is defined as*

$$\begin{aligned} H(X,Y) &= -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log p(x,y) \\ &= -E \log p(X,Y) \end{aligned} \tag{2.4}$$

Definition 12 (Conditional entropy). *If $(X,Y) \sim p(x,y)$, the conditional entropy $H(Y|X)$ is defined as*

$$\begin{aligned} H(Y|X) &= -\sum_{x \in \mathcal{X}} p(x) H(Y|X = x) \\ &= -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log p(y|x) \\ &= -E \log p(Y|X) \end{aligned} \tag{2.5}$$

Theorem 2 (Chain rule of entropy).

$$H(X,Y) = H(X) + H(Y|X) \tag{2.6}$$

Moreover, the *chain rule* is easy to extend to multi-variables as Eq. (2.7).

$$H(X_1, \cdots, X_n) = H(X_1) + H(X_2|X_1) + \cdots + H(X_n|X_1, \cdots, X_{n-1}) \tag{2.7}$$

2.4.2 Relative Entropy

The *relative entropy* is a measure of the distance between two distributions. In statistics, it arises as an expected logarithm of the likelihood ratio. The relative entropy $D(p\|q)$ is a measure of the inefficiency of assuming that the distribution is q when the true distribution is p Thomas and Joy, 2006.

Definition 13 (Relative entropy). *The relative entropy or Kullback-Leibler distance between two probability mass function $p(x)$ and $q(x)$ is defined as*

$$\begin{aligned} D(p\|q) &= \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \\ &= E_p \log \frac{p(X)}{q(X)} \end{aligned} \tag{2.8}$$

To continuous random variables, the relative entropy $D(f\|g)$ [17] between two densities f and g is defined by:

$$D(f\|g) = \int f \log \frac{f}{g} \tag{2.9}$$

2.4.3 Mutual Information

Mutual information is a measure of the amount of information that one random variable contains about another random variable and is the reduction in the uncertainty of one random variable due to the knowledge of the other Thomas and Joy, 2006.

Definition 14 (Relative entropy). *Consider two random variables X and Y with a joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$. The mutual information $I(X; Y)$ is the relative en-*

[17] $D(f\|g)$ is finite only if the support set of f is contained in the support set of g.

tropy between the joint distribution and the product distribution $p(x)p(y)$:

$$
\begin{aligned}
I(X;Y) &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \\
&= D(p(x,y)\|p(x)p(y)) \\
&= E_{p(x,y)}\log\frac{p(X,Y)}{p(X)p(Y)}
\end{aligned}
\tag{2.10}
$$

To continuous random variables, The mutual information $I(X;Y)$ between two random variables with joint density $f(x,y)$ is defined as

$$
I(X;Y) = \int f(x,y)\log\frac{f(x,y)}{f(x)f(y)}dxdy \tag{2.11}
$$

From above definitions, it is clear that:

$$
\begin{aligned}
I(X;Y) &= H(X) - H(X|Y) = H(Y) - H(Y|X) \\
&= H(X) + H(Y) - H(X,Y)
\end{aligned}
\tag{2.12}
$$

and

$$
I(X;Y) = D(f(x,y)\|f(x)f(y)) \tag{2.13}
$$

Note that $D(f\|g)$ and $I(X;Y)$ have the same properties in the discrete and continuous case.

2.4.4 Principle of Maximum Entropy

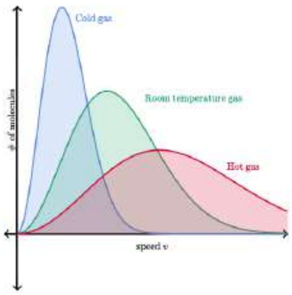
In physics, the *Maxwell–Boltzmann distribution* [18] is the maximum entropy distribution under the temperature constraint. It is implicit that the maximum entropy methods in physics have the asymptotic equipartition property, which means that all microstates are equally probable Thomas and Joy, 2006.

From the information theory perspective, the *principle of maximum entropy* states that the probability distribution which best represents the current state of knowledge about a system is the one with the largest entropy in the context of precisely stated prior data Jaynes, 1957a, 1957b.

Theorem 3 (Maximum entropy distribution). *Let f be a probability density satisfying the constraints*

$$
\int_S f(x)r_i(x) = \alpha_i,\ 1 \le i \le m \tag{2.14}
$$

[18] The **Maxwell-Boltzmann distribution**



gives the distribution of speeds of molecules in thermal equilibrium as given by statistical mechanics KhanAcademy, 2016

Let $f^*(x) = f_\lambda(x) = e^{\lambda_0 + \sum_{i=1}^{m} \lambda_i r_i(x)}$, $x \in S$, and let $\lambda_0, \cdots, \lambda_m$ be chosen so that f^* satisfies Eq. (2.14)*. Then f^* uniquely maximizes $h(f)$ over f satisfying these constraints.* [19]

2.5 Probabilistic graphical models (PGM)

Probabilistic graphical models (PGM) use a graph-based representation as the basis for compactly encoding a complex distribution over a high-dimensional space Koller and Friedman, 2009. The graphical representation encodes probability distributions over complex domains: joint (multivariate) distributions over large numbers of random variables that interact with each other. Specifically, the nodes represent the random variables in the domain, and the edges (links) correspond to probabilistic relationships or interactions among them. Generally, there are two types of PGM: the directed PGM, so-called *Bayesian networks* (BN), and the undirected PGM, also known as *Markov Random Fields* (MRF) [20]. Here, we mainly discuss the Bayesian networks as follow.

2.5.1 Bayesian Networks

Bayesian networks (BN) represent a set of variables and their conditional dependencies via directed acyclic graphs (DAG) $\mathcal{G}$, whose nodes are the random variables in the domain and whose edges correspond, intuitively, to direct influence of one node on another Koller and Friedman, 2009. They are ideal for taking an event that occurred and predicting the likelihood that any one of several possible known causes was the contributing factor. Figure 2.5 provides an simple example of Bayesian network Gonzalez, 2007.

In other words, the graph $\mathcal{G}$ can be viewed in two different ways Koller and Friedman, 2009:

- As a data structure that provides the skeleton for representing a joint distribution compactly in a factorized way;

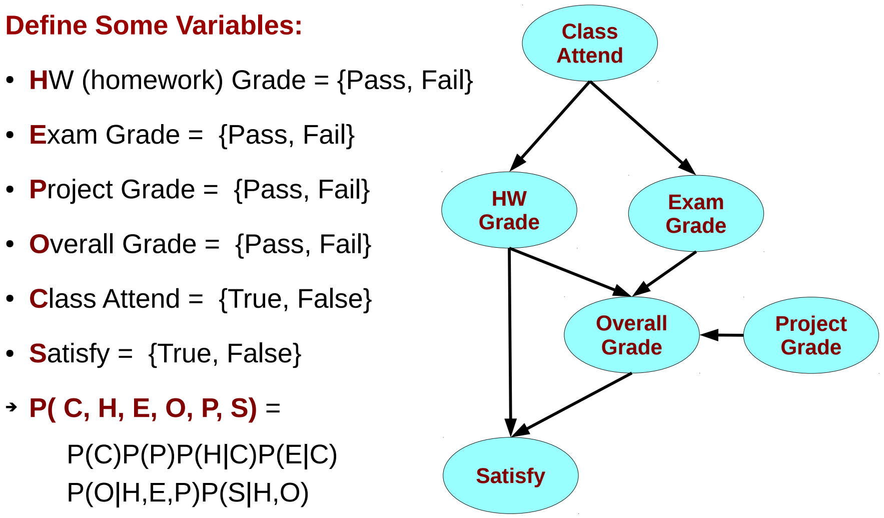
- As a compact representation for a set of conditional independence assumptions about a distribution.

Formally, the *Bayesian network* is defined as following Russell and Norvig, 2002:

Definition 15 (Bayesian Network). *Let $G = (V, E)$ be a directed acyclic graph (DAG) and let $X = (X_v), v \in V$ be a set of random variables indexed by V. X is a Bayesian network with respect to G if its joint probability function can*

[19] It has the same properties in the discrete case.

[20] **Markov Random Fields** or undirected graphical model is a set of random variables having a Markov property described by an undirected graph. In other words, a random field is said to be a Markov random field if it satisfies Markov properties. Sherrington and Kirkpatrick, 1975



Define Some Variables:

- **H**W (homework) Grade = {Pass, Fail}
- **E**xam Grade = {Pass, Fail}
- **P**roject Grade = {Pass, Fail}
- **O**verall Grade = {Pass, Fail}
- **C**lass Attend = {True, False}
- **S**atisfy = {True, False}

➔ **P(C, H, E, O, P, S)** =
P(C)P(P)P(H|C)P(E|C)
P(O|H,E,P)P(S|H,O)

Figure 2.5: An example of a Bayesian network describing how to satisfy a course

be written as a product of the individual density functions, conditional on their parent variables:

$$p(x) = \prod_{v \in V} p(x_v | x_{pa(v)}) \tag{2.15}$$

Where $\mathrm{pa}(v)$ *is the set of parents of* v*. For any set of random variables, the probability of any member of a joint distribution can be calculated from conditional probabilities using the chain rule as follow:*

$$p(X_1 = x_1, \cdots, X_n = x_n) = \prod_{v=1}^{n} P(X_v = x_v | X_{v+1} = x_{v+1}, \cdots, X_n = x_n) \tag{2.16}$$

2.5.2 Inference and Learning

Typically, Bayesian networks perform three main inference and learning tasks: *inferring unobserved variables*, *parameter learning*, and *structure learning*.

For *inferring unobserved variables*, the most common inference methods are listed as follow: *variable elimination* (VE), which eliminates the non-observed non-query variables one by one by distributing the sum over the product; *clique tree propagation*, which caches the computation so that many variables can be queried at one time and new evidence can be propagated quickly; and *recursive conditioning and AND/OR search*, which allow for a space-time tradeoff and match the efficiency of variable elimination when enough space is used. All of these methods have complexity that is exponential in the network's treewidth Koller and Friedman, 2009. Although various learning and inference algorithms are available for each PGM, the most cost-effective one is probably *maximum a posteriori* (MAP) [21], which maximizes the latent variable's posterior probability H. Wang and Yeung, 2020.

Parameter learning is the simplest learning task for Bayesian networks. There are two main approaches to dealing with *parameter-estimation task*: one is *maximum likelihood estimation* (MLH), and the other is *Bayesian prediction*. Furthermore, automatically learning the graph structure of a Bayesian network is a challenging problem within machine learning, which refers to learning the structure of the directed acyclic graph (DAG) from data. There are two major approaches for *structure learning*: score-based and constraint-based Ermon, 2021.

Strictly speaking, for Bayesian networks, the process of finding parameters is learning, and the process of finding latent variables based on given parameters is inference, but when only observed variables are given, inference and learning are often intertwined H. Wang and Yeung, 2020.

2.6 Deep reinforcement learning (DRL)

One of the primary goals of the field of AI is to produce fully autonomous agents that interact with their environments to learn optimal behaviors, improving over time through trial and error Arulkumaran et al., 2017. However, the traditional RL framework was based on experience-driven autonomous learning, which lacked scalability and was inherently limited to fairly low-dimensional problems . On the other hand, the development of deep learning significantly improved the state-of-the-art tasks in machine learning, such as object detection, speech recognition, and language translation LeCun et al., 2015. Currently, deep learning enables reinforcement learning (RL) to tackle many intractable problems with high-dimensional state and action space, such as learning to play video games directly from pixels, and DRL algorithms are also applied to robotics, allowing control policies for robots to be learned directly from camera inputs in the real world Levine et al., 2016; Levine et al., 2018.

[21] In Bayesian statistics, a **maximum a posteriori probability** (MAP) estimate is an estimate of an unknown quantity, that equals the mode of the posterior distribution. The MAP can be used to obtain a point estimate of an unobserved quantity on the basis of empirical data. It is closely related to the method of maximum likelihood (ML) estimation, but employs an augmented optimization objective which incorporates a prior distribution (that quantifies the additional information available through prior knowledge of a related event) over the quantity one wants to estimate. MAP estimation can therefore be seen as a regularization of maximum likelihood estimation. Bassett and Deride, 2019; K. P. Murphy, 2012

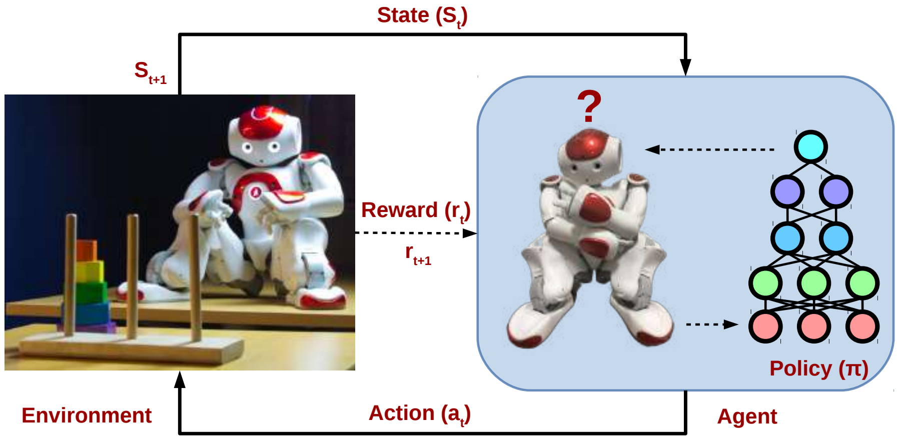


Figure 2.6: The perception-action-learning loop in NAO building AI Wallkötter, 2019

2.6.1 Reward-driven behaviors

Generally speaking, the essence of RL is learning from interaction. When an RL agent interacts with the environments, it can observe the consequence of its actions and learn to change its behaviors based on the corresponding rewards received. Moreover, the theory foundation of RL is the paradigm of trial-and-error learning rooting in *behaviorist psychology* Sutton and Barto, 2018.

In this process, the best action sequences of the agent is determined by the rewards through interaction and the goal of the agent is to learn a policy π maximizing its *expected return* (cumulative rewards with discount). In a word, given a state, a policy returns an action performance and learn an optimal policy from the consequence of actions through trial and error to maximize the *expected return* in the environment. Figure 2.6 illustrates this perception-action-learning loop.

Formally, RL can describe the interaction of an agent with the environment through a Markov decision process (MDP) Puterman, 2014.

- a set of state $\mathcal{S}$ with a distribution of state $p(s)$;
- a set of actions $\mathcal{A}$;
- the probability of transition $\mathcal{T}(s_{t+1}|s_t, a_t)$ mapping a state-action pair from time t to $t+1$ under a distribution of state;

- an instantaneous reward function $\mathcal{R}(s_t, a_t, s_{t+1})$;
- a discount factor $\gamma \in [0, 1]$.

In general, the policy π map states to a probability distribution over actions $\pi : \mathcal{S} \to p(\mathcal{A} = a|\mathcal{S})$, and the sequence of states, actions, and rewards in an episode follow a trajectory of the policy. The accumulated reward in each episode results in return $R = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$. The goal of RL is to find an optimal policy π^* maximizing *expected return* from all states:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[R|\pi\right] \tag{2.17}$$

As the above discussion, the rewards received from the environment play a crucial role in determining the agent's interaction behaviors. In other words, through designing a suitable reward mechanism, we can train an agent presenting the corresponding behaviors to achieve a specific goal. It is much similar to the needs-driven behaviors of humans we discuss in Section 2.2.

Furthermore, the rise of deep learning, which relies on the powerful function approximation and representation learning properties of deep neural networks, has provided a new tool to help RL overcome its limitation such as memory complexity, computational complexity, and sample complexity Arulkumaran et al., 2017. The combination, deep reinforcement learning (DRL), teaches agents to make decisions on high-dimensional state space in an end-to-end framework and dramatically improves the generalization and scalability of traditional RL algorithms Shao et al., 2019.

Generally speaking, RL has two branches: *model-free* and *model-based* [22]. Here, we mainly review various DRL model-free methods, including value-based methods, policy gradient methods, actor-critic methods, especially the soft actor-critic method, and the applications as follow:

2.6.2 Value-based Methods

Deep Q-network (DQN) Mnih et al., 2015 is the breakthrough work in DRL, which combines deep learning and Q-learning. DQN uses a deep neural network for function approximation and maintain an *experience replay* (ER) buffer to store interactions $\langle s, a, r, s' \rangle$ Hernandez-Leal et al., 2019. DQN uses neural network parameters θ stabilizing the learning and other parameters, θ^{-}, keeping an additional copy for the target network. For each episode i, DQN will minimize the mean-squared error (MSE) between the Q-network and the target

[22] The key idea behind **model-based** RL is to learn a transition model that allows for simulation of the environment without interacting with the environment directly Arulkumaran et al., 2017

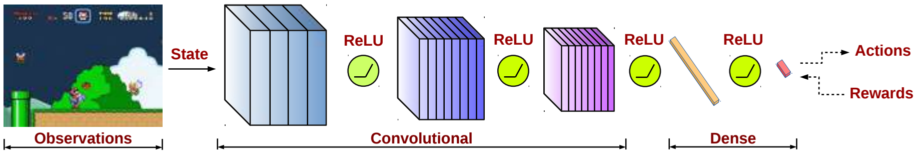


Figure 2.7: Super Mario Deep Q-Network (DQN) training process illustration Hernandez-Leal et al., 2019

network through the Eq. (2.18).

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2\right] \tag{2.18}$$

Where the target network parameters θ^- are set to Q-network parameters θ periodically and the mini-batches of $\langle s, a, r, s' \rangle$ tuples are sampled from the ER buffer Hernandez-Leal et al., 2019. Figure 2.7 illustrated an example of the DQN training process.

There are lots of variety for DQN. For example, Van Hasselt et al., 2016 built Double DQN (DDQN) to reduce the overestimation bias through double estimators Hasselt, 2010 and Z. Wang et al., 2016 proposed a dueling-DQN architecture improving the Double DQN by decomposing the Q-function. To address the limited memory and imperfect information from *partial observability* [23], Deep Recurrent Q-networks (DRQN) Hausknecht and Stone, 2015 replaces the first fully-connected layer with a recurrent neural network, particularly the Long Short-Term Memory (LSTMs) cell Hochreiter and Schmidhuber, 1997, in DQN. Furthermore, Schulman, Chen, et al., 2017 introduced an entropy-regularized version of Q-learning termed Soft DQN, providing better robustness and generalization.

2.6.3 Policy Gradient Methods

Unlike the value-based DRL methods learning implicit policies, the *policy gradient* DRL methods optimize the parameters directly.

Trust region methods: *Trust Region Policy Optimization* (TRPO) Schulman, Levine, et al., 2015 computes an ascent direction optimizing policy gradient with guaranteed monotonic improvement, ensuring a slight change in the policy distribution. The constrained optimization problem of TRPO can be

[23] A Partially Observable Markov Decision Process (POMDP) Åström, 1965; Cassandra, 1998 explicitly models environments where the agent no longer sees the true system state and instead receives an observation (generated from the underlying system state) Hernandez-Leal et al., 2019.

described as Eq. (2.19).

$$\begin{aligned} maximize\, e_\theta \quad & \mathbb{E}_{s\sim\rho_{\theta'},a\sim\pi_{\theta'}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta'}(a|s)}A_{\theta'}(s,a)\right], \\ s.t. \quad & \mathbb{E}_{s\sim\rho_{\theta'}}\left[D_{KL}(\pi_{\theta'}(\cdot|s))\right] \leq \delta_{KL}. \end{aligned} \tag{2.19}$$

Compared to common policy gradient algorithms, *proximal policy optimization* (PPO) Schulman, Wolski, et al., 2017 prevents abrupt changes in policies during training through the loss function. It samples data by interaction with the environments and optimizes the objective function Eq. (2.20) with stochastic gradient ascent Schulman, Levine, et al., 2015. Here, $r_t(\theta)$ denotes the probability ratio.

$$L_i(\theta_i) = \hat{\underset{t}{\mathbb{E}}}\left[\min(r_t(\theta)\cdot\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\cdot\hat{A}_t\right] \tag{2.20}$$

Interestingly, Ilyas et al., 2018 studied PPO and TRPO, arriving at the surprising conclusion that these methods often deviate from what the theoretical framework would predict: gradient estimates are poorly correlated with the true gradient and value networks tend to produce inaccurate predictions for the true value function Hernandez-Leal et al., 2019.

Deterministic policy methods: Apart from stochastic policy, *deep deterministic policy gradient* (DDPG) is a deterministic policy gradient approach, which constructed an exploration policy μ' by adding noise sampled from a noise process $\mathcal{N}$ to the actor policy. Meanwhile, the Q function, as the critic, tells the actor what kind of behavior will obtain more value. The training process uses the loss function of the critic expressed in Eq. (2.21). Through the sample policy gradient, the actor can be updated using Eq. (2.22).

$$\begin{aligned} L &= \mathbb{E}_{s_i,a_i,r_i,s_{i+1}\sim\mathcal{D}}\left[Q(s_i,a_i|\phi) - y\right] \\ y &= r_i + \gamma\left[Q'(s,a|\phi')|_{s=s_{i+1},a=\mu'(s_{i+1})}\right] \end{aligned} \tag{2.21}$$

$$\nabla_\theta J \approx \mathbb{E}_{s_i\sim\mathcal{D}}\left[\nabla_a Q(s,a|\phi)|_{s=s_i,a=\mu(s_i)}\nabla_\theta\mu(s|\theta)_{s=s_i}\right] \tag{2.22}$$

DDPG is an off-policy algorithm and can learn reasonable policies on various tasks. For an advanced version, distributed distributional DDPG (D4PG) Barth-Maron et al., 2018 combines multiple distributed workers writing into the same replay table, providing better performance on continuous control problems.

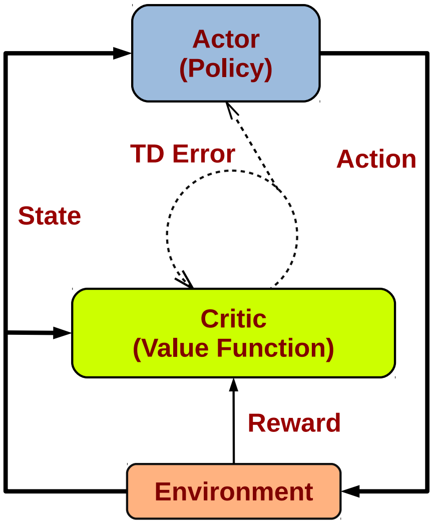


Figure 2.8: The training process of the classical actor-critic model Arulkumaran et al., 2017

2.6.4 Actor-Critic Methods

The common policy-based methods used to have low data efficiency (sample utilization) and large error variance, making the model hard to converge. Konda and Tsitsiklis, 2003 and Schulman, Moritz, et al., 2015 introduced the actor-critic algorithm tradeoff variance reduction of policy gradient with bias overcoming those drawbacks. Generally, the actor-critic methods use the value function as a baseline for policy gradients, such that the only fundamental difference between actor-critic methods and other baseline methods is that actor-critic methods utilize a learned value function Arulkumaran et al., 2017. Figure 2.8 shows the training process of the classical actor-critic model.

Especially for the asynchronous advantage actor-critic (A3C) Mnih et al., 2016, it provides an efficient DRL framework to train several agents in multiple environments using asynchronous gradient descent to optimize the policy. Eq. 2.23 shows the corresponding objective function of the actor. Where $H_\theta(\pi(s_t)$ is the entropy used to encourage exploration.

$$J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} A_{\theta,\theta_v} \log \pi_\theta(a_t|s_t) + \beta H_\theta(\pi(s_t)) \right] \tag{2.23}$$

2.6.5 Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) Haarnoja et al., 2018 is based on the *Principle of Maximum Entropy* discussed in Section 2.4.4 and *soft Q-learning* Haarnoja et al., 2017. Instead of the conventional RL approaches specifying a unimodal policy distribution Figure 2.9 (a) representing the maximal Q-value, the *soft Q-learning* directly builds the policy in terms of exponential Q-values Eq. (2.24) Figure 2.9 (b) and defines the policy through the energy form as an optimal solution for the maximum-entropy RL objective Eq. (2.25).

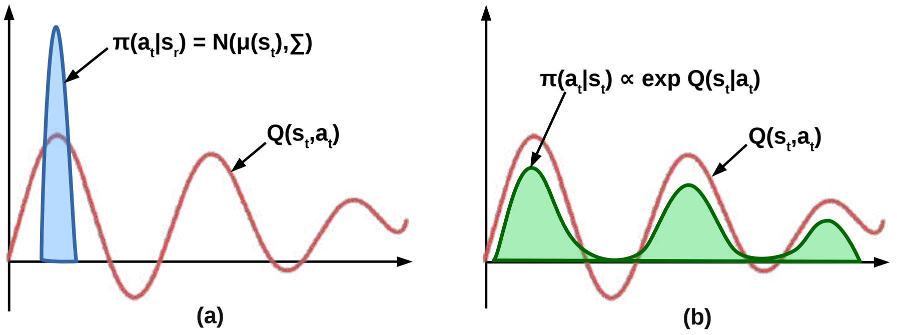


Figure 2.9: An example of multi-model Q-function Haoran Tang, 2017

$$\pi(a|s) \propto \exp Q(s|a) \tag{2.24}$$

$$\pi^*_{MaxEnt} = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} r_t + \mathcal{H}(\pi(\cdot|s_t))\right] \tag{2.25}$$

As a consequence, the agent will become aware of all behaviours that lead to solving the task, which can help the agent adapt to changing situations in which some of the solutions might have become infeasible Haarnoja, 2018; Haoran Tang, 2017. Figure 2.9 shows an example of a multi-model Q-function.

Then, we can obtain the optimal solution of the maximum entropy objective by employing the *soft Bellman equation* Ziebart, 2010 Eq. 2.26.

$$\begin{aligned} &Q(s_t, a_t) = \mathbb{E}\left[r_t + \gamma \operatorname*{softmax}_{a} Q(s_{t+1}, a)\right] \\ &where \quad \operatorname*{softmax}_{a} f(a) := \log \int \exp f(a) da \end{aligned} \tag{2.26}$$

Furthermore, a more general maximum entropy objective Eq. 2.27 which favors stochastic polices by augmenting the objective with expected entropy of the policy over $\rho_\pi(s_t)$. Here, the temperature parameter α determines the relative importance of the entropy term against the reward and controls the stochasticity of the optimal policy Haarnoja et al., 2018.

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_p i} \left[r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))\right] \tag{2.27}$$

The architecture consider a parameterized state value function $V_\psi(s_t)$, soft Q-function $Q_\theta(s_t, a_t)$, and a tractable policy $\pi_\phi(a_t|s_t)$. It updates the policy parameters by minimizing the Kullback-Leibler divergence between the policy π' and the Boltzmann policy in Eq. (2.28).

$$\pi_{new} = \arg\min_{\pi'} D_{KL}\left(\pi'(\cdot|s_t) \middle\| \frac{exp(Q_\theta(s_t, \cdot)}{Z_\theta(s_t)}\right) \tag{2.28}$$

Generally, SAC concurrently learns a stochastic policy, two Q-functions, and a value function, alternating between collecting experience with the current policy and updating from batches sampled from the *experience replay* buffer. As an off-policy policy gradient method, SAC establishes a bridge between DDPG and stochastic policy optimization and incorporates the clipped double-Q trick Shao et al., 2019.

Chapter 3

Self-Adaptive Swarm System (SASS)

With the background discussed in Chapter 2, now we are ready to introduce the new distributed MAS/MRS cooperative architecture – *self-adaptive swarm system* (SASS) and its corresponding modules. Specifically, this chapter will present our proposed SASS framework as a principled and probabilistic way of integrating low-level planning and control and high-level decision-making and learning.

3.1 From agent needs to motivation

In nature, the interaction between and within various elements of a system are complex Chan, 2001. Simple swarm [24] principles acting at the individual agent level can result in complex behavior at the global level. Many natural systems (e.g., brains, immune system, ecology, societies) and artificial systems (parallel and distributed computing systems, artificial intelligence, evolutionary programs) are characterized by apparently complex behaviors that emerge as a result of often nonlinear spatiotemporal interactions among a large number of component systems at different levels of organization Levin, 1998.

More specifically, the **needs** describe the necessities for a *self-organizing system* [25] to survive and evolve, which arouses an agent to action toward a goal, giving purpose and direction to behavior. Based on *Maslow's hierarchy of needs* discussed in Chapter 2.2, an agent needs to satisfy a certain amount of needs in the current level as a condition to arise at the next stage – *upgrade* and *evolution*. Furthermore, according to the *expected utility hypothesis* Von Neumann and Morgenstern, 2007, considering the probabilities (risks) of the agent needs in the specific situation, the agent's decisions always want to maximize expected

[24] We use the term **swarm** to denote the multi-agent context of the proposed multi-agent/robot cooperative behaviors.

[25] **Self-organizing systems** are structures that process where some form of overall order or coordination arises out of the local interactions between smaller component parts of an initially disordered system. The process of self-organization can be spontaneous, and it is not necessarily controlled by any auxiliary agent outside of the system. Yates, 2012

utility (needs) also maximize the probability of the decision's consequences being preferable to some uncertain threshold.

For example, to maximize the chance of detecting predators, forage, and save more energy while migrating to different locations, animals usually split into different swarms to minimize their aggregated threat and maximize benefits according to different scenario forming complex adaptive systems. These behaviors and patterns are also regarded as *swarm intelligence*, which is the collective behavior of distributed and self-organized systems Beni and Wang, 1993. Therefore, the **intelligence** exhibited here is the global adaptive behaviors caused by individual agents' interaction and cooperation, building complex and organized relationships based on their current needs or the entire system utilities in specific scenarios.

MAS/MRS Parker, 2008 potentially share the properties of *swarm intelligence* in practical applications such as search, rescue, mining, map construction, exploration, etc. MAS/MRS that allow task-dependent dynamic reconfiguration into a team are among the grand challenges in Robotics G.-Z. Yang et al., 2018 necessitating the research at the intersection of communication, control, and perception. Currently, planning-based approaches combined with star-shaped communication models can not generally scale or handle a large number of agents in a distributed manner Desaraju and How, 2012; Luo et al., 2016.

Especially, *swarm robotics* and *swarm intelligence* have been well studied in the literature Hamann and Wörn, 2008. Multi-robot modeling and planning algorithms are among those well-studied topics yet require task-specific or scenario-specific application limitations. Martinoli, 1999 presents the modeling technique based on rate equations, a promising method using temporal logic to specify and possibly prove emergent swarm behavior by Winfield et al., 2005. Soysal and Şahin, 2006 apply combinatorics, and linear algebra is deriving a model for an aggregation behavior of swarms. Some studies also applied control theory to model and analyzed multi-robot and swarm systems Feddema et al., 2002; Gazi and Passino, 2003. Recently, Otte et al., 2019 discussed various auction methods for multi-robot task allocation problem in communication-limited scenarios where the rate of message loss between the auctioneer and the bidders are uncertain.

From the MAS perspective, one of the earliest pioneering works, especially in the distributed artificial intelligence, includes Smith and Davis, 1981, where the authors defined the Contract Net Protocol (CNP) for decentralized task allocation. Aknine et al., 2004 extended this idea to m manager agents and n contractor agents negotiation. A protocol for dynamic task assignment (DynC-

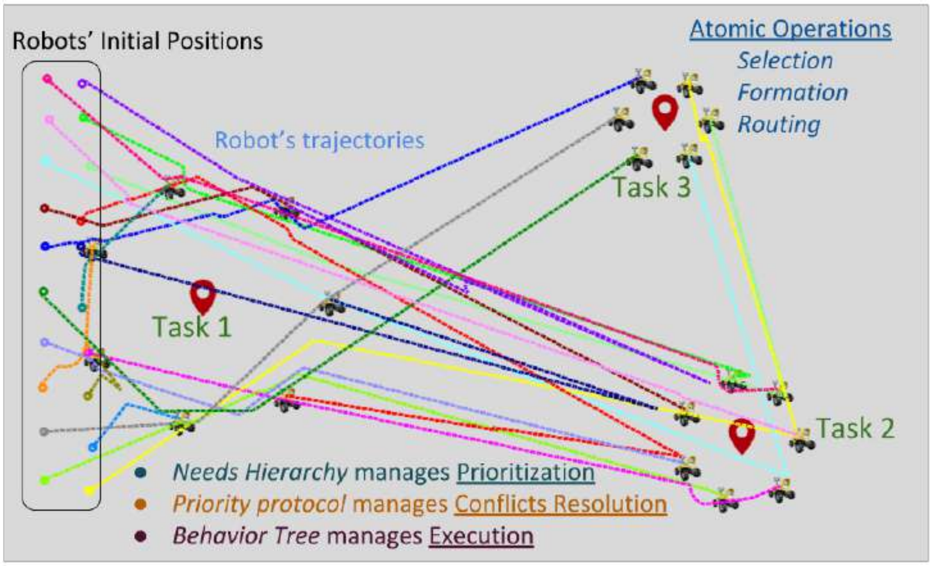

Robots' Initial Positions
Atomic Operations
Selection
Formation
Routing
Robot's trajectories
Task 3
Task 1
Task 2
● Needs Hierarchy manages Prioritization
● Priority protocol manages Conflicts Resolution
● Behavior Tree manages Execution

Figure 3.1: Illustration of reactive multi-robot planning

NET) has been developed by Weyns et al., 2007. However, these methods rely on a central agent (such as an auctioneer or a contractor) to design the negotiation protocol and a supportive framework. They do not generally consider the changes in the agents' status, which restricts the direct applicability in real-world scenarios but only with adaptation.

Importantly, unlike distributed robotic systems, MAS/MRS emphasizes a large number of agents and promotes scalability, for instance, by using local communication Hamann, 2018, which plays a vital role in the whole system building various relationships between each robot to adapt to different environments and situations.

In multi-robot planning and control, several studies focus on navigating a robot from an initial state to a goal state Ayanian and Kumar, 2010; Wagner and Choset, 2015; Wu et al., 2019. So the individual robot's entire route planning is usually computed in high-dimensional joint configuration space. Since formations require robots to maintain stricter relative positions as they move through the environment, the flocking problem could be viewed as a sub-case of the formation control problem, requiring robots to move only minimal requirements for paths taken by specific robots Olfati-Saber, 2006. The research question here is how to design suitable local control laws for each robot to complete the globally assigned tasks efficiently and cooperatively and how paths can be planned for permutation-invariant multi-robot formations Kloder and

Hutchinson, 2006. Earlier, the solutions for such problems in flocking and formations are based on local interaction rules Reynolds, 1987 or behavior-based approaches Balch and Arkin, 1998; Mataric, 1993. More recent approaches focus on proving stability and convergence properties in multi-robot behaviors basing on control-theoretic principles Y. Cao et al., 2012; Murray, 2007; Parasuraman et al., 2018.

To summarize, we aim to develop more advanced MAS/MRS by seeking the fourth level of the autonomous system, which combines task decomposition, group formation, planning, and control Rizk et al., 2019. Then, we propose a MAS/MRS cooperation concept, Self-Adaptive Swarm System (SASS), that combines the parts of the perception, communication, planning, and execution to address this gap. See Figure 5.1 for an illustration of the concept [26].

3.2 Agent needs hierarchy

3.2.1 Agent (Robot) needs hierarchy

To model an artificial intelligent agent's (like robots) motivation and needs in the interaction, we define the *Agent Needs Hierarchy* inspired by Maslow's hierarchy of human psychological needs Maslow, 1943. The *Agent Needs Hierarchy* has five different levels (see Figure 3.2): safety needs; basic needs (energy, time constraints, etc.); capability (heterogeneity, hardware differences, communication, etc.); team cooperation (global utility, team performance, cooperation, and global behaviors); and self-upgrade (learning). Particularly, considering the interaction and cooperation between artificial intelligent agents and humans, we put the safety needs at the lowest level to guarantee their cooperative partners' safety.

In *Agent Needs Hierarchy*, the abstract needs of an agent for a given task are prioritized and distributed into multiple levels, each of them preconditioned on their lower levels. At each level, we express the needs as an expectation over its distribution of the factors/features corresponding to that level.

More specifically, the lowest (first) level is the safety features of the agent (e.g., features such as collision detection, fault detection, etc., that assure safety to the agent, human, and other friendly agents in the environment). The safety needs (Eq. (3.1)) can be calculated through its safety feature's value and corresponding safety feature's probability based on the current state of the agent. After satisfying safety needs, the agent considers its basic needs (Eq. (3.2)), which includes features such as energy levels, data communication levels that help maintain the basic operations of that agent. Only after fitting the safety and

[26] Where robot agents move to Task 1 and Task 2 from their initial positions. During this execution, Task 3 is assigned and the robots react to this new task requirements

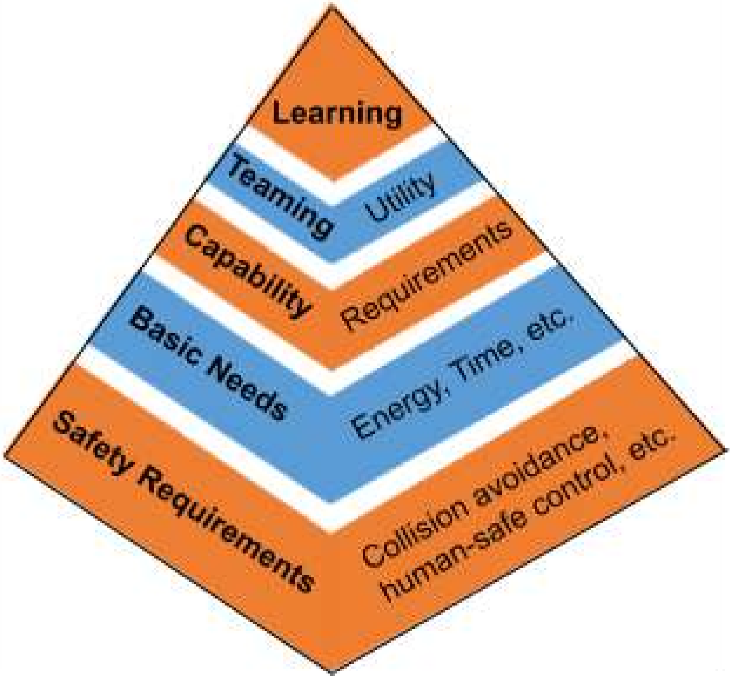


Figure 3.2: Hierarchy of Agent (Robot) Needs

basic needs, an agent can consider its capability needs (Eq. (3.3)), which are composed of features such as its health level, computing (e.g., storage, performance), physical functionalities (e.g., resources, manipulation), etc.

At the next higher level, the agent can identify its teaming needs (Eq. (3.4)) that accounts the contributions of this agent to its team through several factors (e.g., heterogeneity, trust Q. Yang and Parasuraman, 2021, actions) that the team needs so that they can form a reliable and robust team to successfully perform a given mission.

Ultimately, at the highest level, the agent learns some skills/features to improve its capabilities and performance in achieving a given task. For instance, an agent may use RL to learn its policy features (e.g., Q table or reward function) using which it can execute appropriate actions based on the current state. Such learning features are accounted into its learning needs expectation (Eq. (3.5)). The expectation of agent needs at each level are given below:

$$Safety\ Needs : \mathbb{E}\left[N_{s_j}\right] = \sum_{i=1}^{s_j} S_i \cdot \mathbb{P}(S_i|X_j, T); \tag{3.1}$$

$$Basic\ Needs : \mathbb{E}\left[N_{b_j}\right] = \sum_{i=1}^{b_j} B_i \cdot \mathbb{P}(B_i|X_j, T, N_{s_j}); \tag{3.2}$$

$$Capability\ Needs : \mathbb{E}\left[N_{c_j}\right] = \sum_{i=1}^{c_j} C_i \cdot \mathbb{P}(C_i|X_j, T, N_{b_j}); \tag{3.3}$$

$$Teaming\ Needs : \mathbb{E}\left[N_{t_j}\right] = \sum_{i=1}^{t_j} T_i \cdot \mathbb{P}(T_i|X_j, T, N_{c_j}); \tag{3.4}$$

$$Learning\ Needs : \mathbb{E}\left[N_{l_j}\right] = \sum_{i=1}^{l_j} L_i \cdot \mathbb{P}(L_i|X_j, T, N_{t_j}); \tag{3.5}$$

Here, $X_j = \{P_j, C_j\} \in \Psi$ is the combined state of the agent j with P_j being the perceived information by that agent and C_j representing the communicated data from other agents. T is the assigned task (goal or objective). S_i, B_i, C_i, T_i, and L_i are the utility values of corresponding feature/factor i in the corresponding levels - Safety, Basic, Capability, Teaming, and Learning, respectively. s_j, b_j, c_j, t_j, and l_j are the sizes of agent j's feature space on the respective levels of needs.

The collective need of an agent j is expressed as the union of needs at all the levels in the needs hierarchy as in Eq. (3.6).

$$N_j = N_{s_j} \cup N_{b_j} \cup N_{c_j} \cup N_{t_j} \cup N_{l_j} \tag{3.6}$$

The set of agent needs in a multiagent system can be regarded as a kind of motivation or requirements for cooperation between agents to achieve a specific group-level task.

Furthermore, in more complex scenarios, many different motivations from various levels of the agent needs hierarchy can occur simultaneously. Instead of focusing on a certain need at any time, a specific need might dominate the decision at a certain time, and the likelihood of the *dominant need* or *needs' distribution* will follow the order presented in the needs hierarchy.

3.2.2 Conflict-solving mechanism

In order to solve the conflicts in MAS/MRS cooperation and get *consensus*, we provide the distributed prioritization technique –*negotiation-agreement mechanism* – based on the *agent (robot) needs hierarchy*. The *negotiation-agreement mechanism* solves the conflicts associated with the sub-tasks/elements in task planning. Particularly, in our scenario, it executes the *atomic operations* – the *selection* (task assignment), *formation* (shape control), and *routing* (path planning) – in MAS/MRS through automated planning of state-action sequences. The more details see in Section 3.3 and 3.4.

3.2.3 Task-orient atomic operation

SASS decomposes the complex tasks into a series of simple sub-tasks through which agents can recursively achieve those sub-tasks until we complete the high-level task. We provide several *atomic operations* for the swarm behavior: *selection*, *formation*, and *routing*, which allows us to decompose a particular robot's action plans as flocking, pattern formation, and route planning under the same framework. Specifically, according to different tasks' features and requirements, the MAS/MRS can form various shapes or patterns such as *morphogenesis* [27] to adapt to corresponding scenarios.

3.3 Integrated architecture

In our framework, we decompose the complex tasks into a series of sub-tasks and recursively achieve those sub-tasks until the entire task is completed. Accordingly, we can divide the task allocation and execution into three steps: selection, formation, and routing. This process can be illustrated as a Behavior Tree Colledanchise and Ögren, 2018 that integrates the sense-think-act cycle Q. Yang et al., 2019. The robots are assumed to have low-level motion control and sensor-based perception system for sensing and navigation.

First, the robots are partitioned into one or more groups to perform multiple tasks such as surveillance and patrolling. Then, they will compute the placement within a formation shape at each task (for simplicity, we assume a circular shape to circle the area of the task location, but other formation shapes can also be considered). Finally, the robots choose a suitable path to get to the goal point in that formation. When the new tasks are assigned, these robots need to split up to form new groups or merge into existing groups.

Each robot will first verify that there is a task assigned. If assigned, it will compute an appropriate plan according to its current state and needs. It will then communicate with other robots and perform the negotiation and agreement process until there are no conflicts. Finally, the robots will execute their plans. This process is continuously repeated as a loop in a behavior tree, which processes the flow from left to right.

The proposed framework also can be formalized using the tuple M=(S, A, δ, F). Here, S=(Pe, $Pl_{1}\dots{}_{n}$, $Ne_{1}\dots{}_{n}$, $A\&E_{1}\dots{}_{n}$), where Pe represents perception, Pl represents plan, Ne represents negotiation, and $A\&E$ repre-

[27] **Morphogenesis** is the biological process that causes a cell, tissue or organism to develop its shape. D. W. Thompson and Thompson, 1942; Turing, 1990

Table 3.1: SRSS States Transition Table

State	Event	Behavior	Next State
Pe(Perception)	Broadcasting Current State(BCS)	when have tasks or do not finish them, if do not get all the feedback from other robots, do not change current state	Pe
	Current State Feedback(CSF)	when have tasks or do not finish them, if get all the other robots' feedback, change to the Pl state	Pl
	Finishing Tasks(FT) or No Tasks(NT)	if sense in the Initial State(IS), change the state to Ac	Ac(Accept)
		if sense not in the Initial State(IS), broadcast its state and change the state to Pe	Pe
Pl(Planning)	Broadcasting Plans(BP)	compute the plans and broadcast, if do not get all the plans' feedback from other robots, do not change current state	Pl
	Plans Feedback(PF)	get all the other robots plans' feedback, change to the Ne state	Ne
Ne(Negotiation)	Broadcasting Conflicting Plans(BCP)	get the conflict result and broadcast it, if do not get all the conflicting plans' feedback from other robots, do not change current state	Ne
	Conflicting Plans Feedback(CPF)	get all the other robots conflicting plans' feedback, change to the Pl state	Pl
	Broadcasting Non-Conflicting Plans(BNCF)	get the non-conflict result and broadcast it, if do not get all the non-conflicting plans' feedback from other robots, do not change current state	Ne
	Non-Conflicting Plans Feedback(NCPF)	get all the other robots non-conflicting plans' feedback, change to the A&E state	A&E
A&E(Agreement & Execution)	Finishing Planning(FP)	perception the current environment and set the state to Pe	Pe
	Not Finishing Planning(NFP)	back to the planning state and change the state to Pl	Pl

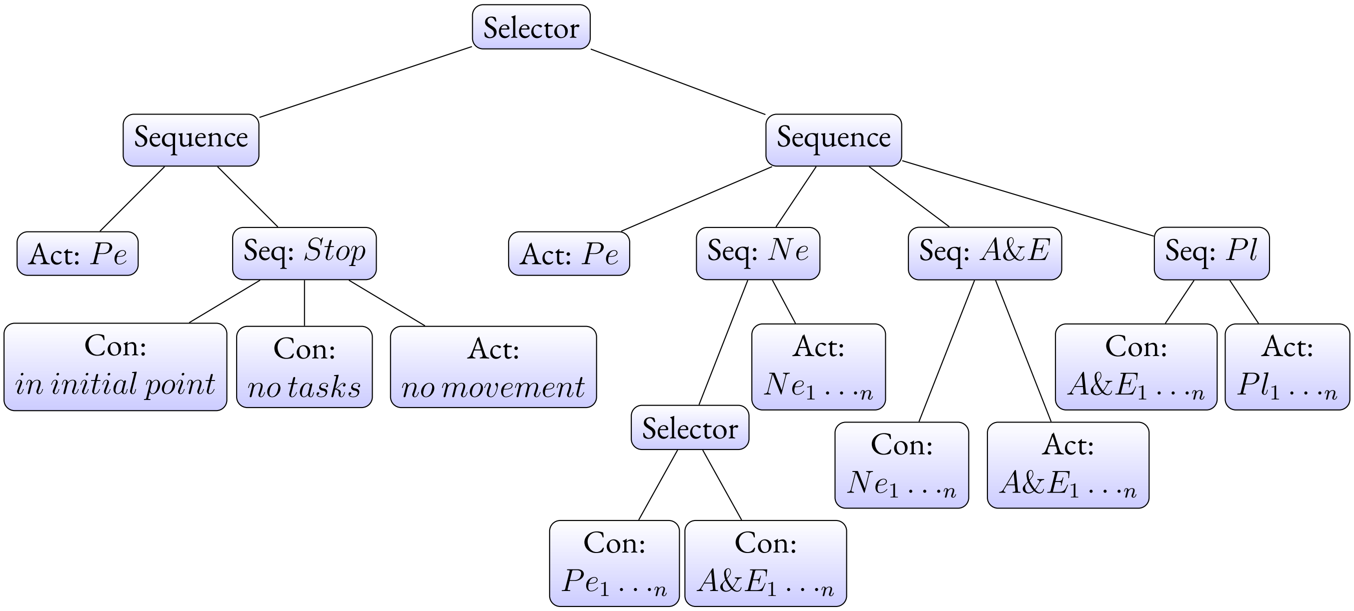


Figure 3.3: Behavior Tree acting at every agent in SASS

sents agreement. The subscript represent the number of iterations for each process, which is finite. $A = (A_1, A_2, \ldots, A_n)$ is a set of individual robot's actions (behaviors). The transition function that maps states (conditions) to actions (behaviors) is δ, defined as $S * A \to S$. F represent the accept state.

After the perception phase, a robot could have n plans $Pl = \{Pl_1, Pl_2, \ldots, Pl_n\}$, where each plan depend on each other requiring sequential execution. Therefore, each plan has negotiation $Ne = \{Ne_1, Ne_2, \ldots, Ne_n\}$ and agreement phases $A\&E = \{A\&E_1, A\&E_2, \ldots, A\&E_n\}$ separately. Furthermore, we can represent the entire process as the *state transition table* Table 3.1. According to our formal definition and the description of SRSS state transition table, we can draw the State transition diagram as Fig 3.4.

To model an individual robot's motivation and needs in the negotiation process, we introduce the priority queue technique inspired by Maslow's hierarchy of human psychological needs Maslow, 1943. We define a robot's need hierarchy at several levels, as shown in Fig 3.2. The lowest level represents the robot's safety needs. In all scenarios, a robot should first consider the safety issues (including human-safe operation) like avoiding conflicts or adversarial attacks. When the situation satisfies the robot's safety needs, the robot will consider its basic but vital needs, such as energy and time availability. Then, it will review its capabilities against the task requirements are subsuming the

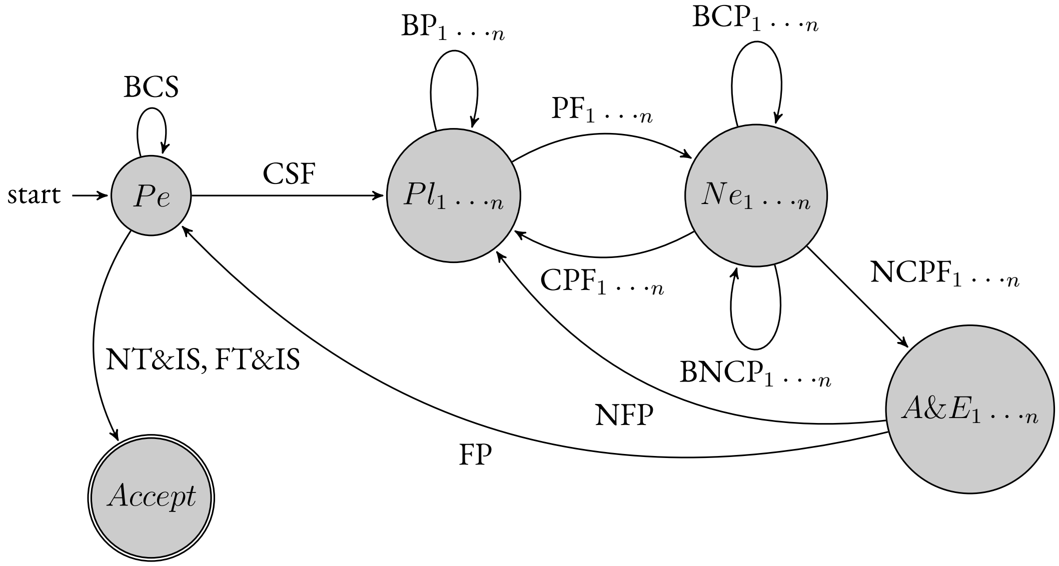


Figure 3.4: SRSS States Transition Diagram

task priority considerations. In the fourth level, the robot finds rewards and utility costs (such as distance and communication) for cooperation within a group. The fifth level is reserved for self-upgrade that could potentially allow multi-robot learning strategies. For instance, after finishing the tasks, robots can upgrade their capabilities based on their experiences through the learning and evolution process. It is worth noting that robots' basic and safety needs are flipped compared to the human needs in Maslow's hierarchy.

The needs at the low level are the precondition of entering higher levels. If a robot cannot satisfy its low-level needs, it will change its behaviors accordingly. We design the priority queue to abstract this model and implement it in the negotiation process of our multi-robots planning framework. Also, the individual robot's current needs can dynamically change according to different scenarios [28]. For example, in the normal state, the robot's behaviors and plans could maximize the task's requirement and minimize its energy. However, in cases of conflicting plans with other robots (e.g., potential collision with a neighboring robot), it will ensure that the safety needs are guaranteed. Similarly, if the robot runs out of battery, it also needs to find its basic needs first (e.g., recharging battery) before completing it.

[28] Currently, in our scenarios, we do not consider the fifth level adding learning methods and complex decision-making approaches into the individual agent and only design a simpler environment to simulate the process.

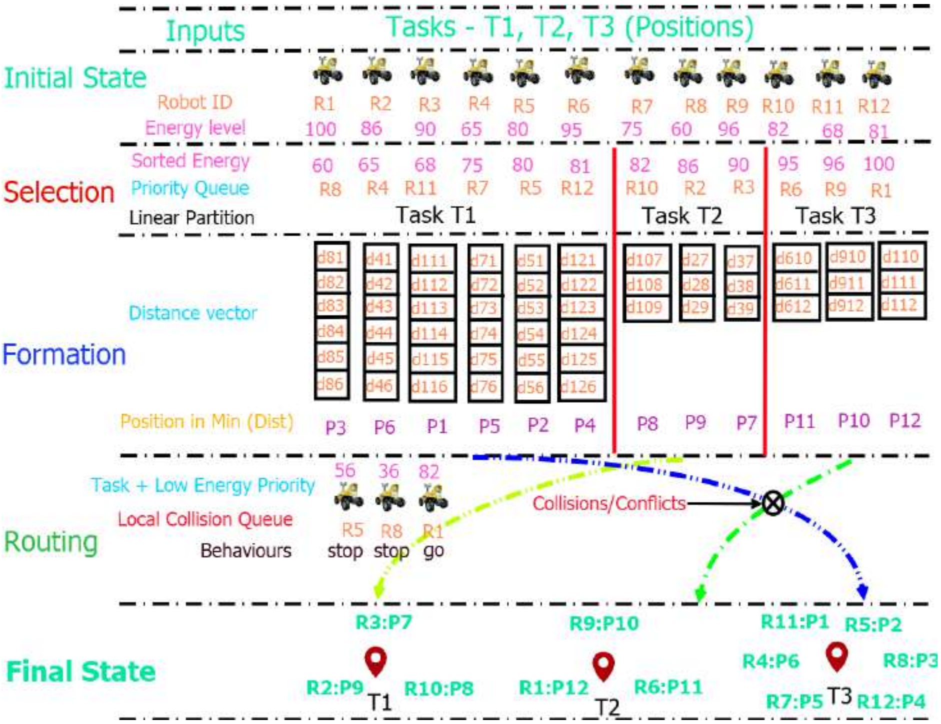


Figure 3.5: Illustration of Task Decomposition through planning at Selection, Formation, and Routing phases with 12 robots and 3 tasks

3.4 Modules introduction

SASS considers the following modules: Perception; Communication; Planning; Negotiation; Agreement, and Execution. We will discuss them separately below.

1) Perception: Each robot uses various on-board (local) sensors for localization, mapping, and recognizing objects/obstacles in the environment.

2) Communication: The process of communication between robots includes broadcasting and reception of the robot's messages (state) to/from other robots Tardioli et al., 2019. The distributed communication in MRS can be regarded as a connected graph. Each robot keeps on communicating with its adjacent/neighbor's robots and exchanging data until they reach *Information Equilibrium*, which means that every group member has the same information for the entire group (see Alg. 1).

3) Planning: In the planning stage, we divide this process into three steps Selection, Formation, and Routing. These *atomic operations* are illustrated in Fig. 3.5 with an example planning problem. In each step, we also introduce a

Algorithm 1: Distributed Communication Mechanism (DCM)

```
Input  : Robot $i$ data $d_i$, The number of robots in group $n$ and
         Adjacent Robots Set $r_a$
Output : The Information Equilibrium data set $D_i$
initialization;
memory data set $D_i = \{\varnothing\}$
    $D_i$.add($d_i$)
    while $length(D_i) \neq n$ do
        for $each\, j \in r_a$ do
            $D_i = D_i \cup D_j$
        end
    end
```

priority queue technique, which can help individual robot negotiate with other robots and get an agreement efficiently.

3a) Selection Planning: In our scenario, we assume that each task has an ID representing its priority (level-4 in Fig. 3.2), the minimum number of robots required to perform the task, and a task duration (timeout). Also individual robot has an ID and Energy (battery level) (level-2 in Fig. 3.2). We assume all robots are homogeneous and do not implement the third level of needs (capability). However, for heterogeneous robotic systems, we should consider this priority as well.

For selection planning, we minimize the sum of the group's energy costs to get a reasonable grouping/partitioning. Then, according to the tasks' priority and requirement, we distribute the diverse tasks to different groups, abstracting it to *the linear partition problem*.

Through this process, we can ensure that every group can achieve that specific task. Since each robot computing this process in a distributed manner, the local result at the individual robot could potentially have conflicts with the results of other robots depending on the uncertainty in the data it possesses and receives. For instance, one robot might have been assigned to different groups by different robots in the Selection phase. To prevent such a situation, we initiate a negotiation mechanism. We use the priority queue before this process and sort this priority queue with different priority levels, until we get a unique priority queue for all the robots, which can then be used to perform non-conflicting selection planning.

3b) Formation Planning: Each robot will make a formation plan according to the selected plan. Here, each robot knows which group (task) it belongs to. To simplify the formation models, we assume the robots need to create a regular polygon surrounding the task assignment location (group's center). The initial point will be located at the North position and follow the clockwise order to arrange the other point assignments within the formation by minimizing the system utility as mentioned before (we use distance as utility cost and energy level for prioritizing the robot's needs).

Algorithm 2: Selection/Formation/Routing Negotiation

```
Input  : Unsorted priority queue $q_i$, potential collision queue $q_c$,
Output : Sorted priority queue $q_{si}$
if State == Selection/Formation/Routing then
    $q_n$ = hierarchical needs order queue;
end
if State == Routing then
    $tmp1$ = DCM($q_c$);
    for each item in Union-Find(tmp) do
        if $i \in item$ then
            $q_i = item$
        end
    end
end
if $q_n$ $! = NULL$ then
    $q_{si}$ = Sort $q_i$ with $q_n.first$;
    $tmp2$ = DCM($q_{si}$);
    while $Agreement(tmp2) == "conflict"$ do
        $q_{si}$ = Sort $q_i$ with $q_n.next$;
        $tmp2$ = DCM($q_{si}$);
    end
end
```

3c) Route Planning: Each robot computes the routes (path plan) using the local environment map of the sensor data and selects the shortest path getting to the goal point resulting from the Formation plan. Suppose each robot has two kinds of actions/behaviors that can be selected: one is moving at a constant speed(v), the other is stop. In case some robots have a conflict in the process of making the routing plan, they will create a priority queue with all conflicting robots $ID_{i...j}$ and share with the neighbors through local communication. Then, according to the Task and Energy priority of the robots, each robot in this queue will negotiate to decide on the corresponding actions on the robots that have conflicts. Until an agreement is reached, the priority queue is updated based on the needs hierarchy and solve the conflict.

4) Negotiation: Robots will compare the plans received from other group members with their own. For the Selection and Formation plan, the negotiation will be performed until the robots are assigned to only one group (in case of selection) or one position in the formation. Route planning involved creating a unique priority queue that avoids conflicts. Subsequently, each robot will reach an agreement on the priority queue and the corresponding plans. For example, in the selection and formation section, we use Low Energy (Low_E), which means the robots with lower battery levels get higher priority and similarly the High Energy $High_E$ law. We combine them with using Task-based priority queue (each task will have a specific priority in assigning the tasks), resulting in $T + High_E$ and $T + Low_E$. We also consider whether or not to consider the priority needs of conflicts for route planning (conflict case and No conflict case). We present the algorithmic representation of the negotiation process in Alg. 2.

Algorithm 3: Selection/Formation/Routing Agreement

```
Input  : Sorted priority queue list $Q$
Output : Execute the plan or negotiation again
initialization;
count = 1;
for each item in $Q$ do
    if $Q.first$ != $item$ then
        count++
    end
end
if count == 1 then
    return "end" and execute the corresponding "Atomic Operation";
else
    return "conflict".
end
```

For instance, in the Formation plan negotiation, we use distances between its local position and the task's polygon points in the boundary of the formation shape. Each robot communicates this distance vector with other robots in the same group assigned to this specific task. Each robot will compute a matrix (Level 4 Team needs - Utility cost) representing each robot's distances to all the polygon points in the specific task. Then it will use the unique queue order to select the corresponding distance until all the group member gets the specific task goal point as long as the priority queues of Task, Energy, and Safety are satisfied. For example, the low energy robot will have a high priority choosing the point closest to it, thereby reducing energy consumption.

5) Agreement and Execution: If all the robots' plans do not have conflict after the negotiation phase, they will have a final agreement per process (Selection/Formation/Routing). The algorithm for implementing the agreement process is presented in Alg. 3. After the agreement, each process is executed as and when necessary.

3.5 Experiments and evaluation

We design a simple scenario to implement SASS and distributed algorithms. In our scenarios, a group of swarm robots will cooperate to complete some tasks. Since the tasks are dynamically assigned, the robots need to change their plans and adapt to the new scenario to guarantee the group utility.

To simulate our framework, we chose to use the *Common Open Research Emulator* (CORE) network simulator Ahrenholz, 2010 since we are interested in implementing our algorithm in a network-based tool as CORE allows dynamic changes in the node/agent mobility and communication. We consider 20 robots in our simulations due to limitations in the CORE framework for an illustration of a single task assignment with 20 fully connected robots). In the evaluations, we consider only the lower three levels (Safety + Basic + Capability) in the agent (robot) needs hierarchy (see Figure 3.2) for aiding rigorous analysis and validation of our cooperation framework. Composition of higher level needs will be investigated in our future work.

We suppose each robot has different battery levels in the initial state, and every moving step will cost 0.1% energy. Also, every communication round and non-moving status will cost 0.01% and 0.04% energy, respectively. To simplify the visualization of the utility of the framework, we do not consider any obstacles. We design two scenarios – one to simulate static task assignments (all tasks are added at the initial stage) and another to simulate a dynamic task assignment (a task can be added anytime during the process).

We consider four combinations of priority: $High_E$ (High Energy), Low_E^{Num} (Low Energy + Task Priority Order), $T + High_E$ (Task Priority + High Energy), and $T + Low_E$ (Task Priority + Low Energy). For example, if we adopt a priority queue with task + low energy combination, the scenario would be to first address the emergency task and maintain robots in the field as long as possible. We intend to compare the utility and behaviors of the individual robot and the system with different priority combinations.

To compare our approach with a state of the art method, we implemented the algorithm called Collision-Aware Task Allocation (CATA) in Wu et al., 2019 in which the authors proposed a new method for addressing collision-aware task assignment problem using collision cone and auction-based bidding algorithms to negotiate the conflicts. Since our framework is distributed and does not have a central agent to manage the bidding process, we implemented the algorithm in Wu et al., 2019, which provides the rewards for each robot to each task location. The rewards are converted to a Utility matrix and fed to the negotiation mechanism in our framework. Therefore, we term this method as $CATA_U$ (Collision-aware Task ASsignment + Utility Matrix). Here, a robot first calculates the task's utility based on Wu et al., 2019, and it chooses the maximum utility task based on the low energy priority law. Therefore, we compare this method only with our Low_E priority law. The experiment demonstrations are available online[29].

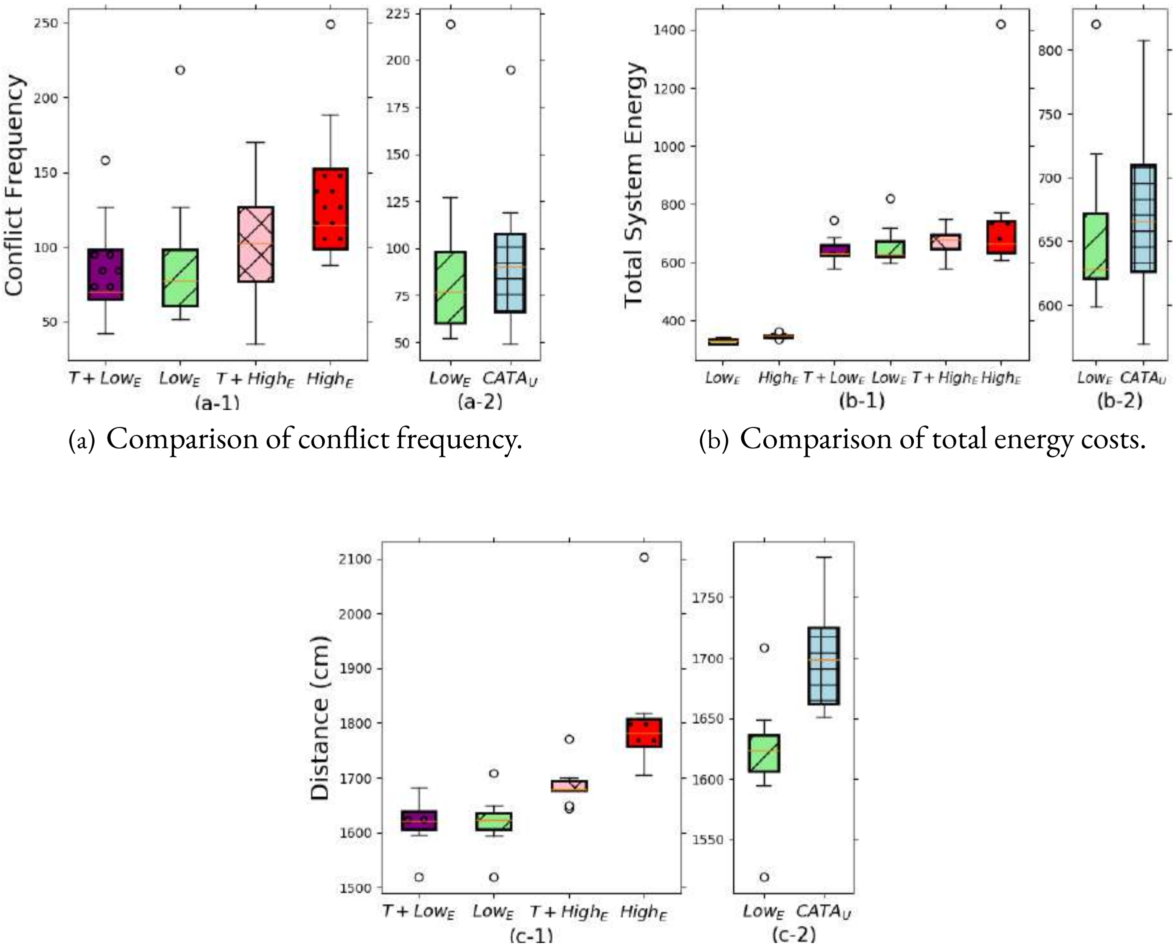
3.5.1 Static Multi-Task Assignments

We conduct ten simulation trials for each priority case in a static task assignment scenario. In every priority case, we use the same ten different initial battery levels sampled from a Gaussian distribution with a mean of 90% and a standard deviation of 10%.

Figure 3.6(a) and 3.6(b) shows the distance matrix results of four priority laws of conflict frequency with and without conflict negotiation costs comparing the CATA approach. In this experiment, the $T + Low_E$ priority combination had the best performance compared with other cases. At the same time, Figure 3.6(b) shows that every group almost cost one-third of the entire energy in the negotiation and agreement part of solving the conflicts. This means that more conflicts will lead to more negotiation rounds and corresponding energy consumption in communication. In Figure 3.6(c), the effect of each priority law in the total system distance is also demonstrated for our finding that different needs and priorities at the individual agent level will lead to various global performances.

In the priority law of Low_E, our method had fewer conflicts than the $CATA_U$ method. We believe this is because prioritization of basic needs in our approach aims to avoid conflicts at the Formation planning stage. On the other hand, $CATA_U$ considers the conflicts only at the Route planning stage, which would leave more room for conflicts if the task polygon points are not efficiently assigned in the formation stage itself.

[29] Experiment demonstration video is available at the website: https://hero.uga.edu/research/sass



(a) Comparison of conflict frequency.

(b) Comparison of total energy costs.

(c) Comparison of total travel distance.

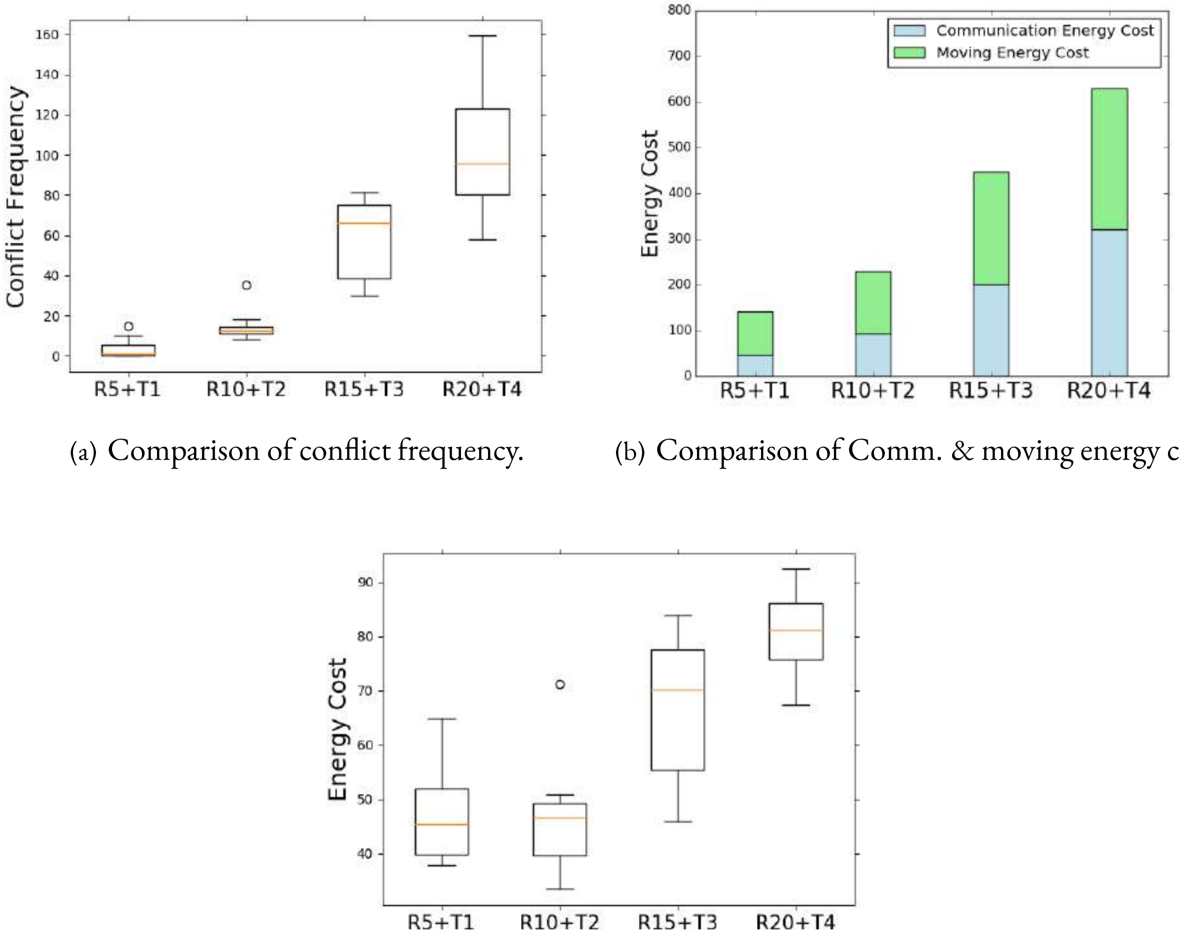
Figure 3.6: Experiments on static task assignments with 20 robots and 3 tasks.

3.5.2 Scalability and Complexity

To verify SASS's scalability and the complexity of the MRS cooperation, we design four scales of robots' team implementing in different numbers of tasks: $R5 + T1$, $R10 + T2$, $R15 + T3$, $R20 + T4$. For example, $R10$ means ten robots in the system, and $T3$ means three different tasks are assigned.

Through Figure 3.7(a) [30], we can notice that as the number of robots and the task complexity increase, the entire system's conflicts (conflict frequency) rise rapidly. The proportion of communication energy cost compared with the moving energy cost also increase with the increase in scale (see Figure 3.7(b)).

[30] **Here**, R15 means 15 Robots and T4 means 4 Tasks for example.



(a) Comparison of conflict frequency.

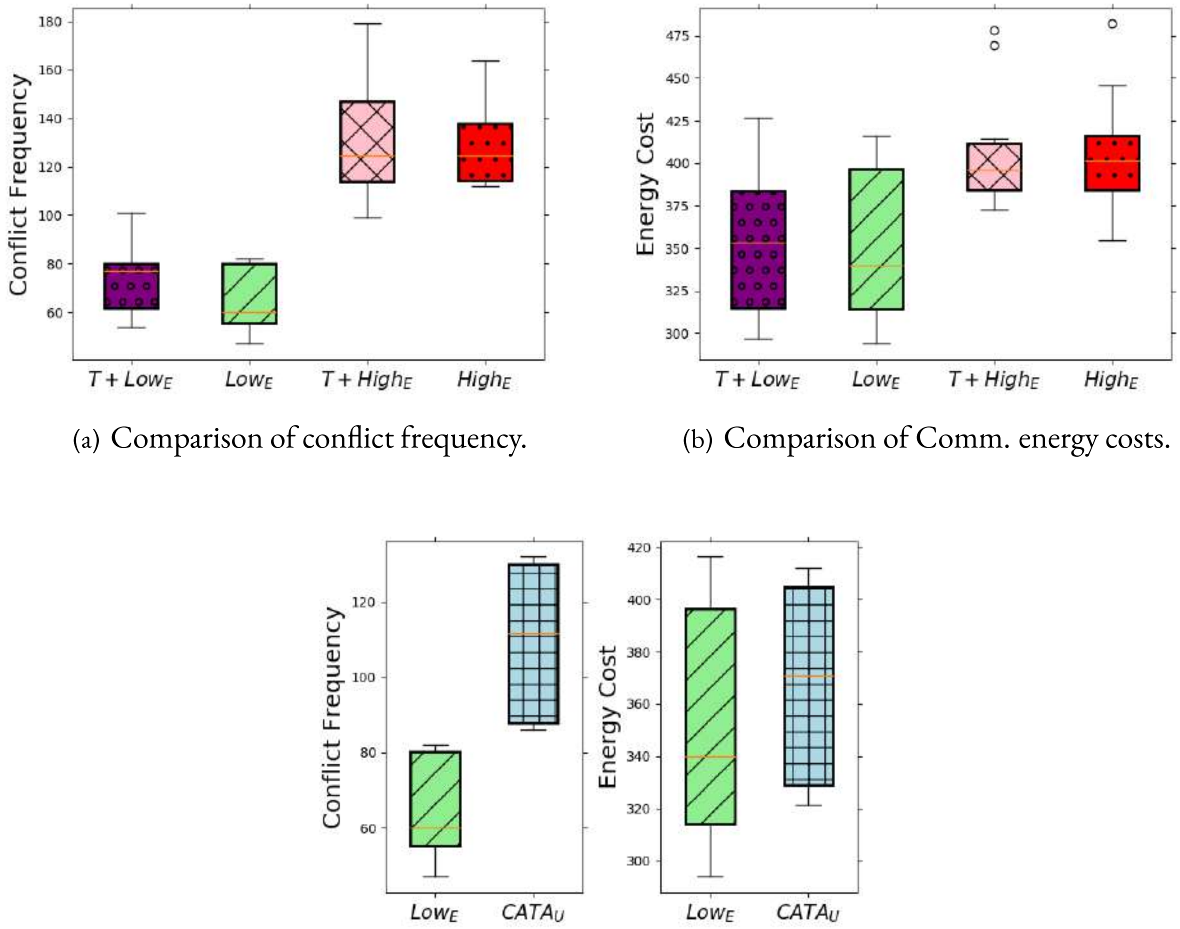
(b) Comparison of Comm. & moving energy costs.

(c) Comparison of per task Comm. energy costs.

Figure 3.7: Experiments on Scalability and Complexity in Static Task Assignments

This points to the fact that the whole system spends more energy and time negotiating and cooperatively converging to a specific agreement. So considering the average communication energy cost per task (see Figure 3.7(c)), if the task complexity is higher in some specific scenarios or the environment is more unstructured and unpredictable, an individual agent will spend more energy and time in communication to fulfill the tasks.

From another perspective, the fully distributed communication graph is inefficient in a large scale system of coordinating robots, especially in the swarm



(a) Comparison of conflict frequency.

(b) Comparison of Comm. energy costs.

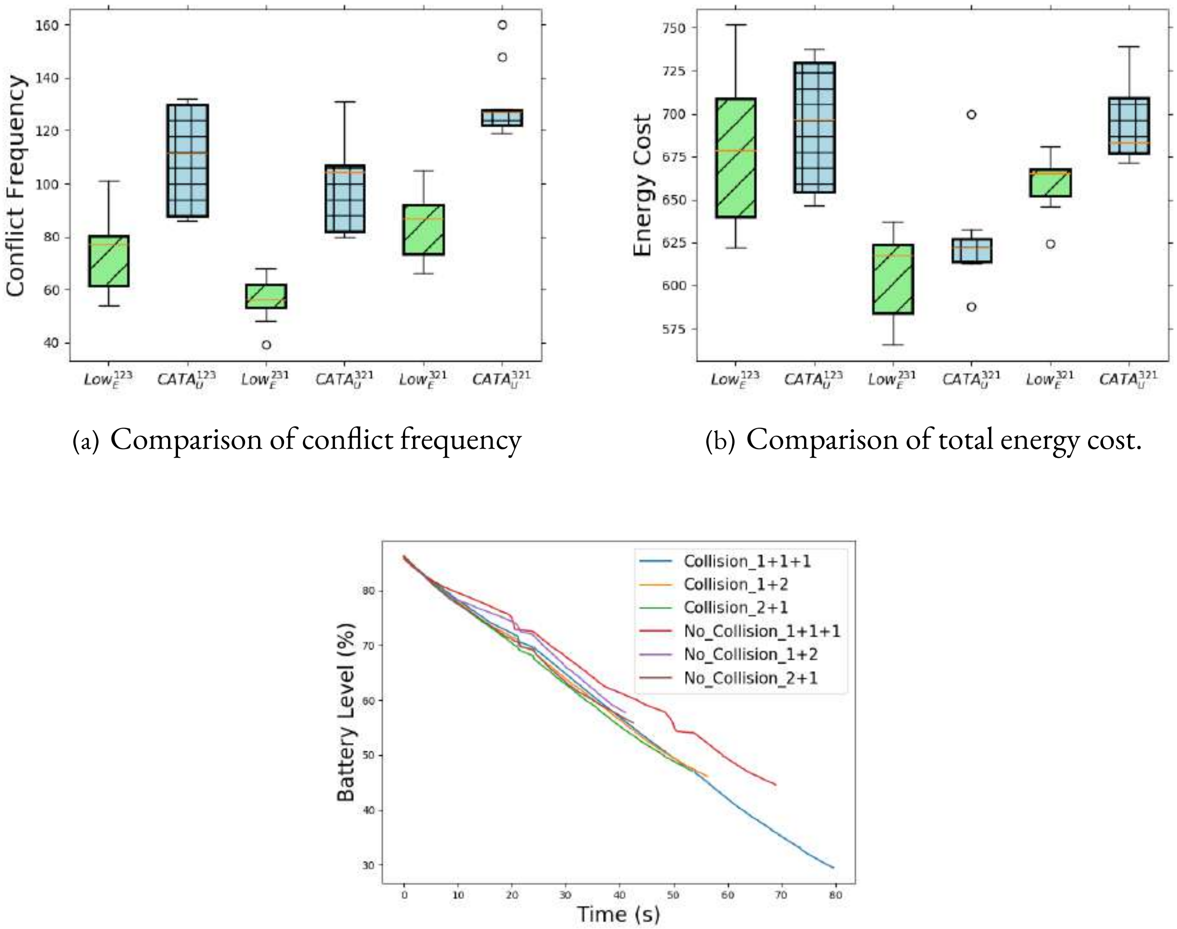
(c) Comparison of Low_E and CATA_U.

Figure 3.8: Experiments on the Impact of Different Priority Laws in Static Task Assignments.

robots. Therefore, designing a proper communication architecture to adapt to a particular scale of agents' group and complexity scenarios is also an important and challenging problem that should be investigated further, which is an avenue for future work.

3.5.3 Impact of Different Priority Laws at Different Levels

If an individual agent has different needs or motivations, it might present various conduct, leading to the entire system displaying different performance. To



(a) Comparison of conflict frequency

(b) Comparison of total energy cost.

(c) Comparison of total battery level.

Figure 3.9: Experiments on the impact of different priority laws and dynamic task assignments with 20 robots and 3 tasks.

verify this hypothesis, we use 20 robots having the same initial battery levels based on four different priority laws comparing with the distance matrix and CATA as we discussed above. We conduct ten trials with different initial battery levels sampled from a Gaussian distribution with a mean 90% and standard deviation 30% to represent heterogeneity in the energy capacities of the robots.

Figure 3.8(a) shows that implementing different priority laws causes the whole system to exhibit different energy use. As we can see, the system has better performance in $T + Low_E$ and Low_E priority laws than $T + High_E$, $High_E$ priority laws. This is because the former two lead to fewer conflicts than the latter two. We can confirm these observations through Figure 3.8(b) and 3.8(c).

We also shuffle the task priorities (e.g., the priority of T1 > T2 > T3 in the superscript 123) to simulate the variation in the third level of the needs hierarchy (Capability/Requirements) and the results are shown in Figure 3.9(a) and Figure 3.9(b) [31]. We can see that our approach's distance matrix always has better performance than the CATA under the same conditions in terms of conflict resolution and system energy.

In practical applications, an intelligent agent might dynamically change its needs or motivation according to the situations, especially in the adversarial or unpredictable environment. This may lead to chaos in the system resulting in higher energy costs and loss of system utility. In the MRS design, letting the individual agents select suitable laws for coordination while guaranteeing the optimal system utility is also an exciting and challenging avenue for future work.

3.5.4 Dynamic Multi-Task Assignments

In the dynamic multi-tasks scenario, we design three kinds of dynamic tasks. One is three tasks added sequentially after every task is completed $(1 + 1 + 1)$. The rest of the two cases are two tasks appearing at the start $(2 + 1)$ and the end $(1 + 2)$ in the entire process. Also, we combine this with a conflict or no conflict cases.

Table 3.2: Energy Level Comparison in Dynamic Task Assignments

Tasks Style	Priority	conflict	Max	Min	Mean
1+1+1	High$_E$	√	69.42	55.52	64.18
1+1+1	Low$_E$	√	63.09	45.70	56.00
1+1+1	T+High$_E$	√	70.04	56.75	63.24
1+1+1	T+Low$_E$	√	63.25	49.18	56.62
1+2	T+Low$_E$	√	45.97	31.78	39.60
2+1	T+Low$_E$	√	44.69	31.66	39.07
1+1+1	T+Low$_E$	-	49.88	29.48	41.20
1+2	T+Low$_E$	-	32.70	23.14	28.50
2+1	T+Low$_E$	-	38.53	24.72	30.36

[31] (Left and Center) Comparison of conflict frequency and energy cost based on the different task's priority in statics task assignments. E.g., In Low_E^{123}, the priority of T1 > T2 > T3. (Right) Comparison of the total battery level of the system measured in points combining all robot energy percentage levels in dynamic task assignments.

The first experiment we consider using four different priority laws and 20 robots implementing the $1+1+1$ scenario (see Table 3.2). Here, the initial energy level and position for each priority laws were set the same. We can observe that the $T + Low_E$ and Low_E combinations achieved the best system utility as in the static assignment cases.

In the second experiment, we evaluate the impact of the conflict negotiation process by comparing the energy costs of three different tasks with and without considering conflicts. In Figure 3.9(c), we notice that the negotiation cost occupies a large part in the entire system costs (hence, the higher battery level consumed when collisions are considered). We can also observe that the difference between each combination's negotiation energy cost reflects the environment's unstructured level, which means the difference will increase in the more chaotic and dynamic considerations.

The results are implicit that the individual agent needs to adopt suitable priority laws and corresponding plans optimizing its current behaviors to adapt to the dynamic environments. On the other hand, the changes in individual needs lead to the system's global performance differences.

3.6 Conclusion

Our work introduces a novel SASS framework for cooperation heterogeneous multi-robot systems for dynamic task assignments and automated planning. It combines robot perception, communication, planning, and execution in MRS, which considers individual robot's needs and action plans and emphasizes the complex relationships created through communication between the robots. Specifically, we proposed *Robot's Needs Hierarchy* to model the robot's motivation and offer a priority queue in a distributed *Negotiation-Agreement Mechanism* avoiding plan conflicts effectively. Then, we provide several *Atomic Operations* to decompose the complex tasks into a series of simple sub-tasks. The proposed solution is evaluated through extensive simulations under different static and dynamic task scenarios. The experimental analysis showed that the needs-based cooperation mechanism outperformed state-of-the-art methods in maximizing global team utility and reducing conflicts in planning and negotiation.

Chapter 4

Needs-driven Tasks Assignment

This chapter will discuss the hierarchical needs-driven diverse behaviors in MAS, especially the role of heterogeneity in a MRS applied to robot-aided urban search and rescue (USAR) missions. Then, we will formalize the hierarchical needs-driven model and prove that the needs-driven cooperation in a heterogeneous robot system enables higher group utility than a homogeneous robot system from the theoretical analysis. Through performing simulation experiments, the results verify the proposed needs-driven collaboration and show that the heterogeneous multi-robot cooperation can achieve better performance and increase system robustness by reducing uncertainty in task execution. After that, we will discuss the extension of the application in human-robot teaming.

4.1 Hierarchical needs-driven diverse behaviors

In nature, from cell to human, all intelligent agents represent different kinds of hierarchical needs such as the low-level physiological needs (food and water) in microbe and animal; the high-level needs self-actualization (creative activities) in human being Maslow, 1943. Different levels of needs stimulate all agents to adopt diverse behaviors achieving certain goals or purposes and satisfying their various needs in the natural world. They also can be regarded as *self-organizing behaviors* which keep on interacting with the environments and exchanging information and energy to support the system's survival and development. In other words, elements or agents of *self-organizing systems* [32] can manipulate or organize others of the same system in a way that stabilizes either structure or function of the whole against external fluctuations Yates, 2012. Especially the process of self-organization is often achieved by growing the internal space-

[32] **Self-organizing systems** (SOS) refers to the ability of a class of systems to change their internal structure and/or their function in response to external circumstances. Banzhaf, 2009

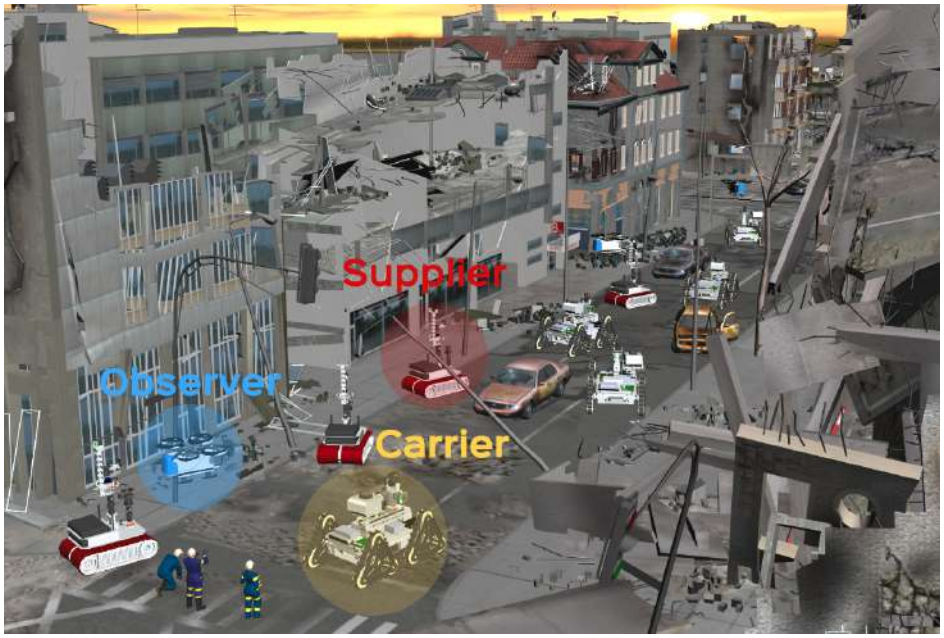


Figure 4.1: Illustration of an integrated team of robots (UGVs + UAVs) and human cooperatively working together in a post-earthquake rescue mission.

time complexity of a system and results in layered or hierarchical structures or behaviors Banzhaf, 2009.

To some extent, the cooperative MAS/MRS is equal to the *self-organizing systems*. Specifically, based on their specific needs, intelligent agents can cooperate to achieve a task gaining rewards or against the adversaries decreasing the threat, which benefits the entire group development or utilities and guarantees the individual needs and interests. Let us take the robot-aided urban search and rescue (USAR) missions as an example.

Rescue missions can be regarded as life-saving, delivering valuable properties, and tackling necessary facilities in disaster or emergency scenarios, including complex, hazardous, uncertain, unstructured, dynamical changing, and adversarial environments. Multi-Robot System (MRS) working in such situations requires rapid response, high adaptation, and strong robustness, reducing the losses in the post-disaster scenarios. Research in robot-aided USAR aims to increase the mission success rate, improve execution efficiency, and minimize system cost during the rescue missions. Fig. 5.1 illustrates an example real-world use-case of MRS in a post-earthquake scenario, where we represent teams of three different robot types - *Carrier*, *Supplier*, and *Observer* - aiding the first responders in close collaboration.

Also, disasters are defined as discrete meteorological, geological, or man-made events that exceed local resources to respond and contain R. R. Murphy et al., 2016. From the robot's needs and motivations perspective, we can classify adversaries into two general categories in disaster or adversarial environments. One is Intentional (such as enemy or intelligent opponent agent, which consciously and actively impairs the MAS needs and capabilities), and the other is Unintentional (like obstacles and weather, which unaware and passively threaten MAS abilities) adversary. We are specifically interested in the MRS collective tackling the *unintentional adversary* in hazardous and disaster scenarios. So the environment models for rescue missions are grounded in two different aspects: individual perception and data sharing across the robots.

Considering individual perception, cooperavtive MRS emphasize cooperation among heterogeneous groups of robots Stone and Veloso, 2000. Each robot class might have different sensors and capabilities to perceive and interact with the environment and corresponding actuators to execute their action. Individual robots present their observations from different angles describing the partial part in the global map. Regarding system data sharing, each robot in the current group needs to update its situation awareness from other group members' information. It can not only help in collectively building a global map Rizk et al., 2019 but also be a foundation for communication between the agents to achieve *consensus*.

As an artificial intelligence agent – robot, to organize its behaviors and actions, we introduced the needs hierarchy of robots to help MRS build cooperative strategies considering their individual and common needs. Specifically, the robots possess the following order of needs hierarchy: Safety needs (avoid collisions, safe environment, etc.); Basic needs (Energy, time, mobility, etc.); Capability needs (task-specific such as carry or supply resources); Teaming needs (enhancing group utility and group survival); and Learning needs (self-upgrade and evolution).

Since the robot needs to rescue the victims from the disaster or cooperate with people to fulfill rescue missions together, the robot's lowest level needs should guarantee human safety and security. This kind of condition reflex or self-reactive behavior in robots can be represented as fundamental control issues like collision avoidance. After satisfying the safety needs, the robot requires enough basic needs, like battery, oil, to support executing relative operations. Then, by comparing their capabilities and the task requirements, they will select how to cooperate maximizing the success rate in rescue missions, and optimize or sub-optimize individual and system *utility*.

To fulfill a high level needs satisfying individual or group's *expectation utilities*, different categories of robots consider working as one or several teams to maximize corresponding utilities or rewards efficiently. When assigned with new rescue tasks or encounter emergency events like some group members run out of battery, robots need to re-organize the group adapting to the current situation minimizing the cost and loss.

In rescue missions, we consider the **group's utility** as the number of lives (victims) or valuable properties saved and rescued as much as possible in a limited time. In the entire process, robots need to consider exploring the uncertain area, tackling the *unintentional adversaries* like obstacles, wind, rain, and so on, repairing necessary facilities, treating injurers, carrying victims and properties to a safe place.

4.2 Related works and projects

An intelligent agent is a physical (robot) or virtual (software program) entity that can autonomously perform actions on an environment while perceiving this environment to accomplish a goal Russell and Norvig, 2002. Cooperation in multiple intelligent agents (robots) working in a disaster environment is a interesting and challenging problem Jorge et al., 2019; R. R. Murphy et al., 2016. Most research focus on the problems of environmental monitoring Bayat et al., 2017; Byrne et al., 2012; Marques et al., 2015, structure inspection Lattanzi and Miller, 2017; Moud et al., 2018, navigation and control Ashrafiuon et al., 2010; Campbell et al., 2012; Fossen, 2000 and higher-level autonomy Schiaretti et al., 2017; F. Thompson and Guihen, 2019. Also, there are various advancements in rescue robotics through the development of heterogeneous robot teaming methods in disaster response scenarios De Cubber et al., 2013; Kruijff-Korbayová et al., 2015; Marconi et al., 2013; Nourbakhsh et al., 2005, disaster detection Fornai et al., 2016; D. Liu et al., 2016, disaster monitoring Guerrero-González et al., 2016; Vasilijevic et al., 2017, target tracking Fahad et al., 2017; Rathour et al., 2015, victims detection Cardona and Calderon, 2019, and reinforcement learning based semi-autonomous controller for urban search and rescue missions Magid et al., 2020.

Considering grouping robots with various capabilities cooperating to pursue specific goals (rescue missions), less literature study the integration of organizing agents' behaviors, solving the conflicts, optimizing system utility, and boosting system adaptability and robustness for the entire group Rizk et al., 2019; G.-Z. Yang et al., 2018. On the other hand, there is little research done from the agent's needs perspective studying individual interaction and behaviors for

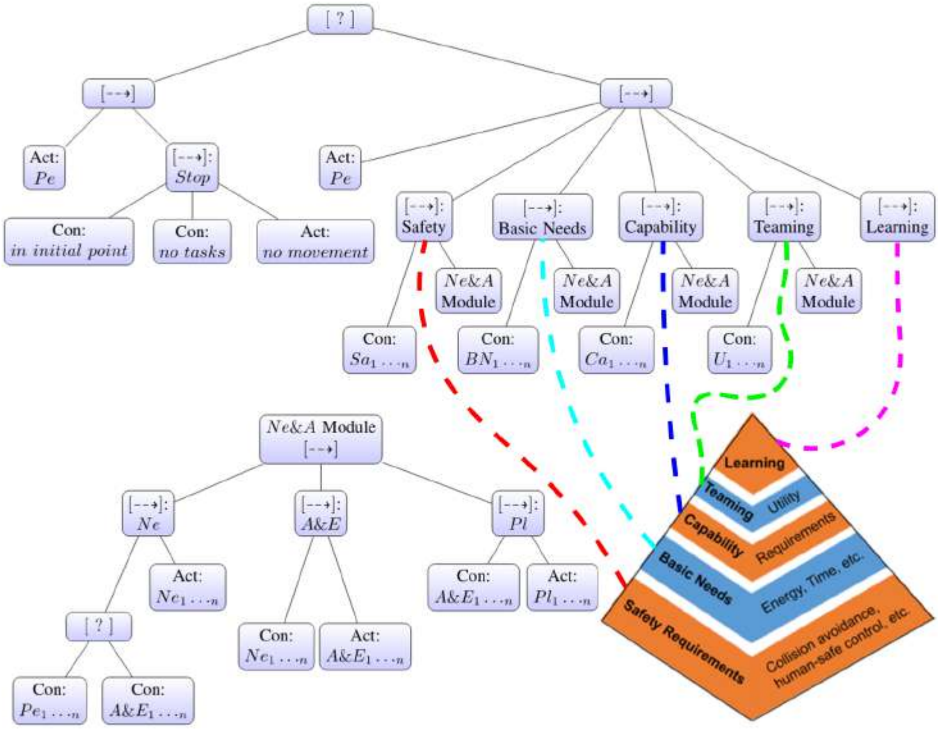


Figure 4.2: Behavior Tree representing *Agent (robot) Needs Hierarchy* at each agent in *SASS*

system performance (group utility) and global actions in MRS, especially in disaster robotics Geihs, 2020; Q. Yang and Parasuraman, 2020b, 2020c.

To address those gaps, we encode the *agent (robot) needs hierarchy* in the robot automated planner, where we represent complex relationships between different types of robots through their immediate needs and motivations. It helps the system to balance and optimize the utilities between the individual and the whole group. Then we analyze the MRS group performance by theoretically deriving and comparing the group utility and their energy cost applied to a USAR mission. Also, the *human-robot* mixed teaming by combining human and robot needs will benefit from the study. More importantly, it can improve system adaptability and solve more complex tasks. On the other hand, it helps individual self-upgrade and self-evolution of the whole system through *adaptation learning* from interaction and experience between robots and humans. Figure 4.2 [33] shows agent *needs-driven SASS* mechanism represented through a *Behavior Tree (BT)* structure Colledanchise and Ögren, 2018.

[33] **Here**, [?] - Selector Node, [--→] - Sequence Node, Con - Conditions, Act - Actions, Pe - Perception, Sa - Safety, BN - Basic Needs, Ca - Capability, U - Utility, Pl - Plan, Ne - Negotiation, $A\&E$ - Agreement and Execution.

4.3 Hierarchy needs-driven model formalization

This section first formalizes the rescue problem and uses mathematical approaches to prove our hypothesis that cooperation in heterogeneous robot system produce better performance in general than the cooperation limited to a homogeneous robot system, with rescue mission as an example application domain.

Consider the following example. Supposing a group of heterogeneous robots executes the search and rescue mission in a post-disaster scenario. The robot's categories can be generally classified as follows.

- **Carrier:** Their main function is carrying injurers and valuable properties from hazardous areas to shelter.
- **Supplier:** Providing various resources for rescue missions such as medicine, food, repairing robots, rescue devices, communication support, and so forth.
- **Observer:** They are good at surveying and acquiring real-time and dynamical rescue information from the disaster environment.

4.3.1 Problem Statement

As discussed in Sec. 4.1, we assume that the number of *Carrier*, *Supplier* and *Observer* are x, y and z ($x, y, z \in Z^+$), respectively. We define the individual capability space according to the robot needs model through the below equations.

$$Carrier := C_C(v_c, com_c, sen_c, eng_c, res_c, cap_c); \tag{4.1}$$

$$Supplier := C_S(v_s, com_s, sen_s, eng_s, res_s, cap_s); \tag{4.2}$$

$$Observer := C_O(v_o, com_o, sen_o, eng_o, res_o, cap_o). \tag{4.3}$$

Here,

- v represents agent's max velocity;
- com and sen represent the range of agent's communication and sensing separately;
- eng represents agent's energy level;
- res represents the amount of rescue resource which agent can provide;
- cap represents agent's the capacity level.

Since each type of robot specialize in different capability, we can assume Eqs. (4.4), (4.5), (4.6), (4.7), (4.8), (4.9) showing the dominance of each robot type (denoted with subscripts c, s, o to represent carrier, supplier, and observer robots, respectively) in different capabilities in terms of sensing and communication ranges, energy level capacities, etc.

Robot Safety Needs:

$$com_o \gg com_s \approx com_c; \tag{4.4}$$

$$sen_o \gg sen_s \approx sen_c; \tag{4.5}$$

$$v_o > v_s \approx v_c; \tag{4.6}$$

Robot Basic Needs:

$$eng_c > eng_s \gg eng_o \tag{4.7}$$

Robot Capabilities for Rescue Mission Requirement:

$$res_s \gg res_c > res_o; \tag{4.8}$$

$$cap_c \gg cap_s > cap_o; \tag{4.9}$$

Supposing rescue mission T has requirement space $TC = (C_1, C_2, ..., C_m)$, $m \in Z^+$, where C_i represents different capabilities expected required to achieve a given global task and m is the capacity of the required to satisfy the tasks. We assume that the heterogeneous group capabilities for rescue mission requirements is $C_G = (..., C_{C_x}, ..., C_{S_y}, ..., C_{O_z})$ and group members' expected round-trips within $t \leq t_n$ is $m(m_1, ..., m_k)$ $k \in Z^+$, where $k = x + y + z$. U and t_n represent the rescue mission's *Group Utility* and mission time restriction, respectively. Then, we can regard rescue problem as an optimization problem Eq. (4.10), which means that in the limited time, fulfilling a rescue mission maximum its *Expectation Utility* based on certain requirements.

$$\begin{aligned} &\underset{C_G}{\arg\max} \quad \mathbb{E}(U(t_n, m \cdot C_G)); \\ &subject\ to \quad \sum_{d=1}^{x} \sum_{e=1}^{y} \sum_{f=1}^{z} m \cdot C_G \geqslant TC,\ d, e, f \in Z^+. \end{aligned} \tag{4.10}$$

To simplify our model, we just consider one specific rescue mission and n identical obstacles distributed in an uncertain disaster environment randomly. The encountering obstacles times X for each agent follow *Poisson Distribution* Eq. (4.11) and λ represents as Eq. (4.12) (c and sen are corresponding coefficient and area of sensing range).

$$X \sim P(\lambda); \tag{4.11}$$
$$\lambda = \frac{cn}{sen} \tag{4.12}$$

And we assume that the average time and energy cost for individuals tackling each obstacle are t_c and e_c, respectively. The distance between the central point of the initial group and rescue position is l. We also assume that agent energy costs mainly consist of traveling, tackling the obstacles, and fulfilling the rescue task Parasuraman et al., 2012. The traveling energy cost can be regarded as constant e_t, which is proportional to l.

Through Eqs. (4.11) and (4.12), we can easily calculate the expectation of time encountering obstacles as Eq. (4.13). Without considering obstacles, individuals coming to rescue position and returning to initial point energy cost are $\frac{2l}{v}$. Then, considering the obstacles, we estimate the expected time cost per round as Eq. (4.14).

$$\mathbb{E}(X) = \sum_{i=0}^{+\infty} iP(X = i) = \lambda = \frac{cn}{sen}; \tag{4.13}$$
$$\mathbb{E}(T) = \mathbb{E}(\frac{2l}{v} + 2t_c X) = \frac{2l}{v} + \frac{2t_c cn}{sen} \tag{4.14}$$

4.3.2 Theoretical Evaluation

In this section, we generally classify the rescue team as two different categories: *Homogeneous* and *Heterogeneous*. A homogeneous robot system means the robots in that group are of the same type with same capabilities (Eq. (4.9)), whereas robots in a heterogeneous group will have different capabilities Stone and Veloso, 2000; Twu et al., 2014. We assume that each agent's sensing range equal to its communication range for the sake of simplicity in analysis. Then, we use mathematical approaches to analyze and compare their performance as follows:

Homogeneous Cooperation In this scenario, we suppose that the number of *Carrier*, *Supplier* and *Observer* are equal Eq. (4.15). According to Eq. (4.14), the time homogeneous group per round can be represented as Eq. (4.16).

$$x = y = z = 3m, \ m \in Z^{+}; \tag{4.15}$$
$$\mathbb{E}(T_h) = \frac{2l}{v} + \frac{2t_c cn}{3m \times sen} = \lambda_h \tag{4.16}$$

Considering rescue mission's tackling time equal to one unit time, the entire expectation rescue per round time is $\mathbb{E}(T_h + 1)$. Then, we can calculate $\mathbb{E}(\frac{1}{T_h+1})$ as Eq. (4.17).

$$\begin{aligned}
\mathbb{E}(\frac{1}{T_h+1}) &= \sum_{k=0}^{+\infty} \frac{1}{k+1} P(T_h = k) \\
&= \sum_{k=0}^{+\infty} \frac{1}{k+1} \frac{\lambda_h^k e^{-\lambda_h}}{k!} \\
&= \frac{1}{\lambda_h} \sum_{k=0}^{+\infty} \frac{\lambda_h^{k+1} e^{-\lambda_h}}{(k+1)!} \\
&= \frac{1}{\lambda_h} \sum_{k=0}^{+\infty} P(T_h = k+1) \\
&= \frac{1}{\lambda_h} \sum_{k=1}^{+\infty} P(T_h = k) \\
&= \frac{1 - P(T_h = 0)}{\lambda_h} = \frac{1 - e^{-\lambda_h}}{\lambda_h}
\end{aligned} \tag{4.17}$$

Finally, we estimate the expected number of rounds for this homogeneous group in the rescue mission with a limited time t_n as Eq. (4.18).

$$\mathbb{E}(\frac{t_n}{T_h+1}) = \frac{t_n(1 - e^{-\lambda_h})}{\lambda_h} \tag{4.18}$$

We are supposing rescuing each agent cost one point supplement, energy, and space, respectively. We can estimate the sum of *Expectation Utility* – the amount of rescued agents U_h in all rounds and the total energy cost E_h in group of *Carrier*, *Supplier* and *Observer* as Eq. (4.19) and (4.20) respectively.

a. Carrier/Supplier/Observer expectation amount of rescued agents

$$\begin{aligned}
\mathbb{E}(U_{hc/s/o}) &= \frac{t_n(1 - e^{-\lambda_{hc/s/o}})}{\lambda_{hc/s/o}} 3m \times res_c/cap_s/cap_o, \\
\lambda_{hc/s/o} &= \frac{2l}{v_{c/s/o}} + \frac{2t_c cn}{3m \times sen_{c/s/o}}
\end{aligned} \tag{4.19}$$

b. Carrier/Supplier/Observer expectation energy cost

$$\begin{aligned}\mathbb{E}(E_{hc/s/o}) =& e_t + \frac{2cn}{3m \times sen_{c/s/o}} e_c + \\ & \frac{t_n(1-e^{-\lambda_{hc/s/o}})}{\lambda_{hc/s/o}} 3m \times res_c/cap_s/cap_o\end{aligned} \tag{4.20}$$

Heterogeneous Cooperation For heterogeneous cooperation, we consider four different combinations as follow:

- (*x Carriers, y Suppliers*), $x + y = 3m$;
- (*x Carriers, z Observers*), $x + z = 3m$;
- (*y Suppliers, z Observers*), $y + z = 3m$;
- (*x Carriers, y Suppliers, z Observers*), $x + y + z = 3m$;

Then, we estimate the expected amount of rescued agents U_e and energy cost E_e for each group.

a. (x Carriers, y Suppliers)

In this scenario, we consider *Carrier* and *Supplier* have the similar sensing range, and they both have enough energy (*Basic Needs*) to support the entire rescue mission. So we can present $\mathbb{E}(U_{e1})$ and $\mathbb{E}(E_{e1})$ as Eq. (4.21) and (4.22).

$$\begin{aligned}\mathbb{E}(U_{e1}) =& \frac{t_n(1-e^{-\lambda_{e1}})}{\lambda_{e1}} \times ((x \times cap_c + y \times cap_s) \cap \\ & (x \times res_c + y \times res_s)),\end{aligned}$$

$$\lambda_{e1} = \frac{2l}{v_c} + \frac{2t_c cn}{3m \times sen_c} \tag{4.21}$$

$$\begin{aligned}\mathbb{E}(E_{e1}) =& e_t + \frac{2cn}{3m \times sen_c} e_c + \\ & \frac{t_n(1-e^{-\lambda_{e1}})}{\lambda_{e1}} \times ((x \times cap_c + y \times cap_s) \cap \\ & (x \times res_c + y \times res_s))\end{aligned} \tag{4.22}$$

b. (x Carriers, z Observers)

Here, we assume the entire group's velocity adapt *Carriers'* speed. Similarly, we can express $\mathbb{E}(U_{e2})$ and $\mathbb{E}(E_{e2})$ as Eq. (4.23) and (4.24),

$$\begin{aligned}\mathbb{E}(U_{e2}) =& \frac{t_n(1-e^{-\lambda_{e2}})}{\lambda_{e2}} \times ((x \times cap_c + z \times cap_o) \cap \\ & (x \times res_c + z \times res_o)),\end{aligned}$$

$$\lambda_{e2} = \frac{2l}{v_c} + \frac{2t_c cn}{x \times sen_c + z \times sen_o} \tag{4.23}$$

$$\begin{aligned}\mathbb{E}(E_{e2}) =& e_t + \frac{2cn}{x \times sen_c + z \times sen_o} e_c + \\ & \frac{t_n(1 - e^{-\lambda_{e2}})}{\lambda_{e2}} \times ((x \times cap_c + z \times cap_o) \cap \\ & (x \times res_c + z \times res_o))\end{aligned} \tag{4.24}$$

c. (y Suppliers, z Observers)
$\mathbb{E}(U_{e3})$ and $\mathbb{E}(E_{e3})$ as Eq. (4.25) and (4.26),

$$\begin{aligned}\mathbb{E}(U_{e3}) =& \frac{t_n(1 - e^{-\lambda_{e3}})}{\lambda_{e3}} \times ((y \times cap_s + z \times cap_o) \cap \\ & (y \times res_s + z \times res_o)),\end{aligned}$$

$$\lambda_{e3} = \frac{2l}{v_c} + \frac{2t_c cn}{y \times sen_s + z \times sen_o} \tag{4.25}$$

$$\begin{aligned}\mathbb{E}(E_{e3}) =& e_t + \frac{2cn}{y \times sen_s + z \times sen_o} e_c + \\ & \frac{t_n(1 - e^{-\lambda_{e3}})}{\lambda_{e3}} \times ((y \times cap_s + z \times cap_o) \cap \\ & (y \times res_s + z \times res_o))\end{aligned} \tag{4.26}$$

d. (x Carriers, y Suppliers, z Observers)
$\mathbb{E}(U_{e4})$ and $\mathbb{E}(E_{e4})$ as Eq. (4.27) and (4.28).

$$\begin{aligned}\mathbb{E}(U_{e4}) =& \frac{t_n(1 - e^{-\lambda_{e4}})}{\lambda_{e4}} \times \\ & ((x \times cap_c + y \times cap_s + z \times cap_o) \cap \\ & (x \times res_c + y \times res_s + z \times res_o)),\end{aligned}$$

$$\lambda_{e4} = \frac{2l}{v_c} + \frac{2t_c cn}{x \times sen_c + y \times sen_s + z \times sen_o} \tag{4.27}$$

$$\begin{aligned}\mathbb{E}(E_{e4}) =& e_t + \frac{2cn}{x \times sen_c + y \times sen_s + z \times sen_o} e_c + \\ & \frac{t_n(1 - e^{-\lambda_{e4}})}{\lambda_{e4}} \times \\ & ((x \times cap_c + y \times cap_s + z \times cap_o) \cap \\ & (x \times res_c + y \times res_s + z \times res_o))\end{aligned} \tag{4.28}$$

4.3.3 Comparative Analysis

After above discussion, in this section, we first compare the performances between (*Homogeneous vs Homogeneous*), (*Heterogeneous vs Heterogeneous*) and (*Homogeneous vs Heterogeneous*), then analyze the exiting of optimal or suboptimal solution for heterogeneous cooperation system in rescue mission. In order to simplify calculation, we assume Eq. (4.29) and also regard *Carrier* and *Supplier* have the similar sensing range, and the sensing range of group *Observer* approaches infinity.

$$res_c = cap_s = cap_o = res_o = k,\ \ k \in Z^+ \tag{4.29}$$

a. Homogeneous vs Homogeneous

Comparing the expected amount of rescued agents between those groups can be represented as Eq. (4.30).

$$\mathbb{E}(U_{hc}) : \mathbb{E}(U_{hs}) : \mathbb{E}(U_{ho}) = 1 : 1 : \frac{\lambda_{hc}(1 - e^{-\lambda_{ho}})}{\lambda_{ho}(1 - e^{-\lambda_{hc}})} \tag{4.30}$$

Also, we can compare the group expectation energy cost of *Carrier* and *Supplier*, *Carrier* and *Observer* and *Supplier* and *Observer* as Eq. (4.31) and (4.32) respectively.

$$\mathbb{E}(E_{hc}) - \mathbb{E}(E_{hs}) = 0 \tag{4.31}$$

$$\begin{aligned} \mathbb{E}(E_{hc}) - \mathbb{E}(E_{ho}) = \mathbb{E}(E_{hs}) - \mathbb{E}(E_{ho}) = \\ \frac{2cn}{3m \times sen_c} e_c + 3mkt_n(\frac{1 - e^{-\lambda_{hc}}}{\lambda_{hc}} \\ - \frac{1 - e^{-\lambda_{ho}}}{\lambda_{ho}}) \end{aligned} \tag{4.32}$$

Through the above discussion, if we assume that *Observers* also does not concern about their energy cost (*Basic Needs*) in the entire rescue mission, they will have the best performance comparing with other groups. Actually, in reality, the energy level and consumption rate of *Observer*, like the drone, are much lower and faster than *Carrier* and *Supplier* correspondingly, which means that *Observer* need to waste lots of time to charge. Considering this issue, we assume that these three groups have a similar performance generally to simplify our calculation.

b. Heterogeneous vs Heterogeneous

Similarly, considering involving *Observers* in the group, the entire group sensing range approach infinity. And according to the assumption Eq. (4.4), (4.5), (4.8) and (4.9), we can estimate the heterogeneous comparison of the expectation amount of rescued agents as Eq. (4.33).

$$\begin{aligned}\mathbb{E}(U_{e1}) : \mathbb{E}(U_{e2}) : \mathbb{E}(U_{e3}) : \mathbb{E}(U_{e4}) \approx \\ \frac{\lambda_{e0}(1-e^{-\lambda_{e1}})}{\lambda_{e1}(1-e^{-\lambda_{e0}})} : \frac{x \times res_c + z \times res_o}{x \times cap_c \cap y \times res_s} : \\ \frac{y \times cap_s + z \times cap_o}{x \times cap_c \cap y \times res_s} : 1, \quad \lambda_{e0} = \frac{2l}{v_c}\end{aligned} \tag{4.33}$$

The corresponding group expected energy cost comparison is shown as Eq. (4.34), (4.35) and (4.36).

$$\begin{aligned}\mathbb{E}(E_{e1}) - \mathbb{E}(E_{e2}) \approx & \frac{2cn}{3m \times sen_c} e_c + \\ & t_n((x \times cap_c \cap y \times res_s) \frac{1-e^{-\lambda_{e1}}}{\lambda_{e1}} \\ & - 3mk \frac{1-e^{-\lambda_{e0}}}{\lambda_{e0}}) > 0\end{aligned} \tag{4.34}$$

$$\mathbb{E}(E_{e2}) - \mathbb{E}(E_{e3}) = 0 \tag{4.35}$$

$$\begin{aligned}\mathbb{E}(E_{e2}) - \mathbb{E}(E_{e4}) \approx & t_n \frac{1-e^{-\lambda_{e0}}}{\lambda_{e0}} (3mk - \\ & (x \times cap_c \cap y \times res_s)) > 0\end{aligned} \tag{4.36}$$

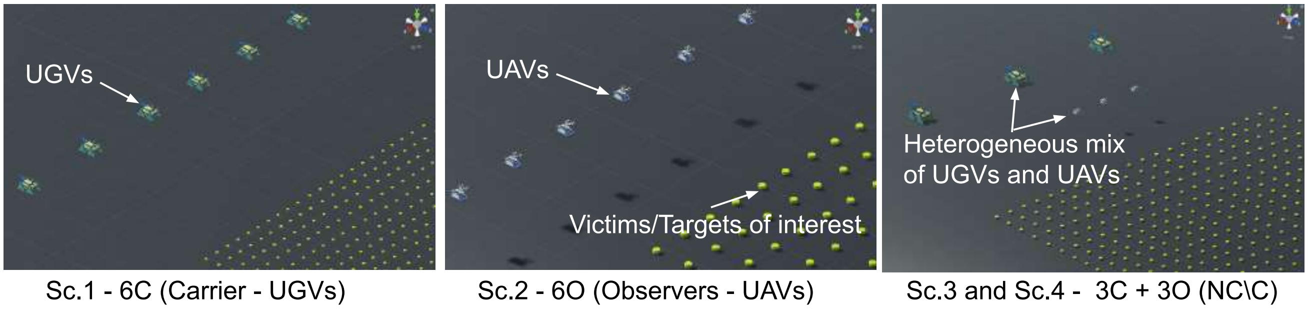
According to Eq. (4.33), (4.34), (4.35) and (4.36), we can notice that the performance of the low bound and the high bound in those groups are the combination of *(Carrier & Supplier)* and *(Carrier & Supplier & Observer)* respectively.

c. Homogeneous vs Heterogeneous

As the above discussion, at this stage, we compare the performance between low bound of heterogeneous cooperation system and homogeneous cooperation system as Eq. (4.37) and (4.38).

$$\mathbb{E}(U_{e1}) : \mathbb{E}(U_{hc}) \approx \frac{x \times cap_c \cap y \times res_s}{3m \times res_c} > 1 \tag{4.37}$$

$$\begin{aligned}\mathbb{E}(E_{e1}) - \mathbb{E}(E_{hc}) \approx & t_n \frac{1-e^{-\lambda_{e1}}}{\lambda_{e1}} ((x \times cap_c \cap y \times res_s) \\ & - 3mk)) < 0\end{aligned} \tag{4.38}$$



Sc.1 - 6C (Carrier - UGVs)

Sc.2 - 6O (Observers - UAVs)

Sc.3 and Sc.4 - 3C + 3O (NC\C)

Figure 4.3: Illustration of the four scenarios with homogeneous and heterogeneous team of *Carrier* (UGV) and *Observer* (UAV) in a rescue mission simulation

According to Eq. (4.37), we can notice that the *Expectation Utility* of heterogeneous cooperation system is larger than the homogeneous cooperation system. Also, Eq. (4.38) shows that the homogeneous cooperation system's energy cost is higher than the heterogeneous system.

4.4 Numerical evaluation

To simulate the above problem, we use "Unity" game engine and build a simple scenario (see Fig. 4.3 [34]) to verify our results. We design two kinds of experiments – *Homogeneous* and *Heterogeneous* MRS Cooperation and consider two categories of robots – *Carrier* and *Observer* implemented in the specific experiments [35].

We suppose the common category has the same battery level in the initial state, and in every moving step, carrier and observer will cost 0.045% and 0.015% energy separately. To simplify the group utility's visualization, we do not consider any obstacles, rescue resource requirement Eq. (4.8), and communication energy cost. We design four scenarios – homogeneous part simulates six carriers (Car) and six observers (Obs) fulfilling rescue mission correspondingly and considering three carriers and three observers cooperation (C) and non-cooperation (NC) for heterogeneous MRS. We also implement a simple *Negotiation-Agreement Mechanism* to avoid the collision in the whole process.

To compare the performance of *Homogeneous* and *Heterogeneous* MRS in the experiments, we calculate the amount rescuers (Group Utility) and the average energy cost per rescuing unit in five minutes Eq. (4.10). Considering observer limited energy store (basic needs) Eq. (4.7), we assume that if the indi-

[34] Scenario 3 is non-cooperative (NC) between the UGVs and UAVs and Sc. 4 is cooperative (C) between the different type of robots.

[35] Video demonstration of the experiments is available at http://hero.uga.edu/research/heterogeneous-cooperation/.

vidual energy level is below 30%, it needs to go to rest place charging 10 seconds, then back to work. Also, we assume the observer can perceive the whole map. In the homogeneous scenarios, due to working in an uncertain environment with limited perception range, the carrier's velocity is equal to a tenth of the observer's speed for avoiding uncertainty risks and satisfying its safety needs Eq. (4.6). But with the observers' assist in heterogeneous MRS cooperation, carriers can share information with observers, enlarge their perception range and double their velocity. And observers will decrease half of the speed to adapt carriers' involvement. Each carrier and observer can rescue eight and one units respectively in each round Eq. (4.9). For a non-cooperation heterogeneous system, the two groups do not interact and fulfill the mission separately.

According to the above assumption, we conduct ten simulation trials for each scenario. Fig. 4.4 shows the number of rescuers and average energy cost per rescuing unit, respectively. For the homogeneous MRS cooperation, comparing with the performance of group carrier and observer separately, although observer can achieve higher group utility (the amount rescuers) than carrier 4.4(a1) in a limited time Eq. (4.30), the average energy cost per rescuing unit represents more consumption 4.4(a2). On the other hand, for the heterogeneous MRS, the non-cooperation system shows a medial performance comparing with the different three scenarios, which does not offer distinguished advantages. Generally speaking, the heterogeneous MRS cooperation not only delivers more excellent group utility Eq. (4.37) and less system cost Eq. (4.38) from the system perspective, but also saves more cost per rescuing unit from the individual angle.

More importantly, from the statistical perspective (Fig. 4.4), comparing with the rest of the scenarios, the heterogeneous cooperation system decreases performance uncertainty (deviation between trials) and provides more stability and robustness for the whole system. It can help the system adapting more complex and uncertain environments efficiently and presents more robust viability.

4.5 Human-robot interaction

As the higher-level intelligent creature globally, humans represent more complex and diversified needs such as personal security, health, friendship, love, respect, recognition, and so forth. When we consider humans and robots work as a team, organizing their needs and getting a common ground is a precondition for human-robot collaboration in urban search and rescue missions.

From a robot needs perspective, it first needs to guarantee human security and health, such as avoiding collision with humans, protecting them from radia-

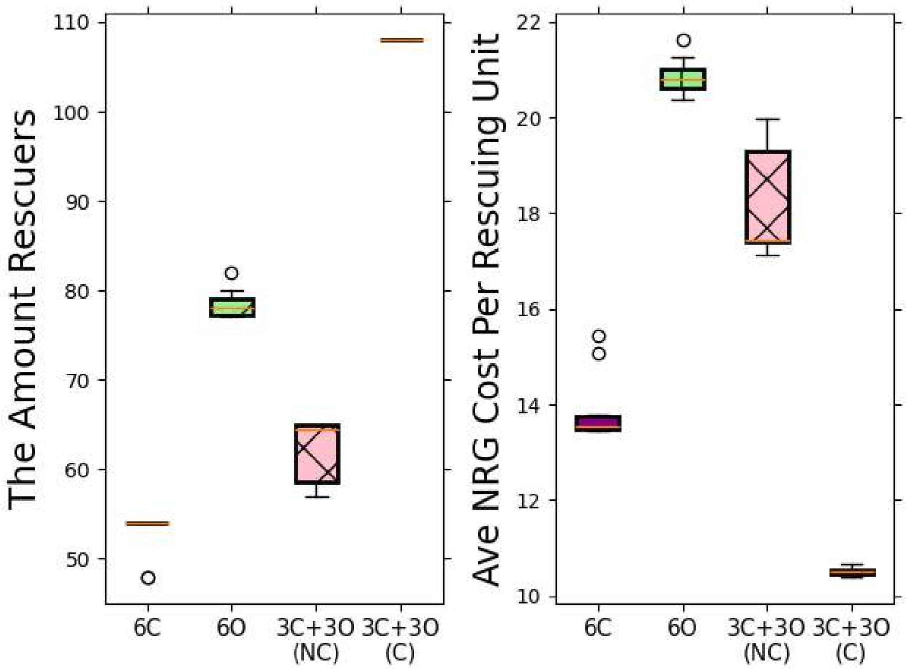


Figure 4.4: The analysis of experiments' results on homogeneous and heterogeneous MRS cooperation in simulation.

tion, and so forth. But in the higher level teaming needs, robots should consider human team members' specialty and capability to form corresponding heterogeneous *Human-Robot* team adapting specific rescue missions automatically.

Humans also expect robots to provide safety and a stable working environment in aiding rescue missions from human needs. Furthermore, efficient and reliable assistance plays an essential element for the entire rescue missions. More importantly, designing an *Interruption Mechanism* can help humans interrupt robots' current actions and re-organize them to fulfill specific emergency tasks or execute some crucial operations manually.

The individual robot learning model can be regarded as constructing models of the other agents, which takes as input some portion of the observed interaction history, and returns a prediction of some property of interest regarding the modeled agent Albrecht and Stone, 2018. In our future work, we enable robots to learn and adapt to human needs and keep up trust and rapport between humans and robots, which are critical for the task efficiency and safety improvement Nourbakhsh et al., 2005. Here, the adaptive learning of *Human-Robot Interaction* will be pursued along the following lines:

- Adopting suitable formation to perceive and survey environments predicting threats (and warn human team members) and explore new rescue tasks.

- Reasonable path planning adaptation in various scenarios avoid collision guaranteeing human security and decreasing human working environment interference.

- Combining the specific capabilities and needs of robots and humans, calculating sensible strategies to organize the entire group collaboration fulfilling corresponding rescue mission efficiently.

Using the above line of thought, assume we can model the human needs and find a way to calculate the expected utility of the human member in a human-robot team, then the proposed framework can be applied to a human-robot team by integrating the human needs with robot needs and capabilities specified in Sec. 4.3.1. It is expected to result in a harmonious teaming with heterogeneous agents (humans/robots) in achieving common goals.

4.6 Conclusion

We presented an overview of the needs-driven cooperation model for heterogeneous multi-robot systems and theoretically analyzed the importance of heterogeneity in increasing rescue mission performance. We advanced the robot needs hierarchy established in our earlier work, formalized the general rescue mission, and categorized the robots in USAR missions as carriers, suppliers, and observers.

We theoretically evaluated the system's performance in terms of the group utility and energy cost to achieve the rescue mission in a limited time. We proved that the needs-drive cooperation in a heterogeneous robot system enabled higher group utility than a homogeneous robot system. We also demonstrated the advantages of needs-driven heterogeneous cooperation through simulation experiments involving two groups of robots, namely carriers (UGVs) and observers (UAVs) in our experiment design. The results verified that heterogeneous multi-robot cooperation increased group utility and robustness and decreased energy costs and performance uncertainties compared to the homogeneous multi-robot grouping for the same task execution.

Chapter 5

Relative Needs Entropy (RNE) Trust Model

Cooperation in MAS/MRS can help agents build various formations, shapes, and patterns presenting corresponding functions and purposes adapting to different situations. Relationship between agents such as their spatial proximity and functional similarities could play a crucial role in cooperation between agents. Trust level between agents is an essential factor in evaluating their relationships' reliability and stability, much as people do. This chapter proposes a new model called *Relative Needs Entropy* (RNE) to assess trust between robotic agents. RNE measures the distance of needs distribution between individual agents or groups of agents. To exemplify its utility, we implement and demonstrate our trust model through experiments simulating a heterogeneous multi-robot grouping task in a persistent urban search and rescue mission consisting of tasks at two levels of difficulty. The results suggest that RNE trust-based grouping of robots can achieve better performance and adaptability for diverse task execution compared to the state-of-the-art energy-based or distance-based grouping models.

5.1 Introduction

Trust describes the interdependent relationship between agents Swinth, 1967, which can help us better understand the dynamics of cooperation and competition, the resolution of conflicts, and the facilitation of economic exchange Lewicki et al., 2006. In different computing fields, trust is different based on various perspectives. However, in computational agents, these models do not involve the trustor's characteristics but focus on forming trust based on past behavior Cho et al., 2015.

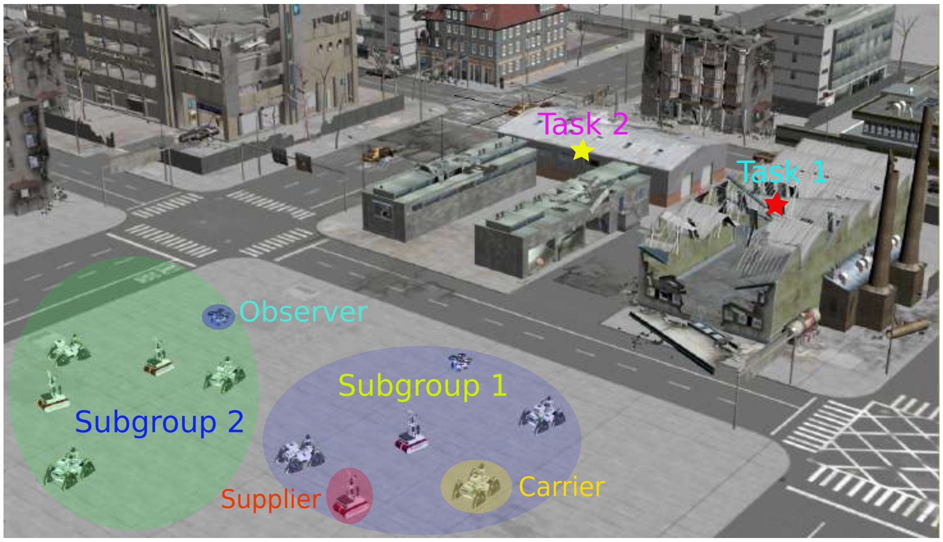


Figure 5.1: Illustration of two heterogeneous robot teams (ground robots + aerial robots) cooperatively achieving tasks of two different difficulty levels in a post-nuclear leak rescue mission.

In *automation*, authors in Lee and See, 2004 describe trust in human automation relationships as the attitude that an agent will help achieve an individual's goals in a situation characterized by uncertainty and vulnerability. The trustor here is a human and the trustee usually is the automation system or an intelligent agent like robot. Those systems' primary purpose is to assess and calibrate the trustors' trust beliefs reflecting the automation system's ability to achieve a specific task.

In *computing and networking*, the concept of trust involves many research fields, such as artificial intelligence (AI), human-machine interaction, and communication networks Cho et al., 2010; Sherchan et al., 2013; J. Wang et al., 2020. They regard an agent's trust as a subjective belief which represents the reliability of the other agents' behaviors in a particular situation with potential risks. Their decisions are based on learning from the experience to maximize its interest (or utility) and minimize risk Cho et al., 2015. Especially in the social IoT domain R. Chen et al., 2015; Mohammadi et al., 2019, trust is measured based on the information accuracy and agents' intentions (friendly or malicious) according to the current environment to avoid attacks on the system.

In *MAS*, trust by one agent on another agent reflects the trusting agents' mission satisfaction, organizational group member behaviors and consensus, and task performance at the individual level. Moreover, trust has been associated with system-level analysis, such as reward and function at the organized group structure in a specific situation, the conflict solving of heterogeneous and diverse needs, and the system evolution through learning from interaction and adaptation. On the other hand, the act of trusting by an agent can be conceptualized as consisting of two main steps: 1) *trust evaluation*, in which the agent assesses the trustworthiness of potential interaction partners; 2) *trust-aware decision-making*, in which the agent selects interaction partners based on their trust values H. Yu et al., 2013. Also, assessing the performance of agent trust models is of concern to agent trust research Fullam et al., 2005.

Considering the interactions between human agents and artificial intelligence agents like human-robot interaction (HRI), building stable and reliable relationships is of utmost importance in MRS cooperation, especially in adversarial environments and rescue missions. Fig. 5.1 illustrates two heterogeneous robot teams cooperating to achieve two different challenging tasks in a post-nuclear leak rescue mission. In such multi-agent missions, appropriate trust in robotic collaborators is one of the leading factors influencing HRI performance Khavas et al., 2020. In HRI, factors affecting trust are classified into three categories Hancock et al., 2011: 1) *Human-related factors* (i.e., including ability-based factors, characteristics); 2) *Robot-related factors* (i.e., including performance-based factors and attribute-based factors); 3) *Environmental factors* (i.e., including team collaboration, tasking, etc.).

Although modeling trust has been studied from various perspectives and involves many different factors, it is still challenging to develop a conclusive trust model that incorporates all of these factors. Therefore, future research into trust modeling needs to be more focused on developing general trust models based on measures other than the countless factors affecting trust Khavas et al., 2020. Specifically, no work from the literature considered trust in socio-intelligent systems from an artificial intelligent agent-perspective (robots) to model trust with agents having similar needs and interests to the best of our knowledge. Balch Balch, 1997 laid the foundations in introducing behavioral difference and social entropy metrics to evaluate diversity in societies of mechanically similar agents from the agent's behaviors perspective in an MRS. However, it only looks from a diversity point of view and does not establish a substantial and clear connection between the agent's behaviors and motivations, especially trust.

To bridge this critical gap, we introduce a general agent-agent trust model based on *Relative Needs Entropy (RNE)*, which is also based on the agent needs hierarchy. Using the current state of the needs of the agents, it is possible to devise a relative needs entropy value, which can help measure social trust between agents. We verify this trust model in a simulated urban search and rescue

(USAR) mission to achieve multi-robot grouping based on trust level within the robots in a group. We analyze the relative mission performance against state-of-the-art methods which use energy capability (battery level) or the spatial proximity (i.e., the inter-distance) between the robots for distributed task allocation. The specific contributions made in this research are as follows:

1. *RNE-based robot-robot and multi-robot group trust models* Considering intelligent agents representing different needs in interaction, especially for the heterogeneous MRS, we introduce a new RNE-based trust assessment model to describe trust levels between agents or groups. It presents the similarity of their needs.

2. *RNE trust based heterogeneous multi-robot grouping mechanism* We propose a hierarchical approach for RNE trust-based formation of groups in a heterogeneous MRS. This method will first form homogeneous agent subgroups, then hierarchically merges according to a specific sequence until forming the final heterogeneous groups' combination.

We provide examples of the proposed trust model and demonstrate its effectiveness in heterogeneous multi-robot grouping for a simulated urban search and rescue (USAR) mission, where the robots should form two specific groups to finish two tasks at easy and hard difficulty levels based on the environment. We hypothesize that RNE trust-based group formation would achieve better cooperation (lower costs and higher group rewards) by balancing and aligning their motivations (i.e., agent needs).

5.2 From needs to trust

5.2.1 Needs-oriented relationships

The *needs* describe the motivation of agents' interaction with the environments and other agents. Through the interaction, agents will build relationships with various categories and levels. In other words, the relationships imply the *category*, *intensity*, *similarity*, and *stability* of agents' needs, representing the *reliability* and *credibility* among agents in their interaction and cooperation. Specifically, if two agents have similar dominant needs or the distribution of the needs in the same level of the agent needs pyramid, they will have more similar motivation to cooperate and a higher probability to form as a team. It also can reduce the chances of conflicts within the group. Therefore, we can regard the *trust level* between these two agents as high and vice versa. Similarly, we can implement the ideas to analyze the trust levels between individuals and groups and between groups. Next, we will define trust in Section 5.2.2 and provide the formal definition in Section 5.3.

5.2.2 Relative needs entropy

Similar to the *information entropy*, we define the needs entropy as the difference or distance of needs distribution between agents in a specific scenario for an individual or groups. Here, needs of the robots are regarded as their motivations. From a statistical perspective, the RNE can be regarded as calculating the similarity of high-dimensional samples from the robot needs vector. **A lower RNE value means that the trust level between agents or groups is higher because their needs are well-aligned and there is low difference (distance) in their needs distributions**. Similarly, a higher RNE value will mean that the needs distributions are diverse, and there exists a low trust level between the agent or groups because of their misalignment in their motivations, which are similar to each other.

Through the above analysis, we adopt *expected utility theory* discussed in Chapter 2.2 to define the agent's fourth level needs – *Teaming Needs* (Eq. (3.4)), representing higher-level needs for an intelligent agent. It can be regarded as a kind of motivation or requirement for cooperation achieving specific goals or tasks to satisfy the individual or group's certain expected utilities. More specifically, we formalize the RNE based trust model on three different situations in Sec. 5.3.

5.3 Relative needs entropy (REN) trust model

Definition 1 (Trust between Agents)**.** *Supposing the needs' vectors of R_1 and R_2 are $N_{R_1}(n_{11}, \dots, n_{1j})$ and $N_{R_2}(n_{21}, ..., n_{2j})$, where j is the number of specific needs (categories) in the needs space. Then, through the corresponding weight vector $W(w_1, \dots, w_j)$, we get the needs' distribution of two agents are $D_{R_1}(d_{11}, \dots, d_{1j})$ and $D_{R_2}(d_{21}, ..., d_{2j})$ respectively. We can present the RNE based Trust value from R_1 to R_2 as Eq.* (5.1)*. Here, d_{1k} and d_{2k} are calculated as Eq.* (5.2)*. ($j, k \in Z^+$)*

$$\mathbb{T}(R_1||R_2) = \sum_{k=1}^{j} D_{R_{1_k}} \cdot \log \frac{D_{R_{1_k}}}{D_{R_{2_k}}} \tag{5.1}$$

$$d_{1k/2k} = n_{1k/2k} \cdot w_k / \sum_{k=1}^{j} (n_{1k/2k} \cdot w_k), \quad d_{1k/2k} \in D_{R_{1/2}} \tag{5.2}$$

Note, $\mathbb{T}(R_1||R_2) \not\equiv \mathbb{T}(R_2||R_1)$ since we consider relative entropy. That is, robot R_1's trust on the robot R_2 is not necessarily reciprocal (equal to the robot R_2's trust on R_1) because the reference "true" distributions are different in both cases. Moreover, the robot's information on other robot's needs expectations of current levels need not be accurate because of information uncertainty. This makes the RNE trust model applicable in most realistic scenarios where robots rely on perception and communication to gather information about other robots in the scenario.

Example 1 Let a robot has three needs value to represent its safety, energy, and health levels. Assume the needs' vectors of three robots R_1, R_2, and R_3 are [86, 120, 30], [20, 30, 10], and [80, 115, 25], respectively. The corresponding weight vector is [6, 4, 2][36], which represents the priority each needs level belongs to. According to Eq. (5.2), we can get their needs' distributions as D_{R_1}(0.4884, 0.4545, 0.0568), D_{R_2}(0.4615, 0.4615, 0.0769), and D_{R_3}(0.4848, 0.4646, 0.0505). Then,

$$\begin{aligned}\mathbb{T}(R_1||R_2) =& 0.4886 \cdot \log \frac{0.4886}{0.4615} + 0.4545 \cdot \log \frac{0.4545}{0.4615} + \\ & 0.0568 \cdot \log \frac{0.0568}{0.0769} = 0.00373\end{aligned} \tag{5.3}$$

$$\begin{aligned}\mathbb{T}(R_1||R_3) =& 0.4886 \cdot \log \frac{0.4886}{0.4848} + 0.4545 \cdot \log \frac{0.4545}{0.4646} + \\ & 0.0568 \cdot \log \frac{0.0568}{0.0505} = 0.0005\end{aligned} \tag{5.4}$$

By calculating the RNE values between R_1 and R_2 Eq. (5.3), and R_3 Eq. (5.4), the trust values $\mathbb{T}(R_1||R_3) < \mathbb{T}(R_1||R_2)$. Therefore, as per the definition of RNE in Sec. 5.2.2, robot R_1 has a higher trust level with R_3 compared to its trust on R_2 because robot R_1 shares similarities with the robot R_3 in terms of needs distributions and hence the RNE value is lower.

Definition 2 (Trust between Agent and Groups). *Supposing the needs' vector of the agent R is N_R, and the needs matrix of the group R_G is (N_1, ..., N_i), where i represents the number of robots in the group with corresponding weights W(w_1, ..., w_j) for the needs categories. According to Def. 1, we can represent the needs' distribution vector of R as D_R. Considering R_G needs' distribution vector as D_{R_G} Eq.* (5.5), *the RNE Trust value from the robot R to the group R_G can be represented*

[36] Due to the difference in individual needs and situations, the weight vector might be different correspondingly. Here, we consider equal weights.

as Eq. (5.6). *(i,j* $\in Z^+$*)*

$$D_{R_G} = \begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_i \end{bmatrix}_{i\times j} . \times \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_j \end{bmatrix}^T ./ \left[\sum_{k=1}^{i}(N_k \cdot W_k)\right]_{j\times 1} \tag{5.5}$$

$$\mathbb{T}(D_R||D_{R_G}) = \sum_{k=1}^{j} D_{R_k} \cdot \log \frac{D_{R_k}}{D_{R_{G_k}}} \tag{5.6}$$

Example 2 In addition to the Robot R_1 from Example 1, assume we have two robot groups R_{G_1} and R_{G_2} with two robots in each group. Their needs' matrices are $\{$[82, 114, 24]; [79, 117, 23]$\}$ and $\{$[40, 56, 12]; [56, 48, 15]$\}$ respectively. The weight vector of needs is the same as in Example 1. Through the Eq. (5.5), we get their needs' distributions as D$_{R_{G_1}}$ (0.4869, 0.4657, 0.0474) and D$_{R_{G_2}}$ (0.5507, 0.3977, 0.0516). Then we can get the RNE values between robot R_1 and group R_{G_1} Eq. (5.7), and R_{G_2} Eq. (5.8) with the help of agent-group RNE trust in Eq. (5.6) correspondingly.

$$\begin{aligned}\mathbb{T}(R_1||R_{G_1}) =& 0.4886 \cdot \log \frac{0.4886}{0.4869} + 0.4545 \cdot \log \frac{0.4545}{0.4657} + \\ & 0.0568 \cdot \log \frac{0.0568}{0.0474} = 0.000915\end{aligned} \tag{5.7}$$

$$\begin{aligned}\mathbb{T}(R_1||R_{G_2}) =& 0.4886 \cdot \log \frac{0.4886}{0.5507} + 0.4545 \cdot \log \frac{0.4545}{0.3977} + \\ & 0.0568 \cdot \log \frac{0.0568}{0.0516} = 0.007674\end{aligned} \tag{5.8}$$

Here, since $\mathbb{T}(R_1||R_{G_1}) < \mathbb{T}(R_1||R_{G_2})$ the trust of robot R_1 on the group R_{G_1} is higher than its trust on the group R_{G_2} because robot R_1 share similar needs with the average needs distribution of the group R_{G_1}.

Definition 3 (Trust between Groups). *For certain state* $s_1 \in S$*, supposing two groups* R_{G_1} *and* R_{G_2} *have the current needs matrix (*N_{11}*, ...,* N_{1i}*) and (*N_{21}*, ...,* N_{2i}*), Eq.* (5.9)*. Considering each group member's needs vector (*$n_{1k_1/2k_1}, \ldots,$ $n_{1k_j/2k_j}$*) consists of different needs elements with corresponding weights W(*w_1*, ...,* w_j*). Let* D_{N1} *(*$D_{11}, \ldots, D_{1j}$*) and* D_{N2} *(*$D_{21}, \ldots, D_{2j}$*) present the agents needs' distribution of two groups. The RNE value (Trust) from group one to two in the* *current scenario can be defined as Eq.* (5.11).

$$\begin{bmatrix} N_{11/21} \\ N_{12/22} \\ \vdots \\ N_{1i/2i} \end{bmatrix} = \begin{bmatrix} n_{11_1/21_1} & n_{11_2/21_2} & \cdots & n_{11_j/21_j} \\ n_{12_1/22_1} & n_{12_2/22_2} & \cdots & n_{12_j/22_j} \\ \vdots & \vdots & \ddots & \vdots \\ n_{1i_1/2i_1} & n_{1i_2/2i_2} & \cdots & n_{1i_j/2i_j} \end{bmatrix} \tag{5.9}$$

$$\begin{aligned} D_{N1/N2} = \begin{bmatrix} D_{11/21} \\ D_{12/22} \\ \vdots \\ D_{1i/2i} \end{bmatrix} = \begin{bmatrix} N_{11/21} \\ N_{12/22} \\ \vdots \\ N_{1i/2i} \end{bmatrix}_{i\times j} \quad .\times \\ \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_j \end{bmatrix}^T ./ \left[\textstyle\sum_{k=1}^{i}(N_{1i_k/2i_k} \cdot W_k)\right]_{j\times 1} \end{aligned} \tag{5.10}$$

$$\mathbb{T}(D_{N_1}||D_{N_2}) = \sum_{k=1}^{j} D_{N_{1_k}} \cdot \log \frac{D_{N_{1_k}}}{D_{N_{2_k}}} \tag{5.11}$$

Example 3 Let us calculate the RNE trust value from group R$_{G_1}$ to R$_{G_2}$ based on Example 2 after ceiling the needs distributions to two decimals.

$$\begin{aligned} \mathbb{T}(R_{G_1}||R_{G_2}) =&0.48 \cdot \log \frac{0.48}{0.55} + 0.46 \cdot \log \frac{0.46}{0.39} + \\ &0.04 \cdot \log \frac{0.04}{0.05} = 0.0007, \\ \mathbb{T}(R_{G_2}||R_{G_1}) =&0.55 \cdot \log \frac{0.55}{0.48} + 0.39 \cdot \log \frac{0.39}{0.46} + \\ &0.05 \cdot \log \frac{0.05}{0.04} = 0.0094. \end{aligned} \tag{5.12}$$

From this example, we can see that the robot group R_{G_1} has a significantly higher level of trust on the group R_{G_2} than the vice versa. This aligns with the intended social needs and motivations for intelligent agents (robots) where each robot views another robot from its own perspective rather than grounding on a common reference.

According to the above definitions and examples, we can see that RNE assesses the trust levels of the specific agent or team to others, which stands on the current individual or group perspective analyzing the needs' similarity from itself to others. It means that different agents or groups might have distinct RNE values or trust levels based on their current needs, knowledge of the environment and other agents, and future interests, much as humans do. In the real world, we usually assign the more challenging tasks to the agents or groups with reliable capabilities and stable performance, and vice versa for simpler tasks where grouping of agents does not need to account for trust within the group. However, most real-world tasks are challenging and has varying spatio-temporal difficulty levels. Therefore, grouping agents with similar corresponding capabilities and needs to the related mission is crucial in practical applications.

5.4 RNE trust-based multi-robot cooperation

We design a robot-aided urban search and rescue (USAR) mission to implement and illustrate the heterogeneous multi-robot grouping concept and corresponding algorithms. The mission is to retrieve (or rescue) as many resources (e.g., victims) as possible from the task area where the resources are present. In our scenarios, a set of robots will be available. Each robot is classified as one of the following: *Carrier*, *Supplier*, and *Observer*. Each type of robot has a specific role and functionality and multiple robots cooperate to fulfill this rescue mission within a limited time. When they form a group, there must be robots from each of these three categories, making the group heterogeneous in terms of functionalities.

This USAR mission is executed in multiple rounds by the heterogeneous MRS. Each round, the entire group will divide into two subgroups based on certain principles, executing the corresponding tasks. There are two kinds of adversaries (debris and radioactive resources) randomly located in our post-nuclear leak rescue mission environment. According to the task's difficulty, the distance from the initial position to the task and the number of adversaries distributed in the task area are different. See Fig. 5.2 for an illustration of the heterogeneous MRS in the USAR mission under study.

Considering the various individual agent's needs in different situations, we organize their needs based on *Robot Needs Hierarchy* as follows:

- Safety Needs: HP (Health Points), Speed, and Sensing Range (detecting adversaries);
- Basic Needs: Energy (battery);
- Capability Needs: Capacity for carrying, Resource for rescuing, and Observing Range (detecting rescuee and rescued objects);

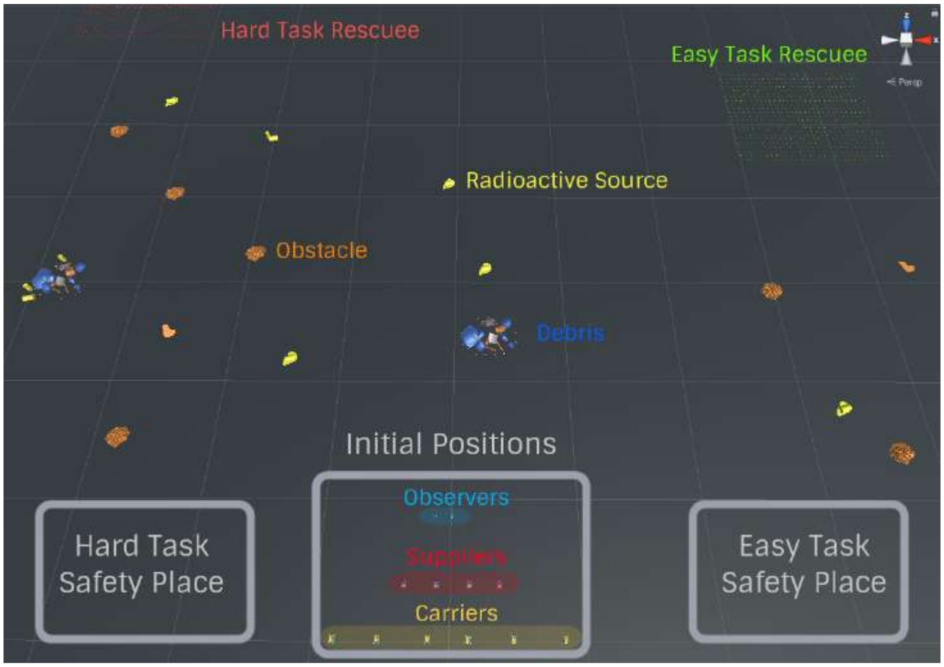


Figure 5.2: Illustration of the simulation of a post-nuclear leak rescue mission.

- Teaming Needs: The number of rescuees or rescued objects (Group Utility).

In our USAR mission, we assume that the number of *Carrier*, *Supplier* and *Observer* are x, y and z ($x, y, z \in Z^+$). According to Section 5.3 and 5.4 discussion, we define their *Needs Space*[37] as Eqs. (5.13), (5.14), and (5.15), respectively.

$$Carrier := N_c(hp_c, v_c, sen_c, eng_c, res_c, cap_c, obs_c); \tag{5.13}$$

$$Supplier := N_s(hp_s, v_s, sen_s, eng_s, res_s, cap_s, obs_s); \tag{5.14}$$

$$Observer := N_o(hp_o, v_o, sen_o, eng_o, res_o, cap_o, obs_o). \tag{5.15}$$

Here,

- hp, v, and sen represent the agent's safety needs like health points, velocity, and sensing range separately;

- eng represents the agent's basic energy needs such as battery level;

[37] The needs space has 7 variable, i.e., j=7 in Eqs. (5.1), (5.6), and (5.11).

```
Algorithm 4: RNE Trust-based Multi-robot Grouping
Input  : Carrier, Supplier, and Observer group state matrixes: $A_o$, $A_s$, and $A_o$. The number of tasks: 2.
Output : The maximum RNE difference group partition ($cso_1$, $cso_2$).
initialization;
memory data set $V_{c/s/o/cs/cso} = \{\varnothing\}$;
$m = A_{c/s/o}$.Count / 2;
for each $i \in ASC(A_{c/s/o}, m)$ do
    $V_{c/s/o}$.Add($GRC(i[0], i[1])$)
end
$G_{c/s/o}$ = Max($V_{c/s/o}$);
for each $i \in G_c$ do
    for each $j \in G_s$ do
        $V_{cs}$.Add($GRC(i, j)$)
    end
end
$G_{cs}$ = Max($V_{cs}$);
for each $i \in G_{cs}$ do
    for each $j \in G_o$ do
        $V_{cso}$.Add($GRC(i, j)$)
    end
end
return $G_{cso}$ = Max($V_{cso}$).
```

- *cap*, res, and *obs* represent the amount of the agent's carrying capacity and rescuing resources, and observing range for searching correspondingly.

When agents pass through the radioactive area, their health will be damaged continuously. For the safety needs, the individual needs should guarantee that the robot has enough HP supporting its normal function. From another perspective, through sensing perception and speed adaptation, they should avoid the collision between agents and adversaries like obstacles (debris) and radioactive sources. Similarly, they also need to satisfy corresponding energy fulfilling tasks Parasuraman et al., 2012. Otherwise, it will return to the resting place for recharging its energy or refreshing its HP.

The individual robot should represent corresponding abilities for the task to satisfy the mission's capability needs, especially for the heterogeneous agents'

Algorithm 5: All Subgroup Combinations (ASC)

```
Input   : Group members' state list $A$; The partition point: $m$.
Output : All subgroup combination list $S_A$.
initialization;
memory data set $agent$, $V$, $D$ = {$\varnothing$}
if $A.Count == m$ And $m \neq 1$ then
    return $S_A$.Add($A$)
end
else if $m == 1$ then
    for each $i \in A$ do
        $V$.Add($i$);
        $S_A$.Add($V$);
        $V = Null$.
    end
    return $S_A$;
end
for each $i \in$ ($A.Count$ - $m$) do
    $agent$ = $A[0]$;
    $A$.Remove(0);
    $D$ = $ASC(A, m-1)$;
    for each $j \in D.Count$ do
        $V$.Add($agent$);
        $V$.Add($D[j]$);
        $S_A$.Add($V$);
        $V = Null$.
    end
end
return $S_A$.
```

cooperation. Our USAR mission assumes that saving one rescued object will cost one space (capacity) and one rescuing resource. Besides, each group member's speed will decrease by 30% without an observer involving the group, and group members can share their resources in the specific task.

Supposing we have two tasks in our USAR mission, which means that the entire group needs to divide into two subgroups fulfilling the corresponding task. These two tasks have different difficulty levels - easy and hard. Compared to the easy task, the hard task has more obstacles (debris) and radioactive sources in the disaster area, which means that agents might cost more energy avoiding

Algorithm 6: Groups RNE Calculation (GRC)

```
Input    : Group 1 and 2 state matrixes $A_1$ and $A_2$.
Output : The RNE value from group 1 to 2 $RNE_{12}$.
initialization;
memory data set $D_{1/2}$ = {∅}
$D_1$ = $NDC$($A_1$.needs, $A_1$.weights);
$D_2$ = $NDC$($A_2$.needs, $A_2$.weights);
for each i ∈ $Max$($D_1$.Count, $D_2$.Count) do
    $RNE_{12} += D_1[i] \times \log(D_1[i]/D_2[i])$
end
return $RNE_{12}$
```

Algorithm 7: Needs' Distribution Calculation (NDC)

```
Input    : Group members' needs and weight matrix: $N$ and $W$.
Output : The group needs' distribution vector $D$.
initialization;
memory data set $V$, $D$ = {∅}
for each i ∈ $N$.Count do
    $V$.Add($N[i] \times W[i]$)
end
for each i ∈ $V$.Count do
    $D$.Add($V[i]$ / $V$.Sum())
end
return $D$.Sort()
```

obstacles and more HP resisting radiation emitting from radioactive resources. Besides, agents will spend more time and energy executing the challenging task because of the longer distance.

For the multi-robot grouping, we organize the group members and implement the *bottom-up* mechanism instead of partition without classification and hierarchy. More specifically, in the three homogeneous groups – *Carrier*, *Supplier* and *Observer*, we first consider separating them into two subgroups based on different combinations $C_x^{\frac{x}{2}}$, $C_y^{\frac{y}{2}}$, and $C_z^{\frac{z}{2}}$, respectively. Then, we can always find a combination with the most significant RNE value difference between those subgroups in the corresponding homogeneous group. Supposing the three groups' final combinations are $G_c(Car_1, Car_2)$, $G_s(Sup_1, Sup_2)$, and $G_o(Obs_1, Obs_2)$, using the same approach, we can merge the subgroups of car-

Algorithm 8: Deadlock Solving Mechanism (DSM)

```
Input  : Sorted collision agents' state list $C$ based on agent's needs.
Output : Solving the deadlock.
% Detect the deadlock.
while $length(C) \neq 0$ And $\Delta C.velocity == 0$ do
    $i$++;
    % Switch the execution order from the high to the low priority
      agent.
    $C_n = Swap(C, 0, C[i])$;
    % Execute the new collision list in the collision avoiding planning.
    CollisionAvoiding($C_n$);
    % Check whether the deadlock is existing.
    $DSM(C_n)$.
end
$i = 0$;
$C_n = \{\varnothing\}$;
return
```

rier and supplier into a new combination $G_{cs}(CS_1, CS_2)$ recursively. Until we get the final group partition $G_{cso}(CSO_1, CSO_2)$, we finish the entire RNE trust-based grouping. The whole process can be represented as the main RNE trust-based multi-robot grouping algorithm in Alg. 4, with supporting functions and algorithms in Alg. 5 (obtaining hierarchical subgroup combinations), 6 (calculating group's RNE trust values), Alg. 7 (calculating needs distribution vector of a robot group), and Alg. 8 (iteratively solving deadlocks in robot's task assignments).

The multi-robot grouping implementation is based on the framwork *SASS*. Here, the individual agent selecting a suitable group relies on the trust (RNE) calculation, which means that agents with similar needs levels will form a group executing the corresponding task. Besides, through the *negotiation-agreement mechanism* and route planning, agents can efficiently avoid conflicts and collisions between agent to agent and agent to static obstacles (debris). More specifically, for the *SASS collision avoidance* planning, the agent with the highest priority in the collision list will move first, and the rest of the agents stop until it out of the list. Then, they keep on executing the process recursively until the collision list is empty. Especially for this deadlock solving problem, when all the collision list agents can not move, we build a dynamic priority-based switching mechanism providing each agent has a fair opportunity to move until solving the deadlocks (see Alg. 8).

5.5 Evaluation through simulation studies

Considering the cross-platform, scalability, and efficiency of the simulations, we chose the "Unity" game engine to simulate the USAR mission. In our simulations, we consider six carriers, four suppliers, and two observers in total which separated into two equally-numbered subgroups executing two complex tasks in a post-nuclear rescue mission to rescue the resources and valuable items as much as possible within a limited time period. We compare the RNE trust-based grouping method with other methods from the literature.

5.5.1 Performance Metrics

We consider the following performance metrics in this study for a comparative analysis.

No. of Rescuees: We calculate the total number of resources the robot groups have rescued in both the tasks in the USAR mission for ten minutes per trial.

Energy Cost Per Rescuee: This is the total battery level spent by all robots divided by the total number of rescuees. From the agent basic needs perspective, we calculate the energy cost per rescuee for each category task and analyze the integrated energy cost per rescuee per trial.

HP Cost Per Rescuee: This is the total Health points used by all robots divided by the total number of rescuees. Here, we analyze the HP cost per rescuee performance similar to the *Energy Cost Per Rescuee*.

5.5.2 Simulation Studies

Until the timeout is reached, multiple rounds of rescue mission is executed. After each round, when all agents are back to the initial position, the whole group members will regroup based on agents' current grouping principle, then distributed to the corresponding task continuing the mission. From the individual perspective, if it passes through the radioactive area, its HP will cost 0.0003% per step, and we assume that each agent costs 0.0045% energy (battery point) for every moving step. Besides, to satisfy the agent's safety and basic needs, we assume that if the individual battery (energy) or HP level is below 30%, it needs to go to the pre-defined rest position for recharging energy (or for refreshing its health HP) 30 seconds, then back to the initial position waiting for the next round to start.

In the experiments, the agent's HP level affects its speed, sensing, and observing range, which means that the lower its HP, the smaller its velocity, sensing, and observing range. Besides, without observer involvement, each group member's speed will decrease by 30%. Moreover, each group member can share their resources and information between a heterogeneous subgroup to achieve the specific task cooperatively in every round. We present the needs' relationships between *Carrier*, *Supplier*, and *Observer* as follows:

- *Safety Needs*: $hp_s = hp_c = 2hp_o$; $v_c = v_s = 1.5v_o$; $sen_o = 6sen_c = 6sen_s$.
- *Basic Needs*: $eng_s = eng_c = 1.5eng_o$;
- *Capability Needs*: $cap_c = 6cap_s$, $cap_o = 0$; $res_s = 10res_c$, $res_o = 0$; $obs_o = 1000obs_c = 1000obs_s$.

Moreover, each subgroup (i.e., each task) has one observer, two suppliers, and three carriers. We conduct ten simulation trials for each principal case (method) in the whole experiment. In every case, we use the same ten different initial battery (energy) and HP levels sampled from two Gaussian distributions with means of 80% and 90%, and standard deviations of 20% and 10% respectively.

5.5.3 Compared methods from the literature

To evaluate our RNE Trust model for grouping, we implement the state-of-the-art energy-based (ENG) method Q. Yang and Parasuraman, 2020b and use the distance-based (DIS) grouping approach as the baseline in the performance comparison. Besides, we combine the agent's HP level and distance (HP$_{DIS}$) as another grouping principle applied in the experiments. We discuss the specific details for each case (method) as follows:

DIS - Distance-based: Each agent will compare the distances between its initial position and the goal points of two tasks, then selects the nearest one forming a group to execute. They continue to form a group until the necessary count of robots has reached for that group. This is a typical mode of multi-robot grouping used in the literature. For instance, in Parasuraman et al., 2018, the authors used shortest distance to the task as the measure to determine the group membership.

ENG - Energy-based: The agents with the same category will sort based only on their current energy (battery) levels at the initial positions, then divide into two parts with equal numbers. By selecting the high or low parts in each

category to form the corresponding two heterogeneous subgroups, we assign the higher energy group to the challenging task and the lower one to the easy one, respectively.

HP_{DIS} *- Health points combined with the distance based*: Similar to the energy-based grouping, here we use the health level of the robots. First, consider sorting the robots in each category (type) based on their current health points (HP) at the initial position. For the agents with the same HP level, we prioritize them by the DIS approach.

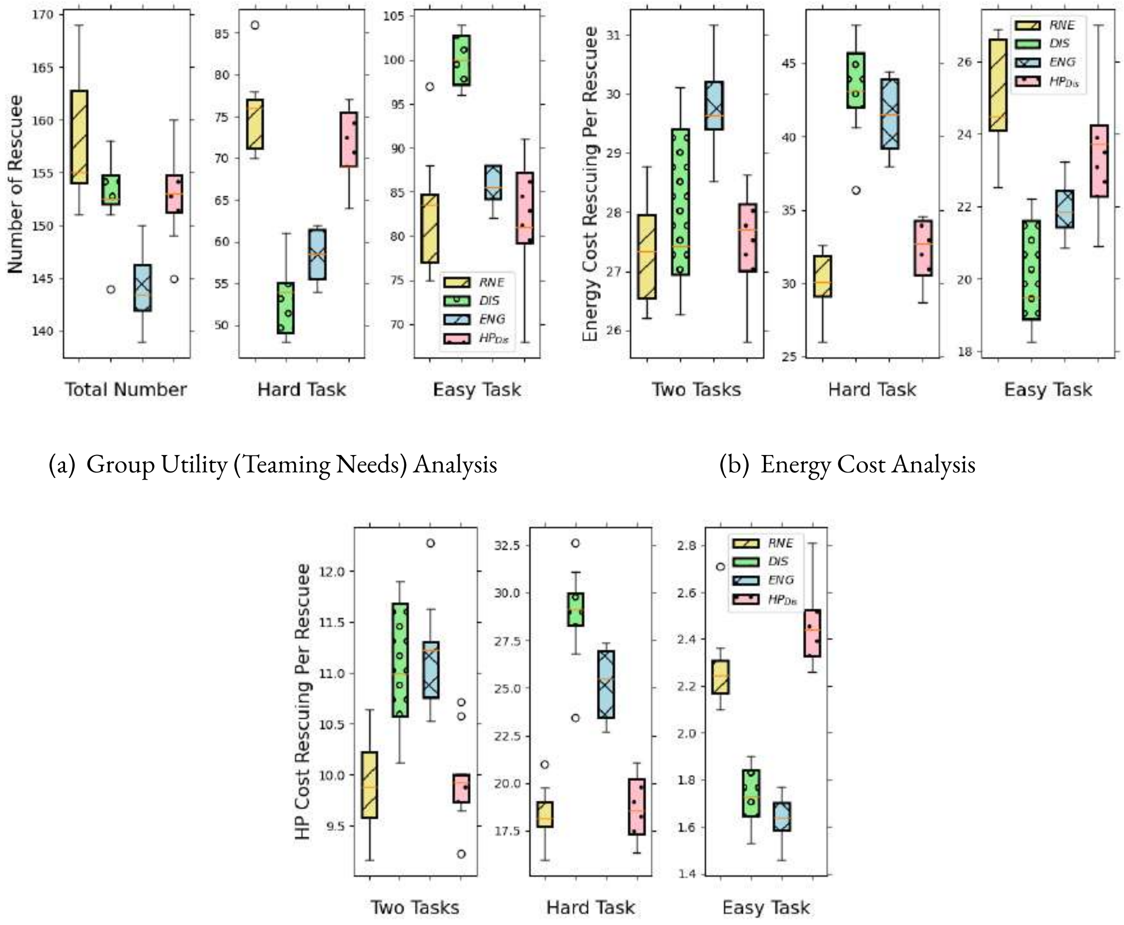
RNE Trust: By calculating the proposed RNE values between agents at the initial position, the whole group classifies as two subgroups with maximum trust level difference between each other. It means that the subgroup gathers agents with similar higher capabilities and needs levels. Then, the subgroup with higher trust level within the group executes the hard task, and the lower trust group fulfills the easy task.

Figure 5.3 show the final results of all the methods compared in this study showing all three performance metrics [38]. Through Fig. 6.7(a), we can notice that RNE has the best performance for the total number of rescuees than other methods. More specifically, RNE presents more advantages in the challenging task but inferior results in the easy task, where pure distance-based group formation is highly effective. The RNE trust-based grouping mechanism balances various needs between agents and organizes more reliable and stable groups for achieving the tough or emergent task, and uses the lower trusted agents for easy task.

From the perspective of optimizing the efficiency of system resources, Figs. 6.7(b) and 6.7(c) show that RNE essentially decreases the energy and HP cost rescuing per rescuee, especially in the challenging task. The RNE trust model helps the system reassign the resources and gathers agents with similar capabilities, needs, and interests to fulfill the suitable tasks, which improves the system performance and lets each group member's abilities be fully utilized. Like human society, the multi-robot system can make the best possible use of resources and materials based on trust-based cooperation models.

Comparing with the natural intelligent agent, when the artificial intelligence (AI) agent becomes more advanced and smart, it also represents more complex, multilayered, and diverse needs in evolution such as individual security, health, friendship, love, respect, recognition, and so forth. When we consider intelligent agents, like robots, working as a team or cooperating with human beings, organizing their needs building certain reliable and stable relationships such as trust is a precondition for robot-robot and human-robot collaboration in complex and uncertain environments.

[38] The experiment demonstration video is available at our website http://hero.uga.edu/research/trust/.



(a) Group Utility (Teaming Needs) Analysis

(b) Energy Cost Analysis

(c) HP Cost Analysis

Figure 5.3: Performance comparison of different multi-robot cooperation (grouping) models in USAR missions with two difficulty tasks

5.6 Conclusion

We introduce a general agent trust model based on *Relative Needs Entropy (RNE)* to measure and analyze the trust levels between agents and groups, representing the similarity of their diverse needs in a specific situation for heterogeneous multi-robot cooperation. Then, we illustrate how the RNE trust can be used in multi-agent decision-making applications. Specifically, we propose an RNE trust-based effective heterogeneous multi-robot cooperation method to form multiple robot groups based on trust levels within the groups. The proposed model is evaluated through extensive simulations under different difficulty tasks in a post-nuclear radiation leak-like urban search and rescue scenario. We also developed a dynamic priority switching mechanism to solve conflicts in multi-robot cooperation.

The experimental analysis showed that the RNE trust-based grouping model outperformed state-of-the-art energy-based and distance-based methods in maximizing group utilities and represented lower system costs. Trust based on relative needs distributions presents opportunities for improvements and interesting applications. Ultimately, we envision a harmonious team of robots in future multi-robot missions in which each robot values trust on each other robot.

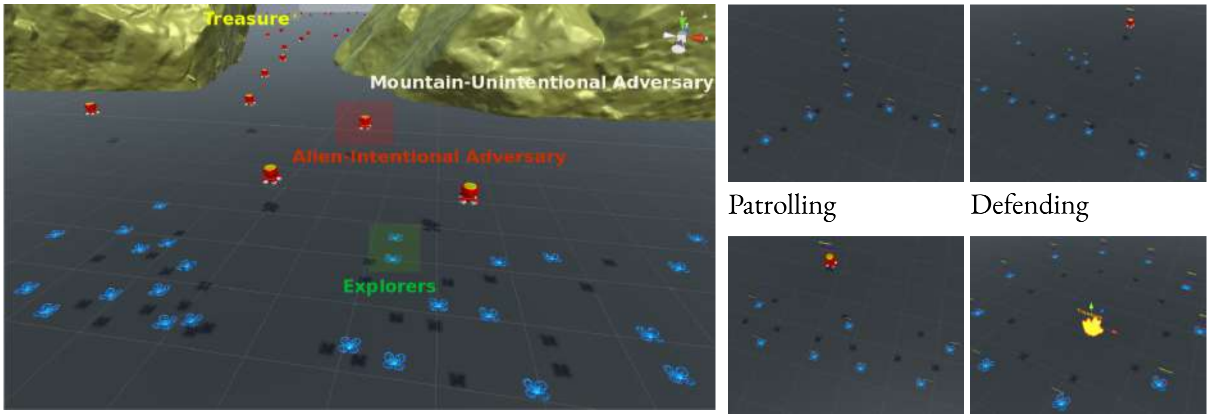
Chapter 6

Game-theoretic Utility Tree (GUT)

Underlying relationships among MAS in hazardous scenarios can be represented as Game-theoretic models. This chapter introduces a new hierarchical network-based model called *Game-theoretic Utility Tree* (GUT), which decomposes high-level strategies into executable low-level actions for cooperative MAS decisions. It combines with a new payoff measure based on agent needs for real-time strategy games. We present an Explore game domain, where we measure the performance of MAS achieving tasks from the perspective of balancing the success probability and system costs. We evaluate the GUT approach against state-of-the-art methods that greedily rely on rewards of the composite actions. Conclusive results on extensive numerical simulations indicate that GUT can organize more complex relationships among MAS cooperation, helping the group achieve challenging tasks with lower costs and higher winning rates. We also demonstrated the applicability of the GUT in the *Robotarium* (a simulator-hardware testbed for multi-robot systems) on two different domains: Explore domain and Pursuit-Evasion domain. The performances verified the effectiveness of the GUT in the real robot application and validated that the GUT could effectively organize MAS cooperation strategies, helping the group with fewer advantages achieve higher performance.

6.1 Introduction

Multiagent systems (MAS) Wooldridge, 2009 that cooperate with each other show *Distributed Intelligence*, which refers to systems of entities working together to reason, plan, solve problems, think abstractly, comprehend ideas and language, and learn Parker, 2007. Here, an individual agent is aware of other



Explore Domain Patrolling Defending Attacking Final State

Figure 6.1: An illustrative of the game scenario where the Aliens (opponent agents) block the paths to a target of the Explorers (protogonist agents)

group members, and actively shares and integrates its needs, goals, actions, plans, and strategies to achieve a common goal and benefit the entire group. They can maximize global system utility and guarantee sustainable development for each group member Parasuraman et al., 2018; Shen et al., 2004.

Systems with a wide variety of agent heterogeneity and communication abilities can be studied, and collaborative and adversarial issues also can be combined in a real-time situation Stone and Veloso, 2000. Under the presence of adversarial agents, opponents can prevent agents from achieving global and local tasks, even impair individual or system necessary capabilities or normal functions Jun and D'Andrea, 2003. Combining multiagent cooperative decision-making and robotics disciplines, researchers developed the *Adversarial Robotics,* focusing on autonomous agents operating in adversarial environments. Yehoshua and Agmon, 2015. From the agent's needs Q. Yang and Parasuraman, 2020b and motivations perspective, we can classify an *Adversary* into two general categories: *Intentional adversary* (such as an enemy agent, which consciously and actively impairs the agent's needs and capabilities as depicted in Figure 7.1 [39]) and *Unintentional adversary* (like obstacles, which passively threaten agent abilities). In this research, we focus on the multiagent tasks in the presence of adversarial agents (physical and intentional adversaries present at random locations in the environment).

MAS research domains focus on solving path planning problems for avoiding static or dynamical obstacles Agmon et al., 2011 and formation control Shapira and Agmon, 2015; Yehoshua and Agmon, 2015 from the unintentional

[39] Further, the Explorers cooperate to keep different formations in specific situations depicted on the right side.

adversary perspective. For intentional adversaries, the "pursuit domain" Chung et al., 2011 primarily deals with how to guide one or a group of pursuers to catch one or a group of moving evaders Makkapati and Tsiotras, 2019. Similarly, the robot soccer domain Nadarajah and Sundaraj, 2013 deals with how one group of robots win over another group of robots on a common game element. Foundations for normal-form team games and extensive-form adversarial team games are provided in von Stengel and Koller, 1997 and Celli and Gatti, 2018, respectively. Nevertheless, it is more realistic and practical for MAS to organize more complex relationships and behaviors, achieving given tasks with higher success probability and lower costs in adversarial environments.

The research contributes to the field of MAS/MRS through the following ways.

- Firstly, this research introduces a new hierarchical network model called *Game-theoretic Utility Tree (GUT)* to achieve cooperative decision-making of team-level multiagent strategies under the presence of adversarial agents. *GUT* consists of *Game-theoretic Utility Computation Units* at multiple levels by decomposing team strategies, thereby significantly lowering the game-theoretic operations in strategy space dimension. It combines the core principles of *Game Theory* Myerson, 2013, and *Utility Theory* Fishburn, 1970.

- Secondly, we propose a novel way of calculating the payoff (utility) values in the game-theoretic utility computation units through the agent needs expectations, which is organized hierarchically similar to the Maslow's human needs pyramid. Here, we extend our previous work on robots' needs pyramid framework proposed in Chapter 3.2 to achieve better cooperation in MAS for task allocation Q. Yang and Parasuraman, 2020b, team formation Q. Yang and Parasuraman, 2020c, and agent trust models Q. Yang and Parasuraman, 2021.

- Thirdly, we present a game of Explorers vs. Aliens (referred as *"Explore domain"*, (see Fig. 7.1) to simplify the theoretical analysis and analyze the MAS performance from the perspective of balancing the success probability of achieving tasks and system costs by organizing involved individuals' relationships and suitable groups' strategies in adversarial environments.

- Fourthly, we demonstrate the effectiveness of GUT against the standard decision-making algorithms such as QMIX Rashid et al., 2018 and greedy approaches in extensive simulations of the *Explore Domain*. The results indicate that *GUT* could organize more complex relationships among MAS, helping the group achieve tasks with low costs and high winning probability. It demonstrates that the group with fewer advantages could still achieve higher performance.

- Finally, to verify the effectiveness of the GUT in the real world (and robotics) applications, we demonstrated the *Explore Domain* in the Robotarium simulator-hardware multi-robot platform Wilson et al., 2020 comparing GUT with *random* and *greedy* approaches in various scenarios.

Furthermore, to prove the generality of the GUT, we also demonstrate its use in the *Pursuit-Evasion Domain* and compare the performance with the state-of-the-art pursuit approaches such as constant bearing (CB) and pure pursuit (PP). The *Pursuit* domain experiments are provided in the Appendix B.

6.2 Related works

Research in multiagent systems in adversarial environments can be classified into four main categories based on the tasks that the MAS will perform: Adversarial Patrol Agmon et al., 2009; Adversarial Coverage Agmon et al., 2011; Yehoshua and Agmon, 2015; Adversarial Formation Shapira and Agmon, 2015; and Adversarial Navigation Keidar and Agmon, 2018. In most adversarial MAS literature, the adversaries are not artificial agents. They might be a natural force like wind, fire, rain, or obstacles. Accordingly, the solutions to these problems are detecting opponents, path planning avoiding static or dynamical obstacles, formation control avoiding collisions, and so forth Shapira and Agmon, 2015; Yehoshua and Agmon, 2015. This is particularly applicable to urban search and rescue missions and robots deployed in disaster environments, where the agents are more concerned about unintentional adversaries such as radiation, clutters, or natural forces such as wind and fire Rizk et al., 2018.

On the other hand, there is extensive research done in studying confrontational strategies, preventive control, and behaviors to mitigate intentional adversaries (or enemy agents). For instance, Lin Lin et al., 2019 examined the problem of defending patrol tasks against a sequential attack in a knowledgeable adversarial environment. In Zheng et al., 2019, the authors studied multi-robot privacy with an adversarial approach. Sanghvi et al. Sanghvi et al., 2017 identified a swarm vulnerability and studied how an adversary takes advantage of it. Paulos et al. Paulos et al., 2019 describe an architecture for training teams of identical agents in a defense game. In Zuo and Yue, 2020, the authors proposed an attack-resilient formation control protocols to mitigate adversarial attacks on heterogeneous multigroup leader-follower formations. As we can see, most of these researches focus on privacy or cyber-adversaries or attack on the sensor or communication signals rather than tackling physical adversarial attacks confronting the agents in the spatio-temporal domain affecting the agents' health.

Cooperative decision-making among the agents is essential to address the threats posed by intentional physical adversaries or determine tradeoffs in tactical networks Cho, 2014. Current research mainly focus on solving multiplayer *pursuit and evasion* game problem Chung et al., 2011; Kolling and Carpin, 2010, which primarily deals with how to guide one or a group of pursuers to catch one or a group of moving evaders. Recent works in this domain concentrate on optimal evasion strategies and task allocation Makkapati and Tsiotras, 2019 and predictive learning from agents' behaviors Shivam et al., 2019.

From a MAS perspective, existing cooperative decision-making models use Markov decision process (MDP) and its variants L. Yu et al., 2019, game-theoretic methods, and swarm intelligence Rizk et al., 2018. They mostly involve using Reinforcement Learning (RL) and Recurrent Neural Networks (RNN) to find optimal or suboptimal action sequences based on current and previous states for achieving independent or transferred learning of decision-making policies Rashid et al., 2018; Zhang et al., 2019. Specifically, the QMIX Rashid et al., 2018 is a state-of-the-art deep multiagent RL method that uses a multi-network structure (consisting of a mixing network at the system level) for Q-learning of action policies. They demonstrate that full factorization of value decomposition networks (VDN) is not necessary to be able to extract decentralized policies that are entirely consistent with their centralized counterpart. However, these approaches mainly concern about partial cooperation, where agents only cooperate with the observable group members to minimize costs and do not consider a deeper cooperation among all the agents in the MAS to build complex group-level strategies.

To address these gaps, we propose a new hierarchical game theoretic model called *GUT* for cooperative decision-making in MAS. *GUT* consists of corresponding *Game-Theoretic Computation Units* distributed in multiple levels to combine agents' tactics in the current situation, the possibility of the previous situation, and relative environment's information. Further, to analyze our approach, we also build a more realistic testing domain, *Explorer Game*, to achieve a given task of reaching a target with minimum cost while simultaneously considering physical adversaries (opponent agents) in the environment. It is possible to extend this game domain to include multiple targets (or unbounded exploration) and combine with other adversarial game domains, such as StarCraft Vinyals et al., 2019.

Below, we discuss GUT compared with other game-theoretic decision-making frameworks. Hierarchical game theory methods mainly use *Game Tree* Myerson, 2013, *Stackelberg Game* Etro, 2013, *Games with permission structure* Gilles et al., 1992, and *Games with coalition structure* Aumann and Dreze, 1974. In *Game Trees*, each game is identical and contains all possible moves from each position. In *GUT*, each sub-game is independent, and their strategy space might be different, especially for the real-time strategy game. *Stackelberg Game* is called a hierarchical game El Oula Frihi et al., 2020 in which the leader firm moves first, and then the follower firms move sequentially. In *GUT*, each agent simultaneously calculates its strategy through the corresponding negotiation mechanism achieving group consensus, which can be organized as various structures and relationships based on different communication protocols. *GUT* has a similar design to the Games with permission structure Álvarez-Mozos et al., 2017 or coalition structure Aumann and Dreze, 1974, but it differs in that GUT is more task-oriented recursive game structure and can be implemented at individual agent level.

To demonstrate the effectiveness of GUT compared to game theoretic solutions (that do not employ a hierarchical model) to tackle the presence of adversarial agents, we compare GUT with greedy algorithms that maximizes utility across all possible strategies under the presence of adversaries, which is used in the *QMIX* (and Q-Learning) action selection component.

6.3 Needs-based game

The *utility theory* is the dominant approach to model an agent's interests or needs in a game, as we discuss in Chapter 2.2. This section will define the adversary from the agent needs perspective and introduce the *explore domain* based on the *expected utility theory*.

6.3.1 Adversarial agent definition

A friendly (ally) agent can contribute to the team, decreasing the individual needs of the team members, while an adversary can harm the team, increasing the overall needs of every team member. Based on this concept, we define an agent R_1 as adversary or friendly with respect to an agent R_2 as follows. For a certain state $\psi \in \Psi$, the agent R_1 is fulfilling a task T. Supposing the current needs of R_1 is $N_{R_1}(\psi, T)$. Considering another agent R_2 entering the R_1's observation space, the needs of R_1 can be represented as $N_{R_1}(\psi \cup R_2, T)$ under the presence of the agent R_2. The following equations define the relationship between R_1 and R_2:

$$N_{R_1}(\psi \cup R_2, T) - N_{R_1}(\psi, T) > 0; \quad (Adversary) \tag{6.1}$$

$$N_{R_1}(\psi \cup R_2, T) - N_{R_1}(\psi, T) < 0; \quad (Friendly) \tag{6.2}$$

$$N_{R_1}(\psi \cup R_2, T) - N_{R_1}(\psi, T) = 0. \quad (Neutral) \tag{6.3}$$

Definition 1 (Adversary). *If the needs of R_1 increase when R_2 is present, then R_1 regards R_2 as an Adversary (Eq.* (6.1)*).*

Definition 2 (Friendly). *If the needs of R_1 decrease when R_2 is present, then R_1 sees R_2 as a friendly agent (Eq.* (6.2)*).*

Definition 3 (Neutral). *If the needs of R_1 do not change because of R_2's presence, then R_2 is neutral to R_1 (Eq.* (6.3)*).*

Note, an obstacle is still an (unintentional) adversary as per this definition since obstacles will increase the needs of an agent in terms of using more energy to avoid collision risk.

6.3.2 Explore domain problem

To simplify theoretical analysis and numerical calculations, we define an exemplar problem domain called *Explore domain*, which is described below. In *Explore Domain*, α number of agents (called *Explorers* hereafter) are performing a task T, which is to explore an environment and collect rewards by reaching treasure locations. Supposing there are β number of (intentional) adversaries (called *Aliens* hereafter). Explorer can choose a strategy s_e from its strategy space S_e and aliens can choose a strategy s_a from its strategy space S_a. We assume both these strategy spaces are known to the Explorer agents. We also assume that the Aliens do not have a cooperation strategy, and each Alien acts independently on its own.

Let C_i represent the system costs of explorer i to perform this task and W denote the success probability (win rate) of the explorer team under the presence of alien(s). We model this as a bi-objective optimization problem (Eq. (6.4)) of finding an optimal collective team strategy for the explorers $s_e^* \in S_e$ under the premise of maximizing success W (against aliens) while minimizing costs C using the collective needs of the explorers $N_e = \sum_{i=1}^{\alpha} N_i$.

$$s_e^* = \arg\max_{s_e \in S_e} [W(s_e | T, S_a, N_e) - \sum_{i=1}^{\alpha} C_i(s_i | T, s_e, N_i)] \tag{6.4}$$

Without an adversary, the problem shrinks to a typical exploration problem (optimizing C alone) Kim, 2018, and without a task T, the problem shrinks to a typical non-cooperative zero-sum game problem (optimizing W alone) Myerson, 2013.

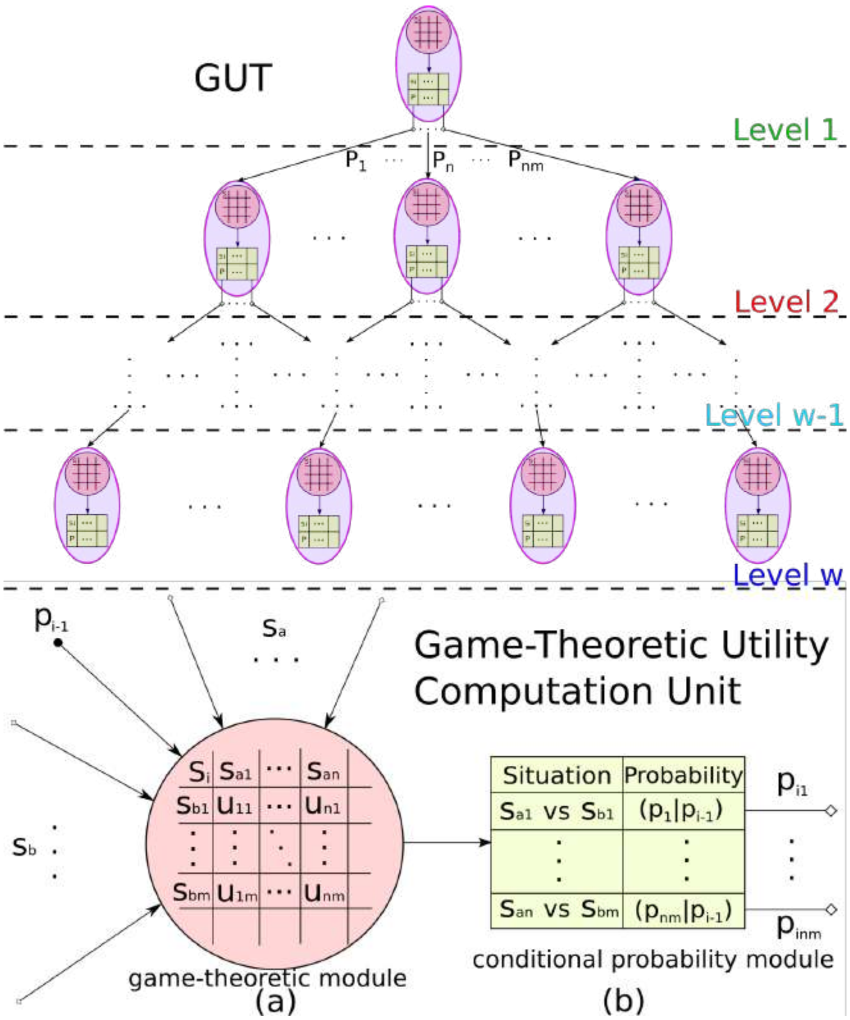


Figure 6.2: Structure of the Game-theoretic Utility Tree (GUT)

The proposed approach – GUT (Figure 6.2) – can be applied to other Real-Time Strategy (RTS) games such as air combat, StarCarft, pursuit-evasion game Chung et al., 2011, robot soccer Nadarajah and Sundaraj, 2013, and in domains involving real-time decisions such as in robotics and multi-robot systems Browning et al., 2005. These domains involve both ally agents and opponent agents (intentional adversaries), and the nature of the strategies the agents or the team can take can allow dissolving the high-level group strategies into primitive or atomic actions.

For instance, in Star Craft Vinyals et al., 2019, the strategies a player can make can be composed of the following primitives: What to do? (e.g., Build, Move, Attack, etc.); Who to perform? (which player to execute this action or which opponent to attach, etc.); Where? (physical location or point of interest); When? (immediately or delayed or how long); etc.

Similarly, in robot soccer competitions Nadarajah and Sundaraj, 2013, a player can choose a composition of low-level sub-strategies such as an action (what to do? - kicking, passing, or shooting a ball, etc.) combined with where or who (ball location or player destination, for examples). As we can see, a final strategy the team can take is a composition of these primitives. Using GUT, we can solve for atomic actions at the level of primitives and hierarchically merge them to find the best high-level strategies for the ally team against enemy agents.

6.4 Game-theoretic utility tree

Figure 6.2 outlines the structure of the *Game-theoretic Utility Tree (GUT)* and its computation units distributed in each level. First, the *game-theoretic module* (Figure 6.2 (a)) calculates the nash equilibrium based on the utility values $(u_{11}, ..., u_{nm})$ of corresponding situations, $(p_1, ..., p_{nm})$ presenting the probability of each situation. Then, through the *conditional probability (CP)* module (Figure 6.2 (b)), the CP of each situation can be described as $(p_{i1}, ..., p_{inm})$, where $p_{inm} = (p_{nm}|p_{i-1}),\ i, n, m \in Z^+$.

Here, p_{i-1} and S_i present the probability of previous situation and current strategy in the game-theoretic payoff table; s_a, s_b and n, m represent their strategy space and size on both sides, respectively. In the following description, we ground the description of the decision-making process with GUT using the "Explore domain" as an example, but the approach can be generalized to other relevant domains mentioned earlier.

To tackle intentional adversaries, agents first decompose the specific group strategy into several independent sub-strategies based on the same category of individual low-level primitives or atomic operations Q. Yang and Parasuraman, 2020b. Then, through calculating various Nash equilibrium based on different situation utility values in each level's *Game-theoretic Utility Computation Units*, agents can get optimal or suboptimal strategy sets tackling the current status according to *Nash Existence Theorem*.

From the task execution perspective, the *GUT* can be regarded as a *Task-Oriented Decision Tree*. It decomposes a big game into several sub-games with conditional dependence in each level, which organizes agents' strategies and presents more complex behaviors adapting to adversarial environments.

We formulate the decision-making problem as a zero-sum game between two groups of agents when both groups are adversaries to each other as per the Definition 1. Let A and B represent the group of ally agents and an adversary agent, respectively. The simultaneous normal-form game representing the non-cooperative game between ally and adversaries is a game structure G=$\langle\{s_A, s_B\}, \{N_A, N_B\}\rangle$. The theoretical guarantees to prove the effectiveness of the GUT solution for the game G is provided in Theorem 1 below.

Theorem 1 (GUT Decision)**.** *Supposing the GUT for the ally group has w levels of action decomposition together, making a group strategy $s_A = \{s_A^1, s_A^2, ..., s_A^w\}$, where s_A^i represents the i^{th} level sub-strategy in the ally group strategy space. Then, we can show that agent group A using GUT against adversary B will have at least one dominant strategy series $s_A^* = (s_A^1*, s_A^2*, ..., s_A^w*)$, which will enable agents A collectively execute an optimal strategy s_A^* to win against adversary B, promising a solution to the problem in Eq.* (6.4).

Proof. For *w*-level GUT, supposing game G$_i$ is the level *i* of GUT describing the corresponding zero-sum game solved using the payoff (utility) values based on the features in the needs hierarchy corresponding to the level of the GUT. To prove the theorem, we will first prove that at each level i, the game-theoretic computation units produce optimal strategies. Then, we will show that by composing strategies at each level, the GUT produces the optimal strategy for the team.

Let the size of (team) strategy space of agents in group A within the level i of GUT is l_i and that of agent B is m_i. For GUT-based decision, the *zero-sum* game at each level is

$$G_i = \langle\{s_A^i, s_B^i\}, \{N_{A_i}, N_{B_i}\}\rangle, i \in w. \tag{6.5}$$

Here, N_{A_i} is the needs/utility values of group A at a level i in the GUT. Based on the needs of agent (Eq. (3.6)), the expectation on the utilities (payoff values) at GUT level i of group A is

$$\mathbb{E}(U^i) = (u_{gk}^i)_{l_i \times m_i} = \sum_{j=1}^{|A|} N_{A_i}^j(\psi \cup B, T | s_A^i = g, s_B^i = k), \tag{6.6}$$

where, ψ is the combined state perceived by the multiagent group A (with group size $|A|$) executed through the sequence of strategies s_A, $1 \leq g \leq l_i$ and $1 \leq k \leq m_i$. Here, the payoff values depend on the agents' collective needs of the group.

According to *Nash Existence Theorem*, it guarantees the existence of a set of mixed strategies for finite, non-cooperative games of two or more players in which no player can improve his payoff by unilaterally changing strategy. So every finite game has a *Pure Strategy Nash Equilibrium* or a *Mixed Strategy Nash Equilibrium*. The process can be formalized as two steps:

a. Compute Pure Strategy Nash Equilibrium

We can present the agents' utility matrix as Eq. (6.7):

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m_i} \\ u_{21} & u_{22} & \cdots & u_{2m_i} \\ \vdots & \vdots & \ddots & \vdots \\ u_{l_i1} & u_{l_i2} & \cdots & u_{l_im_i} \end{bmatrix} \tag{6.7}$$

The row and column correspond to the utilities of agent A and B, respectively. We can compute the maximum and minimum values of the two lists separately by calculating each row's minimum value and each column's maximum value.

$$\max_{1\leq k\leq m_i} \min_{1\leq g\leq l_i} u_{gk} = \min_{1\leq g\leq l_i} \max_{1\leq k\leq m_i} u_{gk} \tag{6.8}$$

If the two value satisfy the Eq. (6.8), we can get the game G$_i$ Pure Strategy Nash Equilibrium (Eq. (6.9)), and corresponding game value in Eq. (6.10).

$$PSNE = (A_{g^*}, B_{k^*}); \tag{6.9}$$

$$V_{G_i} = u_{g^*k^*} \tag{6.10}$$

b. Compute Mixed Strategy Nash Equilibrium

The probability of choosing the strategy (at level i) agent A can be obtained as $X_A = (x_1, x_2, \ldots, x_{l_i})$, satisfying $\sum_{g=1}^{l_i} x_g = 1, x_g \geq 0,\ g = 1, 2, \ldots, l_i$. Similarly, agent B strategies' probability is $Y_B = (y_1, y_2, \ldots, y_{m_i})$, satisfying $\sum_{k=1}^{m_i} y_k = 1, y_k \geq 0,\ k = 1, 2, \ldots, m_i$. Using these probabilities, we can define *(X, Y)* as *Mixed Situation* in certain status. Then, we can deduce the expected utility of agent A and agent B as Eq. (6.11) and (6.12), respectively.

$$\mathbb{E}_A(X, Y) = \sum_{g=1}^{l_i} \sum_{k=1}^{m_i} u_{gk} x_g y_k = \mathbb{E}(X, Y); \tag{6.11}$$

$$\mathbb{E}_B(X, Y) = -\,\mathbb{E}(X, Y) \tag{6.12}$$

In the Game G$_i\langle\{S_e, S_a\}, N_{A_i}\rangle$, if we get all the *Mixed Tactics* of agent A and B as Eq. (6.13) and (6.14), we can deduce the G$_i$'s *Mixed Expansion* as Eq. (6.15). Furthermore, if a tactic (X^*, Y^*) satisfies Eq. (6.16) and (6.17), we define the tactic as the optimal strategy (Eq. (6.18)) in the current state ψ.

$$S_A^* = \{X_A\}; \tag{6.13}$$
$$S_B^* = \{Y_B\}; \tag{6.14}$$
$$G_i^* = \{S_A^*, S_B^*; \mathbb{E}\}; \tag{6.15}$$
$$\mathbb{E}(X^*, Y) \geq V_{s_A}, \forall\, Y \in S_B^*; \tag{6.16}$$
$$\mathbb{E}(X, Y^*) \leq V_{s_B}, \forall\, X \in S_A^*; \tag{6.17}$$
$$V_{s_A} = V_{G_i} = V_{s_B} \tag{6.18}$$

Therefore, we can prove that at each sublevel in the GUT, we will obtain optimal strategy as per the Nash existence theorem.

GUT decomposes a complex big game into conditionally dependent small games and presents them as a tree structure. We use the Probabilistic Graphical Models Koller and Friedman, 2009 expressing the GUT computation process. Supposing the total number of nodes (sub-games) in the GUT is $\mathcal{N}$, according to the *Chain Rule*, the *joint probability* of the GUT in group A can be described as Eq. (6.19).

$$\begin{aligned}\mathbb{P}(X) &= \mathbb{P}(X_1, X_2, ..., X_{\mathcal{N}}) \\ &= \mathbb{P}(X_1)\mathbb{P}(X_2|X_1)...\mathbb{P}(X_{\mathcal{N}}|X_1, X_2, ..., X_{\mathcal{N}-1}).\end{aligned} \tag{6.19}$$

Since *Nash Existence Theorem* guarantees that every game has at least one Nash equilibrium Jiang and Leyton-Brown, 2009, we get Eq. (6.20).

$$\mathbb{P}_i(X_i) \neq 0 \implies \mathbb{P}(X) \neq 0, i \in \mathcal{N} \tag{6.20}$$

Low Bound: If each level Nash Equilibrium calculation in the GUT is the Pure Strategy Nash Equilibrium, the individual agent can obtain a unique tactic entering into the next level, which means the agent has only one strategy trajectory (tactic combination) in the GUT for the current situation.

This shows that we can always find an optimal strategy against the adversary in the current situation through the GUT by calculating the Nash Equilibrium in the corresponding sub-games distributed in each level. □

Theorem 2 (GUT Efficiency)**.** *Assuming the agents' strategies are decomposable into sub-strategies for a specific application domain, then the GUT computes the optimum strategy of an agent (or a group's strategy) more efficiently than applying pure game-theoretic approach applied on the whole strategy space without strategy decomposition (greedy approaches).*

Proof. Using *the master theorem* Cormen et al., 2009 of calculating computational efficiency of an Algorithm with a tree-like structure, we will prove the GUT's efficiency. Suppose all games at each of the GUT's level has the same size of the strategies space, then the GUT can be described as the running time T of an approach that recursively divides a game $G(\xi)$ of size ξ into a sub-games, each of size $\xi/b,\ a, b \in Z^+$ (Eq. (6.21)).

$$T(\xi) = aT(\frac{\xi}{b}) + G(\xi), \tag{6.21}$$

If $G(\xi)$ is the one-level game (without strategy decomposition), the complexity is $O(\xi^{log\ \xi/\epsilon^2})$ Daskalakis et al., 2009, for a game with ϵ-approximate near-Nash equilibrium. Then, the game complexity $G(\xi)$ has the following asymptotic bounds:

$$\xi^{log_b a} \leq G(\xi) \leq \xi^{log\ \xi/\epsilon^2},\ \epsilon \in (0, 1). \tag{6.22}$$

Therefore, it is clear that the GUT with action decomposition is more efficient than a 1-level game of the same size. □

Furthermore, the scalability in terms of the number of ally agents depends on the efficiency of sharing information across the agents, which is impacted by the particular communication graph between the agents. For instance, if agents are connected through a complete graph, then the agents can share information within one round, whereas for any other graphs, the number of rounds can reach up to the diameter of the graph, which is limited by the K-connectedness of the graph.

6.4.1 GUT Implementation in the Explore Domain

In this domain, the explorers group in *patrol* formation (see Figure 7.1) when they explore. They are provided with the location of the treasure and always choose the shortest path to the goal, then circle around the treasure location once reached. In the whole process, explorers present a kind of global behaviors using *collective rationality* and caring about their *group interest*. In contrast, aliens are more powerful than explorers but show only *self-interest* and do not cooperate within themselves.

For this game implementation, we decompose the team strategy into three levels. At the highest strategy level, the explorer agents decide "what" to do (attack or defend) under the presence of an adversary, using their *teaming needs* as the utility function expressed through a win probability function W_{xx} for a specific Attach/Defend strategy combination. This helps make group-aware decisions to maximize the chance of collectively reaching the treasure as a team

Table 6.1: Level 1 (Attack/Defend) Tactics Payoff Matrix

Utility ET AT	Attack	Defend
Attack	W_{AA}	W_{DA}
Defend	W_{AD}	W_{DD}

Table 6.2: Level 2 (Who to Attack/Defend) Payoff Matrix

Utility ET AT	Nearest	Lowest Ability	Highest Ability
Nearest	E_{NN}	$E_{A_L N}$	$E_{A_H N}$
Lowest Ability	E_{NA_L}	$E_{A_L A_L}$	$E_{A_H A_L}$
Highest Ability	E_{NA_H}	$E_{A_L A_H}$	$E_{A_H A_H}$

while minimizing the overall team costs. See Table 6.1 [40] for the payoff matrix at Level 1. At the second strategy level (deciding "who" to attack or defend against), the explorers use their collective basic needs expressed as a function of their current energy level E_{xx} in their payoff table. This helps the decision energy-aware (Table 6.2). At the lowest strategy level (deciding "how" to attack/defend against), the explorers use their collective safety needs expressed through their health status (*HP* value) to calculate the payoff HP_{xx} at this level (Table 6.3). This ensures the decision is safety-aware since safety is the highest priority of needs as per the needs hierarchy defined in Chapter 3.2.

The design of utility functions at each level is critical to determine whether an agent can calculate reasonable tactics. See Appendix C.2 for more details on the utility functions along with their parameters for multiple levels of decomposed strategy for this Explore game domain implementation.

Alg. 9 presents the three-level GUT-based strategic decision-making in the "explore domain." Specifically, the first level defines the agent's high-level strategies: *Attack* and *Defend*, which are represented as *Triangle* and *Regular Polygon* formation shapes of the explorer team. Based on the first level decision, they need to decide the specific opponent attacking or defending in the second level. Here, we assume that agents have three basic tactics: attacking or defending

[40] **Here**, ET - Explorer Tactics, AT - Alien Tactics

Table 6.3: Level 3 (How to Attack/Defend) Payoff Matrix

Utility ET AT	One Group	Two Group	Three Group
Independent	HP_{1I}	HP_{2I}	HP_{3I}
Dependent	HP_{1D}	HP_{2D}	HP_{3D}

against the adversary, which is either the *nearest* or has the *lowest* or the *highest* attacking ability. In the last level, explorers choose how many groups to form based on whether aliens follow other adjacent alien behaviors (dependent) or not (independent).

6.5 Numerical experiments

Considering cross-platform, scalability, and efficiency of the simulations, we chose the "Unity" game engine to simulate the *Explorers and Aliens Game* and selected Gambit McKelvey et al., 2006 toolkit for calculating each level's Nash Equilibrium in GUT [41].

In the experiments, we suppose each explorer has the same initial energy and HP levels, and every moving step will cost 0.015% energy. Every communication round and per time attacking will cost 0.006% and 0.01% energy level, respectively. Attacking the aliens will cost the explorer 0.15% HP level per time. Per design, the aliens are 3x more capable than explorers in the attacks, with per time attacking energy and per time attacked HP cost for an alien are 0.03% and 0.05%, respectively. Below, we clarify the terminologies on communication and cooperation variations used in the experiments.

Partial Communication An agent only communicates and shares information with other ally agents in their observable (sensing) range. They use this partial data in this modality.

Full Communication Agents always keep in touch with each other even not in the observable range, and the communication graph may resemble a completely connected graph. It allows each agent to communicate and exchange data with its neighbor until the group reaches *Information Equilibrium*, which means that every group member has the same information.

[41] The video demonstration of the experiments showing sample trials of experiments using GUT and Greedy/QMIX is available at the link http://hero.uga.edu/research/gut/.

Algorithm 9: Explorer's Collective Strategy Using *GUT* Model in *Explorers and Aliens Game*.

```
Input: Explorers' and Aliens' states. ($\beta$ = number of Aliens)
Output: formation shape $s$; current attacking target $t$; number of
        groups $g$.
set state = "$level\ one$";
while $\Delta|\beta| >= 1$ And $|\beta|$ != 0 i.e., (at least one new alien) do
    if state=="$level\ one$" then
        Compute the Nash Equilibrium;
        Get the most feasible formation shape $s$;
        state = "$level\ two$"
    else if state=="$level\ two$" And s != Null then
        Compute the Nash Equilibrium;
        Get the most feasible attacking target $t$;
        state = "$level\ three$"
    else if state=="$level\ three$" And s, t != Null then
        Compute the Nash Equilibrium;
        Get the most feasible number of groups $g$;
if $\beta$ == 0 then
    $s$ = "$Patrol$";
    $g$ = 1;
return $s, t, g$
```

Noncooperation Agents do not cooperate (and communicate) with others. They make decisions based on their own needs and benefits only, motivated by their *self-interest*.

Partial Cooperation Based on the *partially communicated* information, agents only cooperate with the observable group members to maximize their needs and minimize costs.

Full Cooperation According to the *fully communicated* data from all the agents in a group, agents make decision based on the *group-interest* showing *Collective Rationality*.

6.5.1 Compared Scenarios and Methods

We evaluate *GUT* from two different scenarios: 1) *Interaction Experiments* compares the performance of explorers' cooperative strategies between *GUT* and *Greedy/QMIX* methods; 2) *Information Prediction* demonstrates the *GUT*

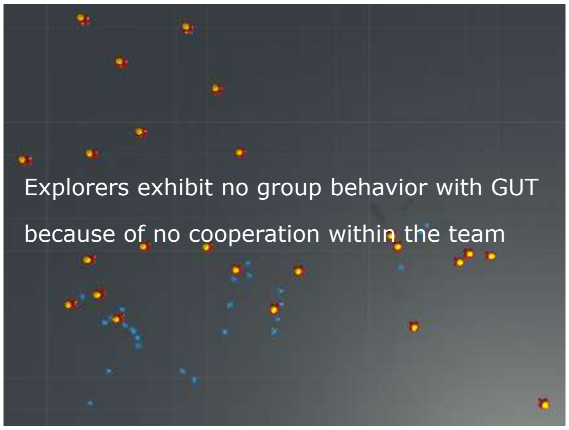

Explorers exhibit no group behavior with GUT
because of no cooperation within the team

Figure 6.3: GUT (NC)

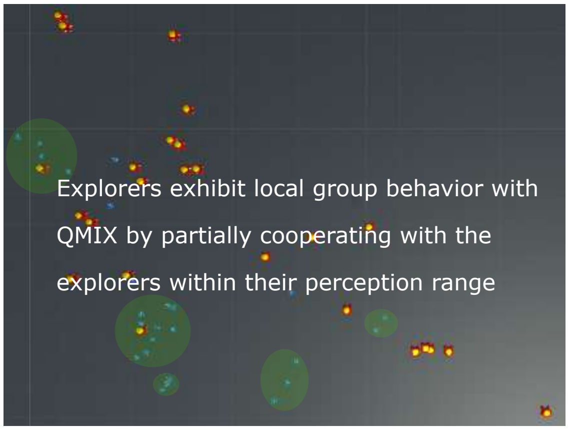

Explorers exhibit local group behavior with
QMIX by partially cooperating with the
explorers within their perception range

Figure 6.4: Greedy/QMIX (PC)

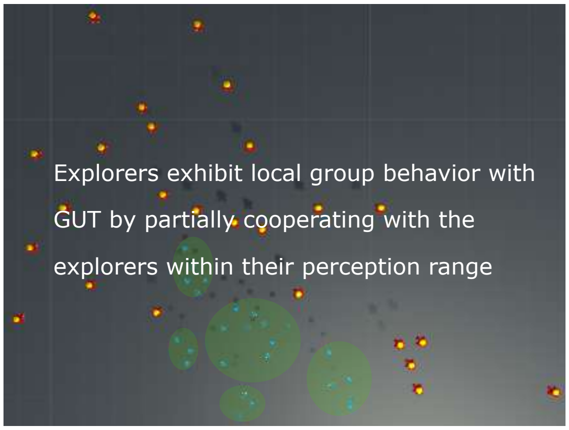

Explorers exhibit local group behavior with
GUT by partially cooperating with the
explorers within their perception range

Figure 6.5: GUT (PC)

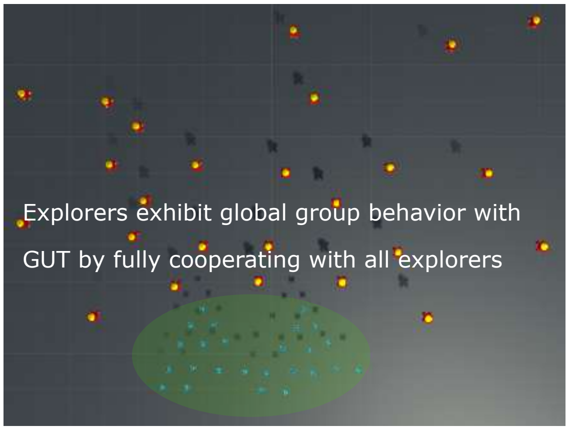

Explorers exhibit global group behavior with
GUT by fully cooperating with all explorers

Figure 6.6: GUT (FC)

when different predictive models are implemented to estimate aliens' states. Further, we analyze *GUT* by simulating different cooperation methods, comparing the performance with the state-of-the-art greedy-based approaches, including QMIX Rashid et al., 2018.

1) GUT (NC) [*Noncooperation* + *No Communication* - Self-interest] In this situation, explorers adopt *GUT* computing the winning rate based on its perceiving information, but no communication, which means that it does not get the consistency to attack or defend the specific alien (Figure 6.3).

2) Greedy/QMIX [*Partial Cooperation* + *Partial Communication*] QMIX Rashid et al., 2018 is a value-based RL method applied to MAS. Here, we only focus on the decision-making part of QMIX, which considers the global benefit yielding the same result as a set of individual rewards. It allows each agent to participate in a decentralized execution solely by choosing greedy actions for its rewards. Accordingly, we assume that each explorer can cooperate, communicate, and share information with its observing explorers. Then through calculating the corresponding win rate based on the number of observed explorers and aliens, it chooses attack or defend the specific *hp lowest* target (Figure 6.4).

3) GUT (PC) [*Partial Cooperation + Partial Communication*] Here, we consider the same situation as *QMIX*, but explorers calculate the winning rate with *GUT* and get the consistency to attack or defend the *hp lowest* alien (Figure 6.5).

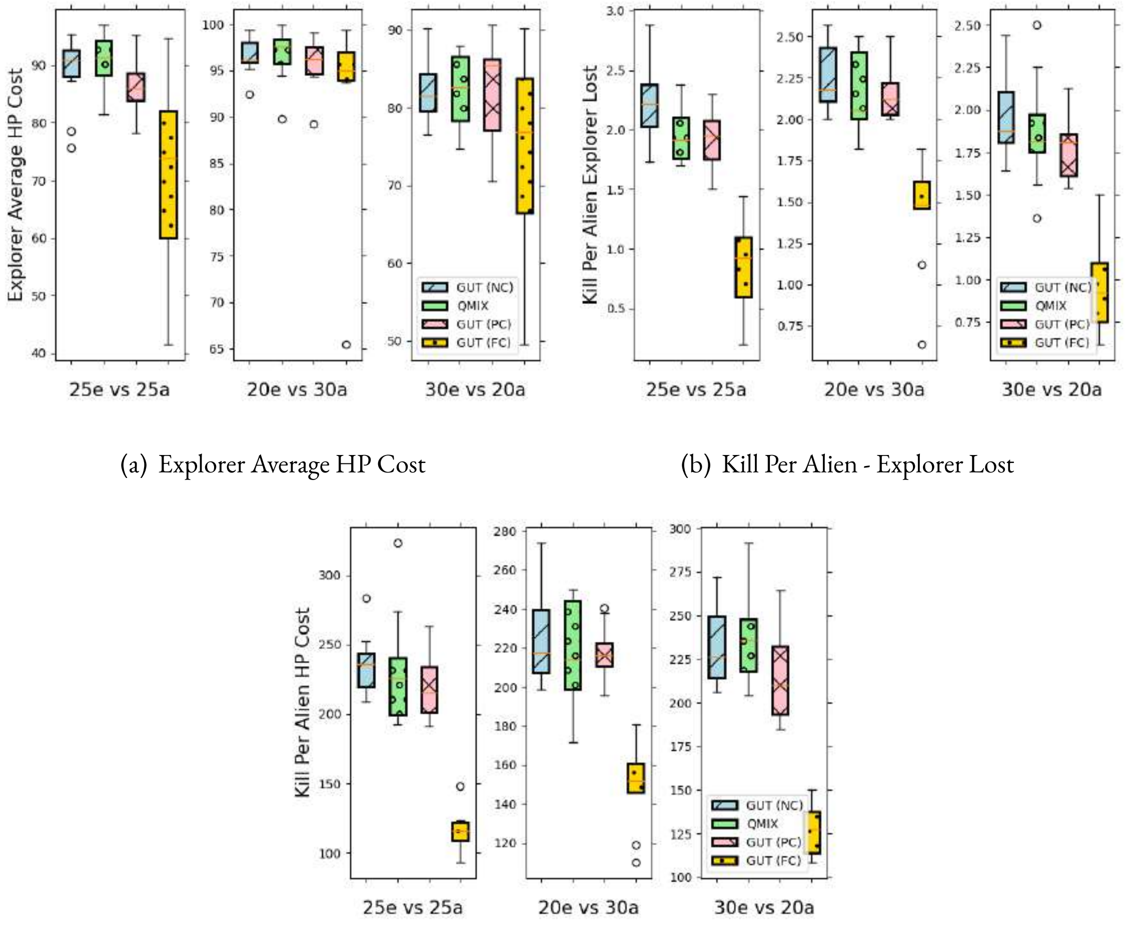
4) GUT (FC) [*Full Cooperation + Full Communication* - Collective Rationality] In this approach, we assume each explorer work and make decisions with *GUT* in full communication mode. This enables every group member to achieve consensus on the decisions in a distributed system (Figure 6.6).

6.5.2 Interaction Experiments

In these experiments, the environment is free of obstacles (*Unintentional Adversary*) We consider three different ratios (A/E) between the number of aliens and explorers as follows: *20 explorers vs 30 aliens*, *25 explorers vs 25 aliens* and *30 explorers vs 20 aliens*. We assume that an agent can detect opponents' current state in its perception range. For each scenario, we conduct ten simulation trials for each proportion with same the environment setting. In these experiments, the aliens follow a self-interest (noncooperation) random action strategy, while the explorers follow either the GUT or Greedy strategy from Section 6.5.1.

Results Figure 6.7 presents the performance results in the interaction experiments. We can see that *GUT* (FC) has the best performance compared with other cases in terms of low HP costs and loss of explorers to win against aliens. Table. 6.4 shows the winning rate, which also reflects similar results.

Specifically, the GUT (NC), QMIX, and GUT (PC) do not have much difference between in *explorer average HP cost* results (Figure 6.7(a)), but in *num. of explorers lost for killing an alien* (Figure 6.7(b)) and *HP cost for killing an alien* (Figure 6.7(c)), the QMIX and GUT (PC) show some advantage comparing with GUT (NC). The rationale for these observations is explained below. The results show that cooperation conduces to decrease the costs and boost the winning rate for more challenging tasks. More importantly, GUT can help agents representing more complex group behaviors and strategies, such as forming various shapes and separating different groups adapting adversarial environments



(a) Explorer Average HP Cost

(b) Kill Per Alien - Explorer Lost

(c) Kill Per Alien - HP Cost

Figure 6.7: The performance of explorers in interaction experiments with different proportions (e-explorers, a-aliens)

in MAS cooperation. It vastly improves system performance, adaptability, and robustness. Besides, communication plays an essential role in cooperation, such as solving conflicts and getting consistency through negotiation. In GUT (NC) and QMIX, agents only share local information about the number of observing agents for naive attacking or defending behaviors. However, GUT (FC) present more complex relationships between agents' cooperation by organizing global communication data.

To further highlight the difference in how GUT and Greedy approaches work, let us consider an example scenario where an explorer team has only two

Table 6.4: Winning Rate Results of the Interaction Experiments

Winning Rate APP PRD	GUT (NC)	QMIX	GUT (PC)	GUT (FC)
20e vs 30a	40%	50%	50%	70%
25e vs 25a	90%	100%	100%	100%
30e vs 20a	100%	100%	100%	100%

strategies, attack and defend. For the Greedy/QMIX approach, selecting attacking or defending relates to the number of the partial group and opponents and their attacking ability directly. However, for the GUT, they need to compute the *Nash Equilibrium* based on relative information then selecting a suitable strategy. It means that agents might choose a different strategy for the same condition, comparing QMIX and GUT.

For example, in a scenario with one adversary, the greedy method chooses to attack the adversary (Figure 6.8), while the GUT choose to defend against the adversary (Figure 6.9). However, in a scenario with two adversaries, QMIX (Figure 6.10) and GUT (Figure 6.11) have the same strategy reacting to the adversaries. It leads to more costs for the whole system, especially the loss of health (HP), which increases the system's instability. The two examples explain that instead of choosing greedy actions for group rewards in some scenarios, we need to consider various needs within the team and individual and balance them in a long round. It can improve the sustainability and robustness of the group, especially in uncertain environments.

6.5.3 Information Prediction

We design two kinds of perceiving models to analyze the individual and system performance. 1) In *complete information* scenario, if an agent can perceive an adversary, it can detect the adversary's status, such as the unit attacking energy cost and energy level. 2) In *incomplete information* scenario, an agent can not gain opponents' state in its observable range. So, it can obtain adversary's state through some predictions.

Assuming that adversary's unit HP cost HP_{uc} and average system cost HP_{asc} can be perceived. Then, we two such prediction models (*Linear* regressor - Eq.

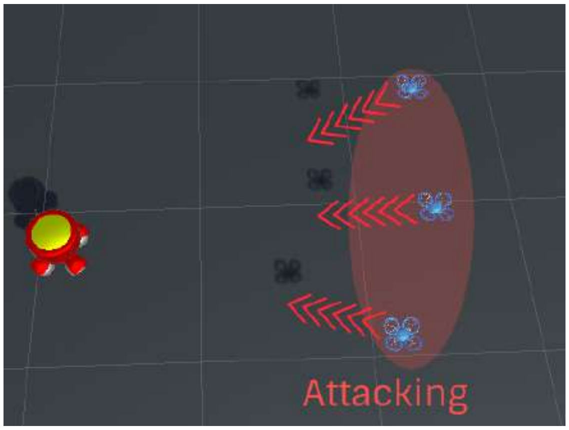


Figure 6.8: One Alien QMIX

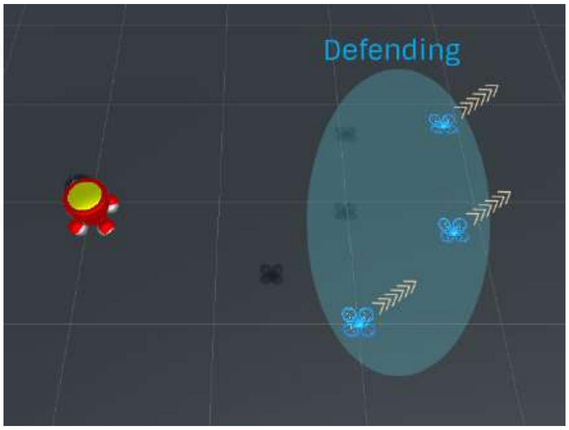


Figure 6.9: One Alien GUT

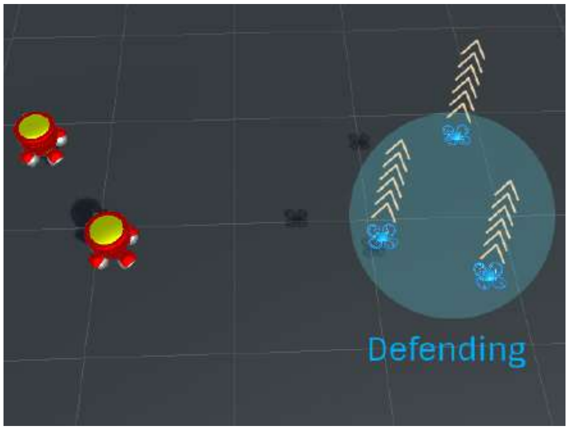


Figure 6.10: Two Aliens QMIX

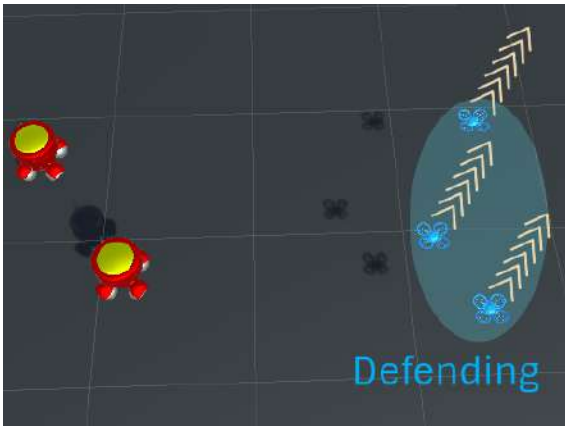


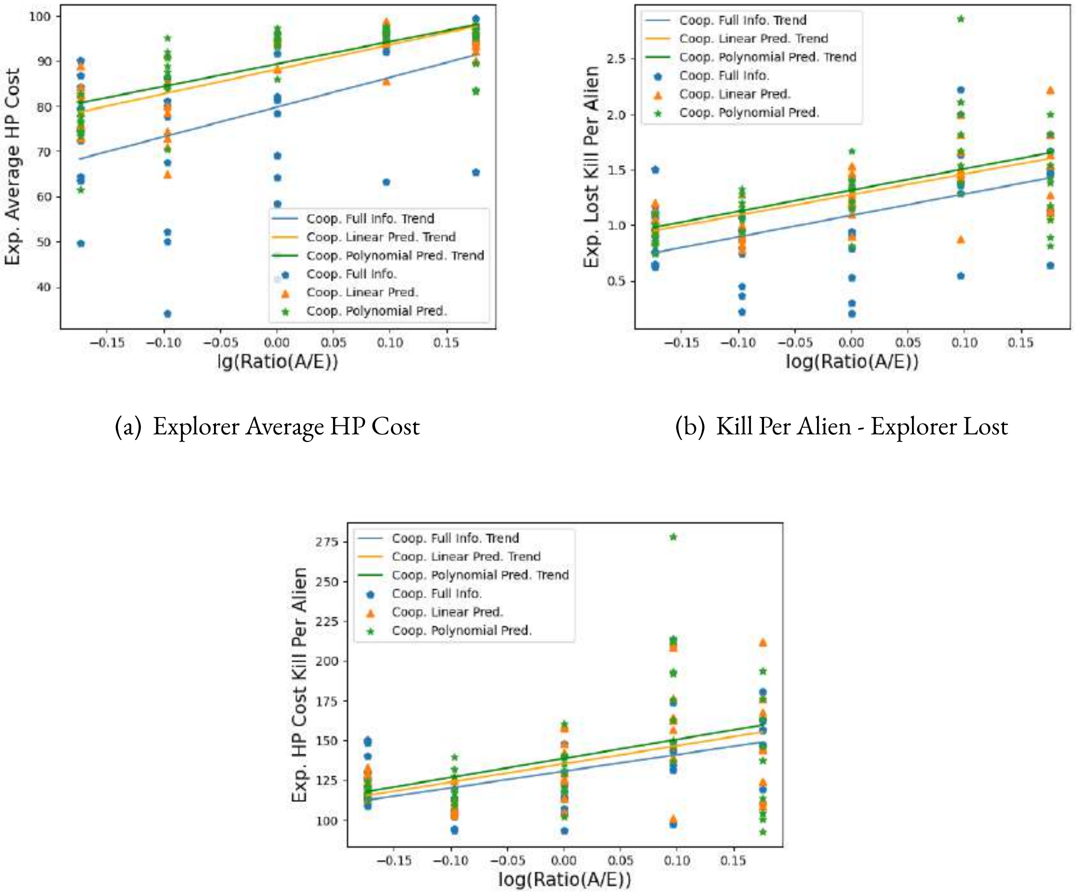
Figure 6.11: Two Aliens GUT

(6.23) and *Polynomial regressor* - Eq. (6.24)) to predict adversary's unit attacking energy cost E_{uc} and current energy level E_{el} based on the corresponding HP costs of the adversary.

$$\begin{aligned} E_{uc} &= HP_{uc} \times \beta_{uc_0} + \varepsilon; \\ E_{el} &= 100 - HP_{asc} \times \beta_{asc_0} + \varepsilon. \end{aligned} \tag{6.23}$$

$$\begin{aligned} E_{uc} &= HP_{uc}^2 \times \beta_{uc_2} + HP_{uc} \times \beta_{uc_1} + \varepsilon; \\ E_{el} &= 100 - HP_{asc}^2 \times \beta_{asc_2} - HP_{asc} \times \beta_{asc_1} + \varepsilon. \end{aligned} \tag{6.24}$$

Here, β represents corresponding regression coefficients ($\beta_{uc_{0,1,2}}$ = {0.08, 0.03, 0.0001}, $\beta_{asc_{0,1,2}}$ = {0.03, 0.0003, 0.00001}), ε represents a Gaussian noise $\mathcal{N}(0, 1)$.



(a) Explorer Average HP Cost

(b) Kill Per Alien - Explorer Lost

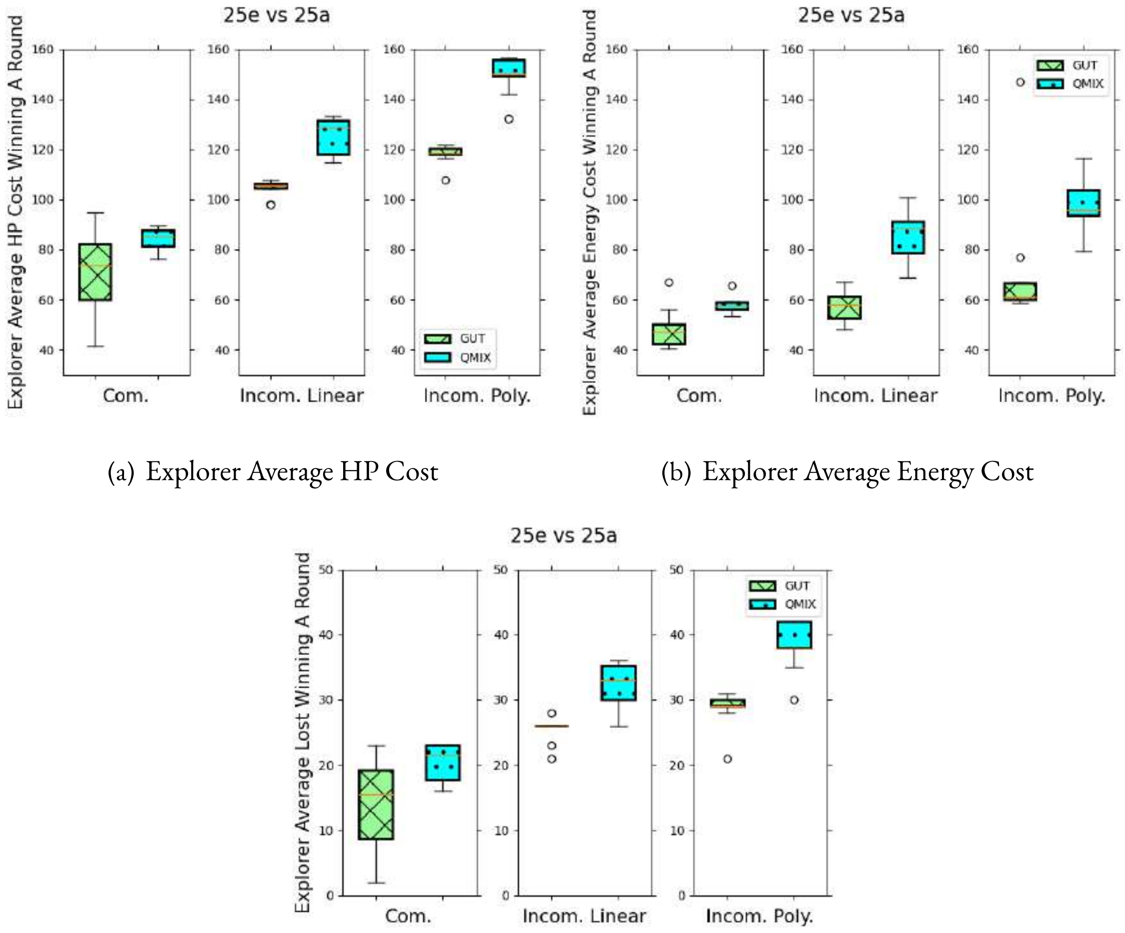
(c) Kill Per Alien - HP Cost

Figure 6.12: Explorers' performance results with different predictive models without obstacles in the environment

In the *information prediction* experiments, we investigate the performance under environments *without* and *with* obstacles.

Environment without obstacles

In this scenario, we consider five proportions of explorers and aliens distributing in the map randomly. For each ratio, we also conduct ten simulation trials with the same experimental setting.



(a) Explorer Average HP Cost

(b) Explorer Average Energy Cost

(c) Number of explorers lost per winning round

Figure 6.13: Explorers' performance results with different predictive models with obstacles in the environment

Results Figure 6.12 shows the performance results for this scenario. From an agent's perspective, the *Linear Regression* model has more accuracy than the *Polynomial Regression* model, comparing with the result trend of *Complete Information* (ground truth). From system perspective, the win rate and the mean energy/HP cost with different predictive models show the similar results. The results in Table. 6.5 support this observation.

Table 6.5: System Utility Comparison

Ra	Com			Incom L			Incom Poly		
	WR	$C_{s_e/w}$	$C_{s_{hp}/w}$	WR	$C_{s_e/w}$	$C_{s_{hp}/w}$	WR	$C_{s_e/w}$	$C_{s_{hp}/w}$
Environment without obstacles									
20:30	70%	1077.37	2649.80	30%	2367.14	6306.90	30%	2726.64	6216.44
20:25	90%	818.63	2027.98	50%	1414.84	3807.23	40%	1375.83	4824.64
25:25	100%	1211.09	1772.00	90%	1432.06	2606.12	80%	1789.15	2949.89
25:20	100%	1414.35	1739.78	100%	1449.07	1960.45	100%	1472.46	2177.52
30:20	100%	1608.18	2241.09	100%	2041.85	2370.76	100%	1961.86	2271.48
Environment with obstacles									
25:25	100%	1443.85	2110.91	70%	2144.10	3143.63	60%	2451.37	3742.98

Environment with obstacles

In this scenario, we consider a more complex environment, where there are obstacles (two mountains in the experiment setting) as well along with the adversaries. Here, the aliens adopt the QMIX approach to make their individual decision, while explorers use either QMIX or GUT. We fix the number of explorers (E=25) and aliens (A=25) and conduct ten trails for each predictive model. To implement this scenario, we designed an obstacle avoidance algorithm, which can help agents collectively avoid obstacles by adapting their edge's trajectory until it finds a suitable route to the goal point. Due to space limitations, we provide more details on this obstacle avoidance strategy in Appendix A.0.2.

Results Through the performance results shown in Figure 6.13, we notice that due to obstacles (unintentional adversaries) involved, agents' average HP and energy cost for winning a round increase distinctly. Specially, Figure 6.13(c) describes the average number of explorers sacrificed in the game to win a round as a team against adversaries. The data shows that the *Linear Regression* model presents higher accuracy than the *Polynomial* model. Table. 6.5 [42] reveals similar conclusion that obstacles lead to the decrease of the winning rate and more system costs compared to no obstacles scenario.

Limitations A suitable predictive model plays a vital role in shrinking biases between the predictive results and ground truth through those experiments.

[42] Ra: Ratio of Explorers to Aliens, WR: Winning Rate, $C_{s_e/w}$: system average energy cost winning a round, $C_{s_{hp}/w}$: system average HP cost winning a round.

More realistically, agents would face *incomplete information* scenarios to estimate opponents' states from indirect information in adversarial environments. Therefore, integrating appropriate learning-based methods to perceive adversary's state into decision-making can alleviate this limitation. Further, like any game-theoretic approach, the GUT can work only when the payoff and state of adversaries are available, which would limit its applicability. However, in such cases, agents can potentially learn those adversaries state and strategy space from interaction through computational approaches such as structural learning and reinforcement learning, which we plan to explore in our future work.

6.6 Robotarium experiments

To demonstrate the GUT on the multi-robot applications, we implement our method in the Robotarium Wilson et al., 2020 platform, a remote-accessible, free-to-use, multi-robot framework that supports controlling up to 20 robots simultaneously on a 3.2m × 2.0m large rectangular area. Each robot has the dimensions 0.11 m × 0.1 m × 0.07 m in the testbed. The platform also provides a simulator helping users test their code, which can rapidly prototype their distributed control algorithms and receive feedback about their implementation feasibility before sending them to be executed by the robots on the Robotarium.

The experiments consider three different propositions between the number of explorers and aliens in the *Explore Game* domain. They are one *explorer vs. one alien* Figure 6.14, *one explorer vs. two aliens* Figure 6.15, and *four explorers vs. three aliens* Figure 6.16. To highlight the difference between each experiment, we do not consider any obstacles in the first two scenarios Figure 6.14 and Figure 6.15. For Figure 6.16, the scenario involves obstacles (prohibited regions) and more diverse combinations between aliens guarding the corresponding areas (Encounter), building a more complex scenario.

We simulated the attack/defend process through robot proximity and the time spent in the zone. We initialize each robot in the same group with equal energy (battery) levels and health points (HP), for example, explorer with 100 points and alien with 150 points in Energy and HP correspondingly. Also, we assume that every moving step and attack damage cost 0.1% of energy and 0.3% of HP, respectively.

We compare our GUT approach with the random value selection approach (Eq. (6.26)), and the greedy algorithm **vince2002framework** with the reward

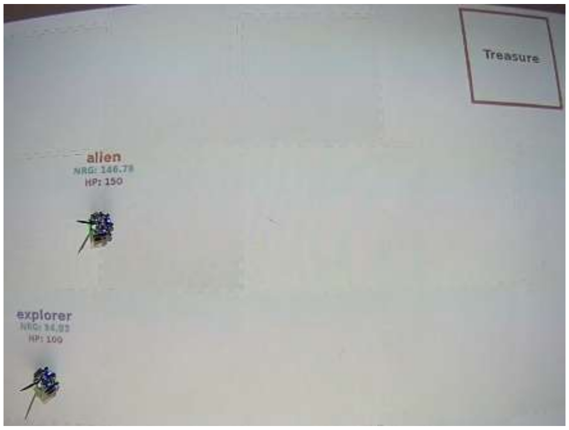


Figure 6.14: 1 explorer vs 1 alien

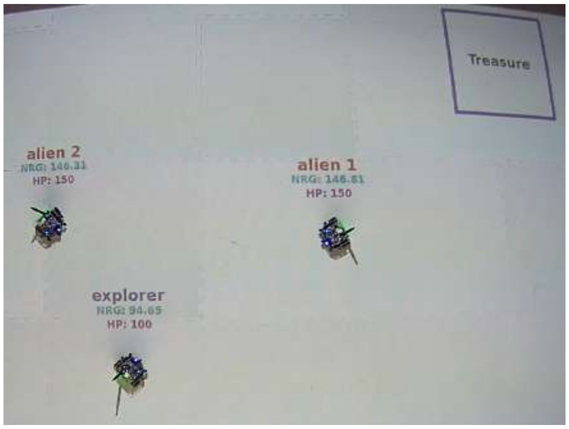


Figure 6.15: 1 explorer vs 2 aliens

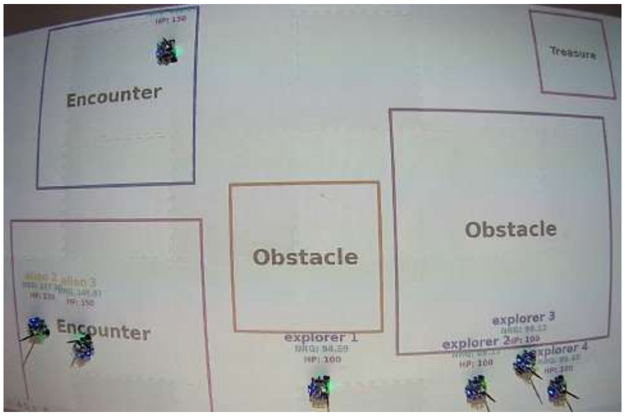


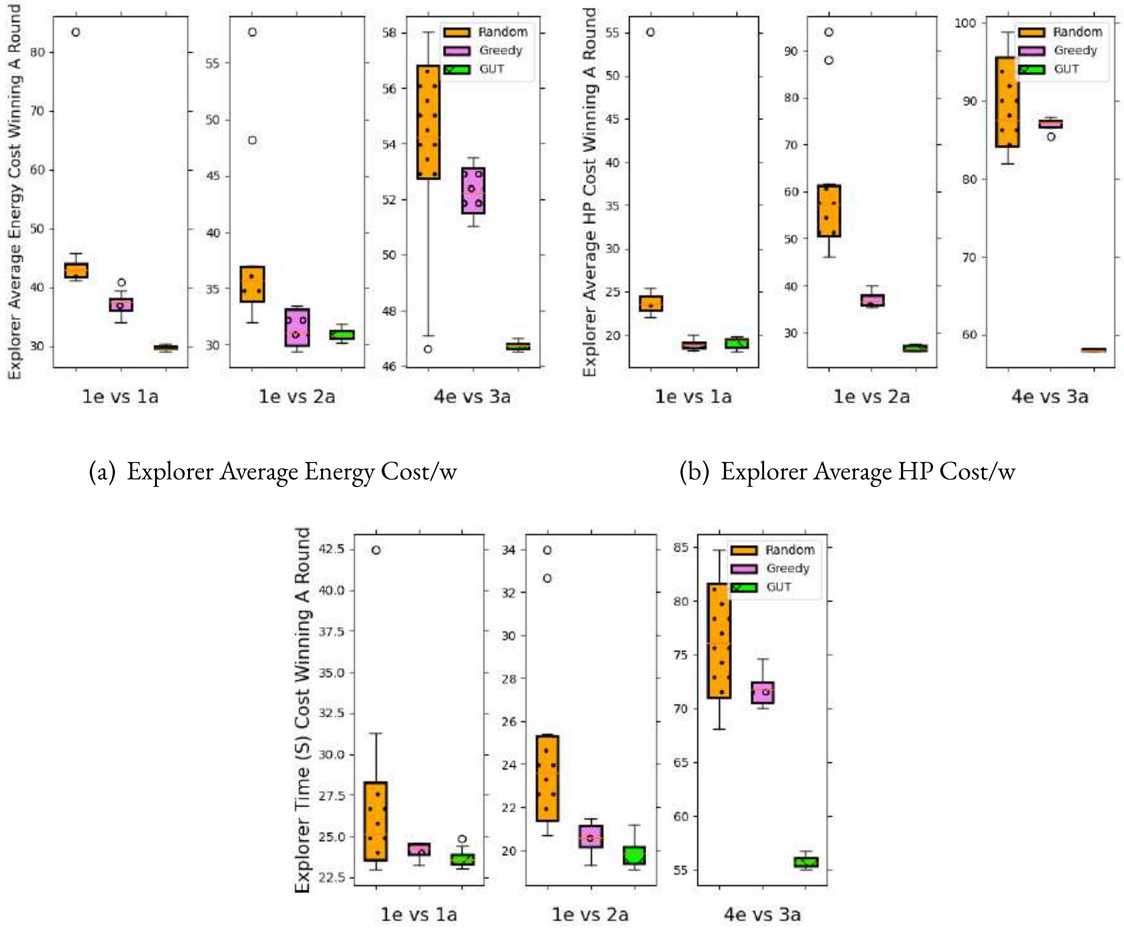
Figure 6.16: 4 explorers vs 3 aliens

(weight) function (Eq. (6.26)).

$$s_e^* = \arg\max_{s_e} \left[100 - c_1 \cdot s_e \cdot d - c_2 \cdot s_{i_2} \cdot n_a \cdot u_{hp}\right] \tag{6.25}$$

$$R_g = \arg\max_{i} \left[W_i \cdot E_i(hp)\right] \tag{6.26}$$

Here, s_{i_1} and s_{i_2} present the reward values of strategy i in the current distance and HP, which follow the normal distributions with different expectations. d is the distance between explorer and goal point. n_a and u_{hp} are the number of active aliens and unit attacking damage cost. c_1 and c_2 are the corresponding coefficients. Similarly, we concatenate the *Winning Utility Expectation* function Eq. (A.1) and *HP Utility Expectation* function Eq. (A.15) in the specific strategy combination i, building the greedy reward function Eq. (6.26).



(a) Explorer Average Energy Cost/w

(b) Explorer Average HP Cost/w

(c) Explorer Time Cost/w

Figure 6.17: The Performance of Robotarium Experiments with Different Proportion in Explore Domain

In the *Random* and *Greedy* cases, our experiments consider four different strategies through the random approach and the greedy algorithm selection in various situations. The strategy set S includes *attacking and changing direction*, *attacking and changing speed*, *defending and changing direction*, and *defending and changing speed*. According to the Eq. (6.26) and (6.26), the explorer iterate calculating the reward between the four strategies and select the strategy with the maximum value to execute in the current situation.

On the other hand, we build a two-level GUT applying in the Robotarium. For the first level tactics payoff matrix, we consider the Table. 6.1. In order to

Table 6.6: Level 2 payoff matrix for single explorer

Utility ET AT	ΔSpeed	ΔDirection
Follow	$E(hp)_{SF}$	$E(hp)_{DF}$
Retreat	$E(hp)_{SR}$	$E(hp)_{DR}$

Table 6.7: Level 2 payoff matrix for multiple explorers

Utility ET AT	Triangle	Diamond
Follow	$E(hp)_{TF}$	$E(hp)_{DF}$
Retreat	$E(hp)_{TR}$	$E(hp)_{DR}$

differentiate the strategies between single-agent and multiagent cooperation, we create two different tactics payoff matrices Table. 6.6 and 6.7 in the second level for corresponding scenarios. Specifically, Figure 6.14 and Figure 6.15 execute the tactics payoff matrices Table. 6.6, and the Figure 6.16 applies the Table. 6.7.

We implement each case with real robots in the Robotarium and conduct ten simulation trials for each scenario in Robotarium simulator. The simulation data analysis shows that in the single explorer case *1e vs 1a* and *1e vs 2a*, the costs of the energy and HP winning one round (Figure 6.17(a) and 6.17(b)) between the greedy and GUT do not have distinguished difference. To some extent, the average execution times of the *random* and *greedy* present better performance in Figure 6.17(c). However, for the *4e vs 3a*, the GUT presents prominent advantages at all levels compared with other methods, demonstrating that the GUT can help MAS organize their behaviors and select a suitable strategy adapting to various situations. At the same time, we can get a similar conclusion from the angle of the winning rate and lost agents winning per round Table. 6.8.

Moreover, we notice that comparing with the *random* approach, the *greedy* shows better performance (Figure 6.17(a), 6.17(b) and 6.17(c)) in the single cases Figure 6.14 and Figure 6.15, which also demonstrates higher *winning rate* and less *lost agents winning per round* Table. 6.8. Furthermore, considering more complex scenario Figure 6.16, even though the *greedy* has greater *winning rate* than the *random*, it costs lots of resources (energy, hp, agent, etc.) to win a round and presents fewer advantages. In future work, we will consider augmenting RL, RNN, or various learning methods on top of *GUT* to be able to fully compare with other multiagent RL methods, including *QMIX*.

Table 6.8: Winning Rate and Average Lost Agents Comparison

APP	Winning Rate			Lost Agents Per Round/w		
PRD	Random	Greedy	GUT	Random	Greedy	GUT
1e vs 1a	100%	100%	100%	-	-	-
1e vs 2a	90%	100%	100%	0.1	-	-
4e vs 3a	50%	90%	100%	2.8	2.0	-

Generally speaking, by demonstrating the *Explore* domain in the Robotarium, we further proved that the GUT could help the intelligent agent (robot) rationally analyze different situations, effectively decompose the high-level strategy into low-level tactics, and reasonably organize individuals or groups' behaviors to adapt to the current scenario. From the system perspective, through applying the GUT, the group presents more complex strategies or behaviors to solve the dynamic changing issues and optimize or sub-optimize the group utilities in MAS cooperation. From the individual angle, it reduces the agent's costs in the process and guarantees sustainable development for each group member, much like human society does.

6.7 Conclusion

We introduce a new Game-theoretic Utility Tree (GUT) for multiagent decision-making in adversarial environments. We then present an example real-time strategy game called *Explore Domain* to simplify our theoretical analysis of GUT and evaluate GUT against the state-of-the-art cooperative decision-making approach QMIX. We verified the effectiveness of GUT through two types of experiments involving interaction and information prediction between the agents. The results presented the GUT can organize more complex relationships among MAS cooperation, helping the group achieve more challenging tasks with lower costs and higher winning rates.

Also, we demonstrated the *Explore Domain* in the Robotarium and compared the GUT with the *random* and *greedy* approaches in various scenarios, which further verified the effectiveness of the GUT in the real robot application. Moreover, to prove the generality of the GUT, we implemented it in the *Pursuit Domain* against the state-of-the-art approaches – constant bearing (CB) and pure pursuit (PP). It demonstrated that the GUT could effectively organize MAS cooperation strategies, helping a group with fewer advantages still achieve higher performance.

Chapter 7

Bayesian Strategy Networks (BSN) in DRL

Adopting reasonable strategies is challenging but crucial for an intelligent agent with limited resources working in hazardous, unstructured, and dynamic changing environments to improve the system utility, decrease the overall cost, and increase mission success probability. Deep Reinforcement Learning (DRL) helps organize agent's behaviors and actions based on its state and represent complex strategies (composition of actions). This chapter proposes a novel hierarchical strategy decomposition approach based on Bayesian chaining to separate an intricate policy into several simple sub-policies and organize their relationships. We integrate this approach into the state-of-the-art DRL method, soft actor-critic (SAC), and build the corresponding Bayesian soft actor-critic (BSAC) model by organizing each sub-policies as a joint policy. We compare the proposed BSAC method with the SAC and other state-of-the-art methods such as DDPG and PPO on the standard continuous control benchmarks such as the Hopper, Walker, and the Humanoid domains in the OpenAI Gym environment. The results demonstrate promising potential of the BSAC method in terms of training efficiency.

7.1 Hierarchical needs-driven systems learning

A *strategy* describes the general plan of an AI agent achieving short-term or long-term goals under conditions of uncertainty, which involves setting sub-goals and priorities, determining action sequences to fulfill the tasks, and mobilizing resources to execute the actions Freedman, 2015. It exhibits the fundamental properties of agents' perception, reasoning, planning, decision-making, learning, problem-solving, and communication in interaction with dynamic and complex environments Langley et al., 2009. Especially in the field of real-time strategy (RTS) game Buro, 2003 and real-world implementation scenarios like robot-aided urban search and rescue (USAR) missions R. R. Murphy, 2014, agents need to dynamically change the strategies adapting to the current situations based on the environments and their expected utilities or needs Q. Yang and Parasuraman, 2020b.

Based on the model of *agent needs hierarchy* and under reasonable assumptions about needs, an agent will always want to satisfy a certain amount of dominant needs or certain distributions of needs calculated by the corresponding utility functions in a specific scenario. Then, based on various motivations caused by different levels of needs, agents present corresponding behaviors and strategies adapting to the scenarios. Through learning from those interactions with the environments, they will choose more suitable and optimal strategies or action sequences to achieve certain dominant needs or balance various needs in the specific situation. We discuss this issue from two different angles as below:

7.1.1 Single-agent systems

From a single-agent perspective, a strategy is a rule used by agents to select an action to pursue goals, which is equivalent to a policy in a Markov Decision Process (MDP) Rizk et al., 2018. More specially, in reinforcement learning (RL), the policy dictates the actions that the agent takes as a function of its state and the environment, and the goal of the agent is to learn a policy maximizing the expected cumulative rewards (needs) in the process. With advancements in deep neural network implementations, deep reinforcement learning (DRL) helps AI agents master more complex strategy (policy) and represents a step toward building autonomous systems with a higher-level understanding of the visual world Arulkumaran et al., 2017.

Specifically, in task-oriented decision-making, hierarchical reinforcement learning (HRL) enables autonomous decomposition of challenging long-horizon decision-making tasks into simpler subtasks Pateria et al., 2021. Moreover, the hierarchy of policies collectively determines the agent's behavior by solving subtasks with low-level policy learning Hengst, 2012. However, a single strategy might involve learning several policies simultaneously, which means the strategy consists of several tactics (sub-strategies) or actions executing a simple task, especially in the robot locomotion Polydoros and Nalpantidis, 2017 and RTS game Shao et al., 2019 domain. As a branching variant of the Dueling Double Deep Q-Network (Dueling DDQN) Z. Wang et al., 2016, the authors in Tavakoli et al., 2018 introduced the action branching architectures called Branching Dueling Q-Network (BDQ) through concatenating the selected sub-actions in a joint-

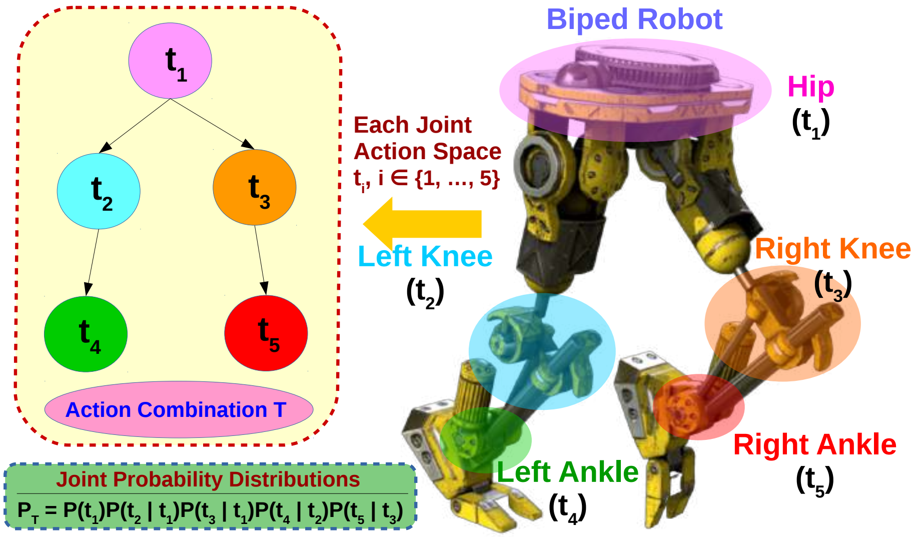


Figure 7.1: An example biped robot model's Bayesian Strategy Network, showing a decomposed strategy based on action dependencies

action tuple. Currently, the soft actor-critic (SAC) approach Haarnoja et al., 2018, an off-policy actor-critic algorithm based on the maximum entropy framework, has shown to be one of the leading algorithm for model-free off-policy DRL.

7.1.2 Multi-agent systems

As discussed in the Chapter 3.2 and 4.1, in the *SASS*, if individuals satisfy the low-level needs – *safety needs*, *basic needs*, and *capability needs*, they will pursue higher level's needs – teaming and learning. By cooperating to achieve a specific task, gaining *expected needs (rewards)*, or against the adversaries decreasing the threat, intelligent agents can benefit the entire group development or utilities and guarantee individual needs and interests. It is worth mentioning that S. Liu et al., 2021 developed the end-to-end learning implemented in the MuJoCo multi-agent soccer environment Tunyasuvunakool et al., 2020, which combines low-level imitation learning, mid-level, and high-level reinforcement learning, using transferable representations of behavior for decision-making at different levels of abstraction. Figure 7.2 represents the training process [43].

[43] Check the video link for more details: https://www.youtube.com/watch?v=KHMwq9pv7mg

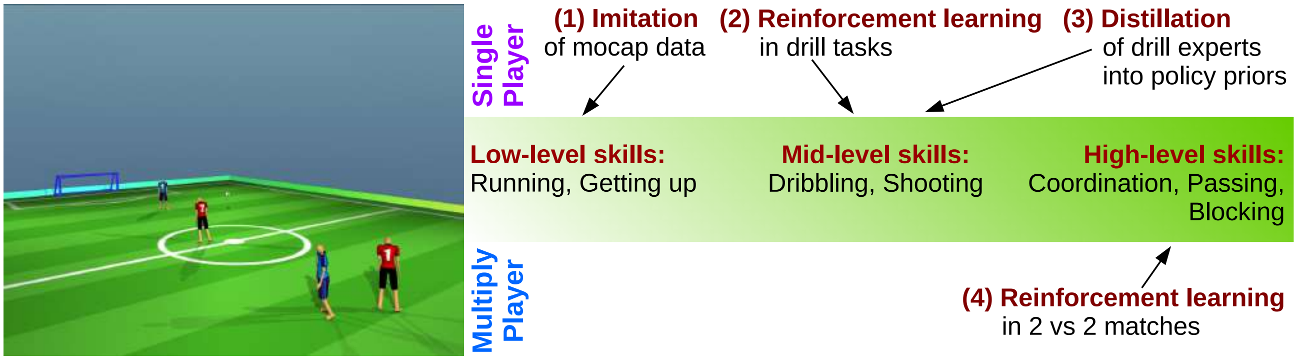


Figure 7.2: End-to-End learning of coordinated 2 vs 2 humanoid football in MuJoCo multi-agent soccer environment S. Liu et al., 2021

Although achieving some progress in those domains, DRL is still hard to explain formally how and why the randomization works, which brings the difficulty of designing efficient models expressing the relationships between various strategies (policies) Zhao et al., 2020. Especially, it is well-known that the naive distribution of the value function (or the policy representation) across several independent function approximators can lead to convergence problems Matignon et al., 2012.

This chapter first introduces the Bayesian Strategy Network (BSN) based on the Bayesian networks to decompose a complex strategy or intricate behavior into several simple tactics or actions. An example of a BSN-based strategy decomposition (or action dependencies) of a Biped Robot is shown in Fig. 7.1. Then, we propose a new DRL model termed Bayesian Soft Actor-Critic (BSAC), which integrates the Bayesian Strategy Networks (BSN) and the state-of-the-art SAC method. Through building several simple sub-policies organizing as BSN, BSAC can form a fitter joint action distribution to adapt the Q-value distribution, which increases the efficiency and performance of the training. We demonstrate the effectiveness of the BSAC against the baseline SAC method and other stat-of-the-art methods, Policy Optimization (PPO) Schulman, Wolski, et al., 2017 and Deep Deterministic Policy Gradient (DDPG) Lillicrap et al., 2015, on some standard continuous control benchmark domains in Gym/MuJoCo Brockman et al., 2016.

7.2 Related works

Reinforcement learning is a framework that helps develop self-learning capability in robots, but it is limited to the lower-dimensional problem because of complexity in memory and computation; Deep RL integrates the deep neural network implementing *function approximation* and *representation learning* to overcome the limitation of RL Singh et al., 2021. On the other hand, current research and industrial communities have sought more software-based control solutions using low-cost sensors with less operating environment requirements and calibration R. Liu et al., 2021. As the most promising algorithm, DRL ideally suits robotic manipulation and locomotion because of no predefined training data requirement. Furthermore, the control policy could be obtained by learning and updating instead of hard-coding directions to coordinate all the joints.

More specifically, compared with value-based RL, policy-based RL can avoid the policy degradation caused by the value function error and is easier to apply in the continuous action space problem Arulkumaran et al., 2017. Especially for the actor-critic algorithm can overcome policy-based methods' common drawbacks such as data efficiency, but it hardly converges in large-scale RL problems as a classical policy gradient algorithm derived from policy iteration Haarnoja et al., 2018. As a deterministic off-policy actor-critic algorithm, DDPG Lillicrap et al., 2015 can learn competitive policies through low-dimensional observations based on the same hyperparameters and network structure, but it is impractically implemented in complex environments with noise interference. On the other hand, many model-free DRL algorithms require many new samples in each gradient step, such as trust region policy optimization (TRPO) Schulman, Levine, et al., 2015, PPO Schulman, Wolski, et al., 2017, and A3C Mnih et al., 2016, which is inefficient to learn the policy and increases the complexity of the tasks. As a maximum entropy framework method, SAC Haarnoja et al., 2018 substantially improves models' performance and sample efficiency by integrating off-policy updates with a stable stochastic actor-critic formulation. However, like the conventional DRL approach, SAC still uses one actor policy network to fit the Q-value distribution. Considering solving problems in multidimensional strategy or action space, if we can optimize for each action or strategy dimension with a degree of independence and organize them appropriately, it has the potential to trigger a dramatic reduction in the number of required network outputs Tavakoli et al., 2018.

To address this issue, based on the soft actor-critic (SAC) approach, we propose a novel DRL architecture termed Bayesian Soft Actor-Critic (BSAC). By decomposing the agent's strategy (or action) into sub-actions and hierarchically organizing them as various Bayesian Strategy Networks (BSN), we can build several sub-policies (sub-actors) to form the corresponding joint policy generating the distribution of a complex strategy or action, which can better fit the Q-value distribution and increase the convergence and efficiency.

7.3 BSN organized behaviors in DRL

Building on top of the well-established suite of actor-critic methods, we introduce the incorporation of Bayesian Networks to decompose a complex actor (policy) into several simple independent sub-actors or sub-policies termed Bayesian Strategy Network (BSN). Then, we integrate the idea of the maximum entropy reinforcement learning framework in SAC, designing the method that results in our Bayesian soft actor-critic (BSAC) approach.

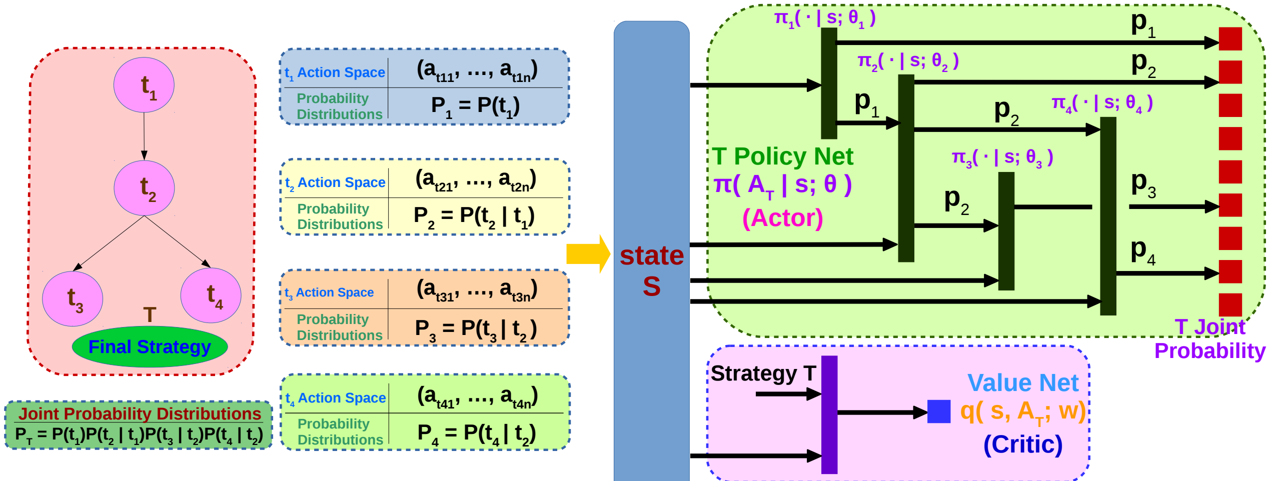
7.3.1 Bayesian Strategy Networks (BSN)

An overview of the BSN implementation in actor-critic architecture is presented in Figure 7.3. Supposing that the strategy $\mathcal{T}$ consists of m tactics ($t_1, \dots,$ t_m) and their specific relationships can be described as the *Bayesian Strategy Network* (BSN) (Figure 7.3(a)). We consider the probability distribution P_i as the policy for tactic t_i. Then, according to the Eq. (2.16), the policy $\pi(a_{\mathcal{T}} \in \mathcal{T}, s)$ can be described as the joint probability function (Eq. (7.1)) through each sub-policy (sub-actor) $\pi_i(t_i, s)$, correspondingly. The training process using the actor-critic architecture with multiple actors using the BSN (i.e., the Bayesian chain) is represented in Figure 7.3(b).

$$\begin{aligned} \pi_{\mathcal{T}}(t_1, \dots, t_m) &= \pi_1(t_1)\pi_2(t_2|t_1)\cdots\pi_m(t_m|t_1, \cdots t_{m-1}) \\ &= \pi_1(t_1)\prod_{i=2}^{m}\pi_i(t_i|t_1, \dots, t_i), \quad m \in Z^+. \end{aligned} \tag{7.1}$$

7.3.2 Derivation of Sub-Policy Iteration

Considering that the agent interacts with an environment through a sequence of observations, strategy (action combinations), and rewards, the goal of the agent is to select strategy in a fashion maximizing cumulative future reward. Accordingly, we can describe the relationships between actions in the strategy as a BSN, represented in Eq. (7.1).



(a) Bayesian Strategy Network

(b) Actor-Critic Model Architecture

Figure 7.3: An overview of the proposed BSN based implementation of the Actor-Critic DRL architecture model

More formally, we can use the corresponding deep convolution neural networks to approximate the strategy *Policy Network* (Actor) in the Eq. (7.1) as Eq. (7.2).

$$\pi(\mathcal{A}_t, s) \approx \pi(\mathcal{A}_t|s;\theta) = \prod_{i=1}^{m} \pi_i(a_{i_t}|s;\theta_i) \tag{7.2}$$

Here, $\mathcal{A}$ is the joint action or strategy space for the policy π; θ_i and a_{i_t} are the parameters and action space of each sub-policy network π_i.

On the other hand, the *Value Network* (Critic) $q(s, \mathcal{A}_t; w)$ evaluate the performance of the specific joint action $\mathcal{A}$ using the value function in Eq. (7.3) with a parameter w.

$$V(s;\theta,w) = \sum_{t\in T} \pi(\mathcal{A}_t|s;\theta) \cdot q(s,\mathcal{A}_t;w) \tag{7.3}$$

We can calculate the corresponding parameters' gradient descent using Eq. (7.4).

$$\begin{aligned}
\frac{\partial V(s;\theta,w)}{\partial\theta} &= \mathbb{E}\left[\frac{\partial \log \pi(\mathcal{A}_t|s;\theta)}{\partial\theta}\cdot q(s,\mathcal{A}_t;w)\right] \\
&= \mathbb{E}\left[\frac{\partial \log \prod_{i=1}^{m} \pi_i(a_{i_t}|s;\theta_i)}{\partial\theta}\cdot q(s,\mathcal{A}_t;w)\right] \\
&= \mathbb{E}\left[\left(\sum_{i=1}^{m}\frac{\partial \log \pi_i(a_{i_t}|s;\theta_i)}{\partial\theta}\right)\cdot q(s,\mathcal{A}_t;w)\right] \\
&= \sum_{i=1}^{m}\mathbb{E}\left[\frac{\partial \log \pi_i(a_{i_t}|s;\theta_i)}{\partial\theta}\cdot q(s,\mathcal{A}_t;w)\right]
\end{aligned} \tag{7.4}$$

Through this process, we decompose the strategy policy network $\pi_{\mathcal{T}}$ into several sub-policies networks π_i and organize them as the corresponding BSN. Furthermore, according to Eq. (7.4), each sub-policies uses the same value network to update its parameters in every iteration.

7.3.3 Bayesian Soft Actor-Critic (BSAC)

Our method incorporates the maximum entropy concept into the actor-critic deep RL algorithm. According to the additivity of the entropy, the system's entropy can present as the sum of the entropy of several independent sub-systems Wehrl, 1978. In our method, for each sub-policy evaluation step of soft policy iteration, the joint policy π will calculate the value to maximize the sum of sub-systems' π_i entropy in the BSN using the below objective function (Eq. (7.5)).

$$J_V(\pi) = \sum_{t=0}^{T}\mathbb{E}_{(s_t,\mathcal{A}_t)\sim\rho_{\pi_i}}\left[r(s_t,\mathcal{A}_t) + \frac{\alpha}{m}\sum_{i=1}^{m}\mathcal{H}(\pi_i(\cdot|s_t))\right] \tag{7.5}$$

In order to simplify the problem, we assume that the weight and the corresponding temperature parameters α_i for each action are the same in each sub-system.

The soft Q-value can be computed iteratively, starting from any function $Q : S \times A \rightarrow \mathbb{R}$ and repeatedly applying a modified Bellman backup operator $\mathcal{T}^{\pi}$ Haarnoja et al., 2018. In the Bayesian Soft Actor-Critic, the $\mathcal{T}^{\pi}$ is given by Eq. (7.6). Considering that the evaluation of each sub-policy applies the same Q-value and weight, the soft state value function can be represented in Eq. (7.7).

$$\mathcal{T}^{\pi}Q(s_t,\mathcal{A}_t) \triangleq r(s_t,\mathcal{A}_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[V(s_{t+1})\right] \tag{7.6}$$

$$V(s_t) = \mathbb{E}_{\mathcal{A}_t \sim \pi_\phi} \left[Q(s_t, \mathcal{A}_t) - \frac{1}{m} \sum_{i=1}^{m} \log \pi_{\phi_i}(a_{i_t}|s_t)) \right] \tag{7.7}$$

Specifically, in each sub-policy π_i improvement step, for each state, we update the corresponding policy according to Eq. (7.8).

$$\pi_{new} = \arg\min_{\pi' \in \Pi} D_{KL} \left(\frac{1}{m} \prod_{i=1}^{m} \pi_i'(\cdot|s_t) \middle\| \frac{exp(Q^{\pi_{old}}(s_t, \cdot))}{Z^{\pi_{old}}(s_t)} \right) \tag{7.8}$$

Here, $Z^{\pi_{old}}(s_t)$ is the partition function to normalize the distribution.

Furthermore, the soft policy evaluation and the soft policy improvement alternating execution in each soft sub-policy iteration guarantees the convergence of the optimal maximum entropy among the sub-policies combination. Here, we will use function approximators for both the Q-function and each sub-policy, optimizing the networks with stochastic gradient descent. Instead of utilizing one policy to generate the actions, we organize the agent's behaviors and actions as a BSN and implement several sub-policies to integrate them as the corresponding action or tactic combinations connected through a Bayesian chain.

Considering a parameterized state value function $V_\psi(s_t)$, soft Q-function $Q_\theta(s_t, \mathcal{A}_t)$, and several tractable sub-policies $\pi_{\phi_i}(a_{i_t}|s_t)$, the parameters of these networks are ψ, θ, and ϕ_i respectively. Then, the joint policy parameters can be updated by minimizing the expected KL-divergence in Eq. (7.9).

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[D_{KL} \left(\frac{1}{m} \prod_{i=1}^{m} \pi_{\phi_i}(\cdot|s_t) \middle\| \frac{exp(Q_\theta(s_t, \cdot)}{Z_\theta(s_t)} \right) \right] \tag{7.9}$$

Especially for each sub-policy network, we implement a Gaussian distribution with mean and variance generated by neural networks building the action distribution to sample the corresponding sub-action. Through integrating every sub-policy, we can form the policy combination fitting the specific joint action or strategy distribution. Furthermore, the joint actions can be generated by sampling from the different sub-distributions, which is domain-specific.

As discussed above, we integrate the BSN into the SAC Haarnoja et al., 2018 algorithm and extend the SAC approach to the proposed BSAC policy *evaluation*, *improvement*, and *iteration*, similar to the foundations in SAC as follows:

Lemma 1 (Bayesian Soft Policy Evaluation). *Consider the soft Bellman backup operator $\mathcal{T}^\pi$ in the Eq.* (7.6) *and define $Q^{k+1} = \mathcal{T}^\pi Q^k$. The sequence Q^k will* *converge to the soft Q-value of the joint policy* π *for each sub-policy* π_i *as* $k \to \infty$, $i \in m$.

Proof. see Appendix. C.1-Lemma 1 □

Lemma 2 (Bayesian Soft Policy Improvement). *Let the joint policy* π_{new} *optimize the minimization problem defined in Eq.* (7.8) *and* $\pi_{old} \in \Pi$, *then* $Q^{\pi_{new}}(s_t, \mathcal{A}_t) \geq Q^{\pi_{old}}(s_t, \mathcal{A}_t)$ *for all* $(s_t, \mathcal{A}_t) \in \mathcal{S} \times \mathcal{A}$ *with* $|\mathcal{A}| < \infty$.

Proof. see Appendix. C.1-Lemma 2 □

Theorem 1 (Bayesian Soft Policy Iteration). *Iterating the Bayesian soft policy evaluation and Bayesian soft policy improvement from any joint policy* $\pi \in \Pi$ *converges to a joint policy* π^* *for* $Q^{\pi^*}(s_t, \mathcal{A}_t) \geq Q^{\pi}(s_t, \mathcal{A}_t)$ *and* $(s_t, \mathcal{A}_t) \in \mathcal{S} \times \mathcal{A}$, *assuming* $|\mathcal{A}| < \infty$.

Proof. see Appendix. C.1-Theorem 1 □

7.3.4 Loss Function

There are numerous ways to calculate the distributed temporal-difference (TD) errors across the sub-actions aggregating to a specific loss, and a simple approach defines the loss to be the expected value of a function of the averaged TD errors across the sub-actions Tavakoli et al., 2018. More specifically, the soft value function is trained to minimize the squared residual error of each sub-policy in BSN, and we define the loss to be the expected value of the mean squared TD error across the sub-policies as expressed in Eq. (7.10).

$$\begin{aligned} J_V(\psi) = &\frac{1}{m}\sum_{i=1}^{m} \mathbb{E}_{s_t \sim \mathcal{D}_i}[\frac{1}{2}(V_\psi(s_t) - \\ &\mathbb{E}_{a_{i_t} \sim \pi_{\phi_i}}[Q_\theta(s_t, \mathcal{A}_t) - \log \pi_{\phi_i}(a_{i_t}|s_t)])^2] \end{aligned} \tag{7.10}$$

Here, D_i is the sub-distribution sampling from the previous states and sub-actions.

According to the Eq. (7.9), we can write the corresponding objective equation as Eq. (7.11) in updating each sub-policy parameter.

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}}\left[\frac{1}{m}\sum_{i=1}^{m} \log \pi_{\phi_i}(a_{i_t}|s_t) - Q_\theta(s_t, \mathcal{A}_t)\right] \tag{7.11}$$

Each sub-policy uses the neural network transformation (Eq. (7.12)) in the reparameterization process.

$$a_{i_t} = f_{\phi_i}(\epsilon_t; s_t), \tag{7.12}$$

Algorithm 10: Bayesian Soft Actor-Critic (BSAC)

```
Get initial parameter vectors $\psi, \bar{\psi}, \theta, \phi_1, \cdots, \phi_m$;
for each iteration do
    for each environment step do
        $a_{1_t} \sim \pi_{\phi_1}(a_{1_t}|s_t)$
        $\vdots$
        $a_{m_t} \sim \pi_{\phi_m}(a_{m_t}|s_t)$
        $\mathcal{A}_t \leftarrow \{a_{1_t}, \dots, a_{m_t}\}$
        $s_{t+1} \sim p(s_{t+1}|s_t, \mathcal{A}_t)$
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, \mathcal{A}_t, r(s_t, \mathcal{A}_t), s_{t+1})\}$
    for each gradient step do
        $\psi \leftarrow \psi - \lambda_V \nabla_\psi J_V(\psi)$
        $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i),\ i \in \{1, 2\}$
        $\phi_{1,\dots,m} \leftarrow \phi_{1,\dots,m} - \lambda_\pi \nabla_{\phi_{1,\dots,m}} J_\pi(\phi_{1,\dots,m})$
        $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
```

where ϵ is the input noise vector sampled from some fixed distribution.

Instead of using one policy network with a Gaussian distribution in SAC to fit the distribution of the Q-value, BSAC generates several simple distributions based on the BSN to adapt it to the given model. In other words, we can generate a fitter joint distribution by training the simple sub-policy networks to adapt the corresponding Q-values in current scenarios.

Like the SAC, we also consider two Q-functions to mitigate positive bias in the policy improvement and using the minimum one for the value gradient. Furthermore, we implement a replay buffer collecting experience from the environment with current policy and updating the parameters of the approximators through the stochastic gradients from batches. Finally, the proposed Bayesian Soft Actor-Critic (BSAC) approach is represented using the pseudocode in Alg. 10.

7.4 Experiments and results

We evaluate the performance of the proposed BSAC agent on several challenging continuous control environments with varying action combination and complexity. We choose the MuJoCo physics engine Todorov et al., 2012 to simulate our experiments in the OpenAI's Gym environment Brockman et al.,

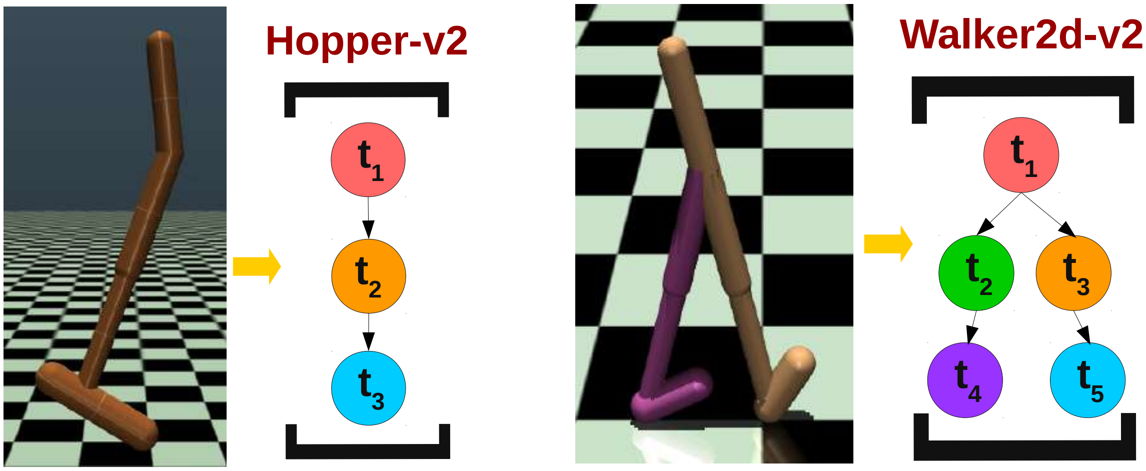


Figure 7.4: The BSN model representing action dependencies implemented on the Hopper-v2 and the Walker2d-v2 domains

2016. We use three of the standard continuous control benchmark domains – Hopper-v2, Walker2d-v2, and Humanoid-v2 in the Gym environment for our experiments. Figure 7.4 shows a sample illustration of the environments used in our experiments and state the corresponding BSN information of these tasks.

We first study the performance of the proposed BSAC against the state-of-the-art continuous control algorithm, the SAC Haarnoja et al., 2018 and other benchmark DRL algorithms, PPO Schulman, Wolski, et al., 2017 and DDPG Lillicrap et al., 2015. All hyperparameters of the models used in our experiments are listed in Appendix C.2. Figure 7.5 shows the total average return of evaluation rollouts during training for BSAC, SAC, DDPG, and PPO.

7.4.1 Hopper-v2 experiments

In this experiment, we decompose the hopper's behaviors into three sub-actions – *Hip Action*, *Knee Action*, and *Ankle Action* – and organize them as a chain in the corresponding BSN (Figure 7.4). In this BSN, the output of factors t_1 and t_2 are the input of factors t_2 and t_3, respectively, expressing the Bayesian relationships between the three actions. In other words, for the joint action of the hopper, we described it as the combination of three distributions corresponding to the

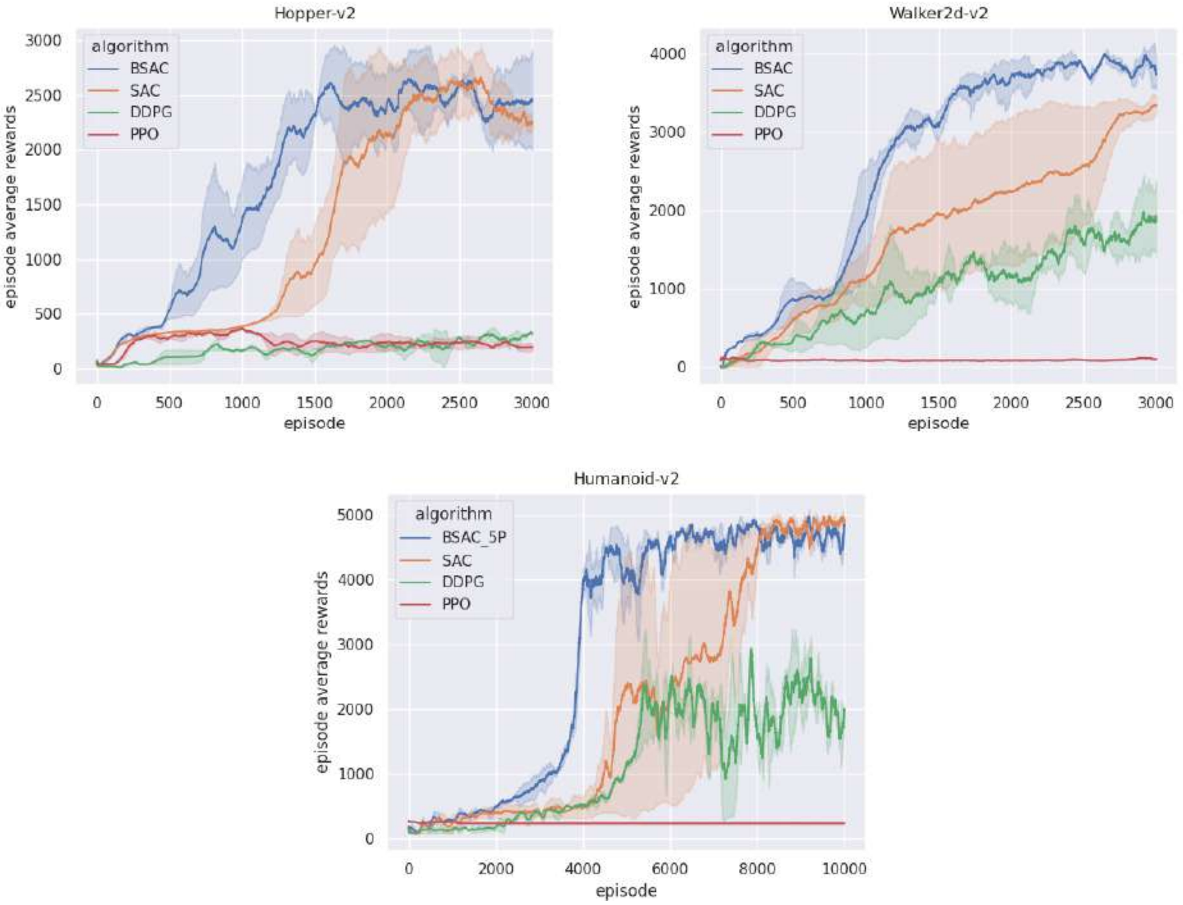


Figure 7.5: Performance comparison of the DRL algorithms in the Hopper-v2 (left), Walker-v2 (center), and Humanoid-v2 (right) domains

joint distribution in the BSN formalized in Eq. (7.13).

$$P(t_1, t_2, t_3) = P(t_1)P(t_2|t_1)P(t_3|t_2) \tag{7.13}$$

Furthermore, in this BSAC model, we implement three sub-policies networks (sub-actors) which can generate the three action distributions – $P(t_1)$, $P(t_2|t_1)$, and $P(t_3|t_2)$, respectively. Then, through sampling from those distributions, we can correspondingly get the hip, knee, and ankle actions and integrate them as one joint action.

Although both the BSAC and SAC outperformed DDPG and PPO in this implementation, the results shown in Figure 7.5 demonstrate that the BSAC competitively perform against SAC, showing faster convergence and higher average rewards per episode. Due to introducing the maximum entropy, our method indicates both sample efficiency and learning stability, compared with other methods.

7.4.2 Walker-v2 experiments

Here, we build a BSN model for the Walker2d domain and decompose the walker's behaviors as five actions – *Hip Action*, *Left Knee Action*, *Right Knee Action*, *Left Ankle Action*, and *Right Ankle Action*. According to the feature of the walker, these actions can be organized as a tree structure in the BSN, which is slightly complex compared to the Hopper-v2 BSN (see Figure 7.4). Similarly, we can formalize the BSN for this Walker domain in Eq. (7.14).

$$\begin{aligned} &P(t_1, t_2, t_3, t_4, t_5) = \\ &P(t_1)P(t_2|t_1)P(t_3|t_1)P(t_4|t_2)P(t_5|t_3) \end{aligned} \tag{7.14}$$

For the corresponding BSAC model, we use five sub-policies networks to approximate the distributions – $P(t_1)$, $P(t_2|t_1)$, $P(t_3|t_1)$, $P(t_4|t_2)$, and $P(t_5|t_3)$ – in the BSN, respectively. Especially, for the factor t_1, the sub-policy needs to generate two actions – *left hip action* and *right hip action* – as an input for the following factors t_2 and t_3. Finally, through sampling from the five distributions, we can integrate them as one joint action for the walker.

Comparing the performance of the BSAC with SAC, DDPG, and PPO in the Walker2d-v2 environment, we can further prove that BSAC can achieve higher performance than other DRL algorithms. From another perspective, as the increasing the complexity of the agent's behaviors and strategy, through decomposing the complex behaviors into simple actions or tactics and organizing them as a suitable BSN, we can build the corresponding joint policy model in the BSAC to substantially increase training efficiency.

7.4.3 Humanoid-v2 experiments

Humanoid-v2 has higher complexity than other domains in the MuJoCo Gym collections. We implement different BSN models in this domain based on composing several parts of the humanoid body, as shown in Figure 7.6. First, we consider the 5 five factor strategy composition (BSAC-5P). Here, we organize the joints of the humanoid as the specific BSN as formalized in Eq. (7.15).

$$\begin{aligned} &P(t_1, t_2, t_3, t_4, t_5) = \\ &P(t_1)P(t_2|t_1)P(t_3|t_1)P(t_4|t_1)P(t_5|t_1) \end{aligned} \tag{7.15}$$

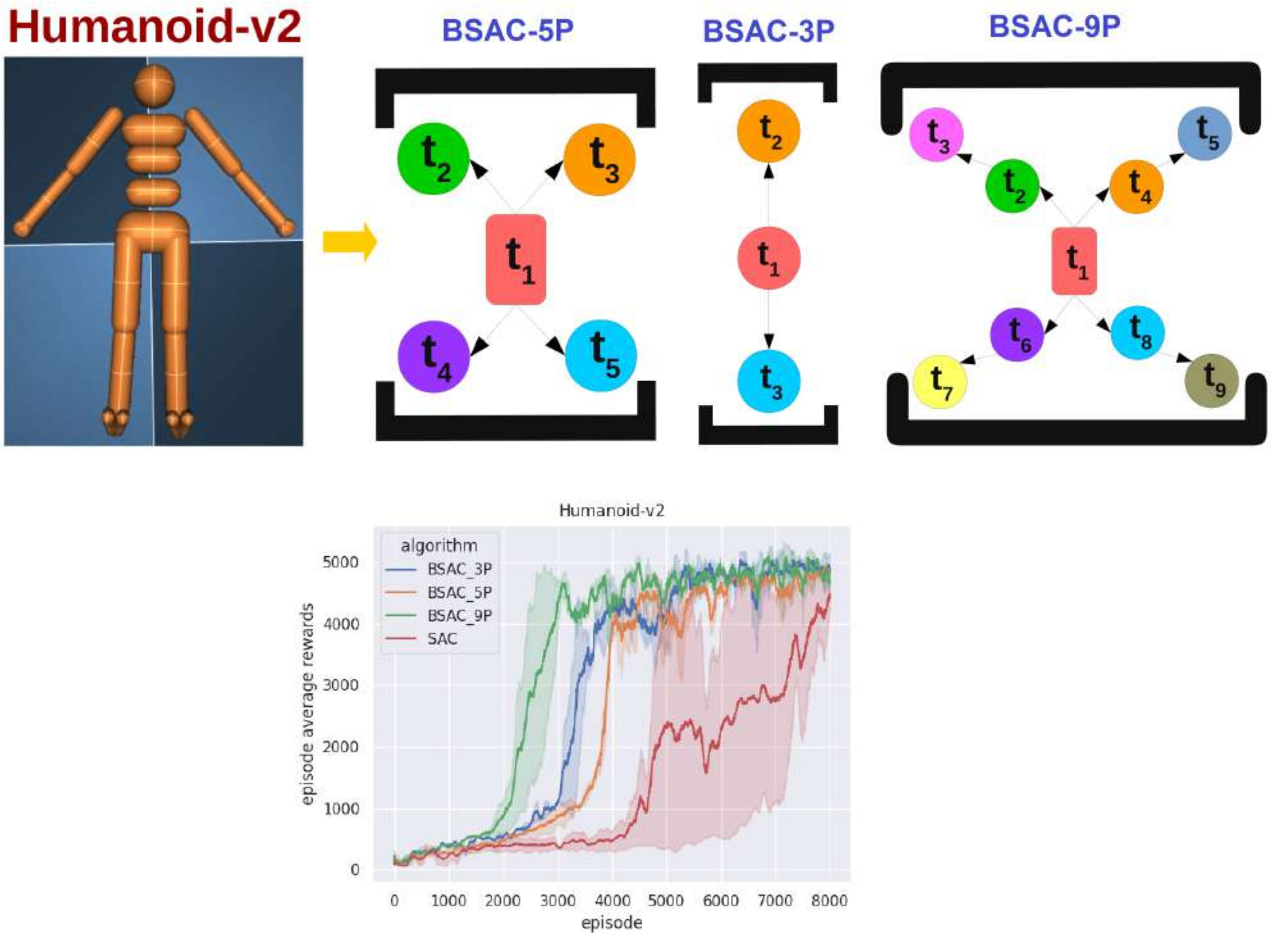


Figure 7.6: The BSN models implemented in the Humanoid-v2 domain showing different strategy decomposition

In the BSAC-5P model, the joint policy is represented as five different sub-policies which generate *Abdomen Action*, *Actions of Right Hip and Right Knee*, *Actions of Left Hip and Left Knee*, *Actions of Right Shoulder and Right Elbow*, and *Actions of Left Shoulder and Left Elbow*, respectively. Figure 7.5 also shows that the BSAC has a faster convergence speed and performance than other methods in our Humanoid-v2 experiments.

Comparing different BSN models

In this section, we consider two other example BSN models corresponding to the three models we designed in the Humanoid-v2 implementation (see Figure 7.6 [44]). For the BSAC three sub-policies model (BSAC-3P), we consider

[44] **Here**, five factors (BSAC-5P), three factors (BSAC-3P), and nine factors (BSAC-9P) are depicted along with their performance comparison results.

that they generate the *Abdomen Actions* distribution $P(t_1)$, the distribution $P(t_2|t_1)$ of the actions of *Shoulder and Elbow*, and the distribution $P(t_3|t_1)$ of the actions of *Hip and Knee*, respectively. Within a sub-space, the actions are independent. For example, the shoulder and elbow joints (from t_2) have no conditional dependencies between each other, but both are dependent on the abdomen joint (t_1).

In the nine networks model (BSAC-9P), we further extend the BSN model and impleFment nine sub-policies to build a joint policy generating the corresponding actions' distributions:

Abdomen $P(t_1)$, *Left shoulder* $P(t_2|t_1)$, *Left Elbow* $P(t_3|t_2)$, *Right shoulder* $P(t_4|t_1)$, *Right Elbow* $P(t_5|t_4)$, *Left Hip* $P(t_6|t_1)$, *Left Knee* $P(t_7|t_6)$, *Right Hip* $P(t_8|t_1)$, and *Right Knee* $P(t_9|t_8)$.

Performance results shown in Figure 7.6) demonstrate that all three BSAC models can achieve higher performance than the SAC. On the other hand, compared within the three BSAC models' performance, the nine sub-policies model BSAC-9P presents more advantages compared to BSAC-5P and BSAC-3P, but the five sub-policies model shows the worst performance, even comparing with BSAC-3P. It implies that the nine sub-policies model can form a joint action distribution more fit for the distribution of the Q-value among the three BSAC models, which means that it describes the relationships between those joints more suitable in the current rewards' mechanism design.

Generally speaking, the reward mechanism plays a crucial role in the agent's training, which directly affects the agent's final behaviors and strategies. Based on the rewards mechanism, the BSAC provides an approach to generating a more suitable joint action or strategy policy to fit the value distribution, which improves the convergence efficiency and the performance of the model. Therefore, ways of designing the BSAC for a specific domain needs to be studied further.

7.5 Conclusions

We introduce a novel agent strategy composition approach termed Bayesian Strategy Network (BSN) for achieving efficient deep reinforcement learning (DRL) methods. Through conditionally coupling the action or tactic dimensions based on a Bayesian network chain, BSN decomposes an intricate strategy or joint action into several simple tactics or actions. By designing corresponding sub-policies networks according to the BSN, we can build a joint policy to generate the complex strategy or action distribution. Then, building on top of the state-of-the-art Soft Actor-Critic (SAC) algorithm, we propose a new DRL model called the Bayesian Soft Actor-Critic (BSAC). BSAC integrates the BSN and forms a joint action distribution to better adapt the Q-value's distribution.

From theoretical derivation, we formulate the training process of the BSAC and implement it in OpenAI's MuJoCo standard continuous control benchmark domains such as the Hopper, Walker, and the Humanoid. The results illustrated the effectiveness of the proposed architecture in enabling the application domains with high-dimensional action spaces and can achieve higher performance against the state-of-the-art RL methods. Furthermore, we believe that the potential generality and practicability of the BSAC evoke further theoretical and empirical investigations. Especially, implementing the BSAC on real robots is not only a challenging problem but will also help us develop robust computation models for multiagent/robot systems.

Chapter 8

Conclusion and Future Works

This thesis started with the principled cooperative MAS/MRS framework and proposed bridging the fourth level automation gap between perception, communication, planning, execution, decision-making, and learning. The motivation is developing a general cooperative MAS/MRS framework that can model the intelligent agents' motivation, present corresponding actions and tactics adapting to various situations, and optimize their behaviors and strategies through learning from interactions with environments and previous experience. Moreover, since the world is intrinsically probabilistic, a probabilistic framework is preferred instead of a deterministic one. In Chapter 2, we discussed the *expected utility theory* and the extension in *Maslow's hierarchy of needs*, the formal *game* and related concepts, the ideas of the *entropy* and the *principle of maximum entropy* in information theory, the formal *probabilistic graphical models (PGM)* formulation for a *Bayesian network model*, and the *deep reinforcement learning (DRL)* when training rewards-driven models.

After discussing the corresponding background, we introduce the new distributed MAS/MRS cooperative architecture – *self-adaptive swarm system (SASS)* and its corresponding modules in Chapter 3. Specifically, we discuss the *agent needs hierarchy* modeling the agent's motivation and offer a priority queue in a distributed *negotiation-agreement mechanism* avoiding plan conflicts effectively. Then, we provide several Atomic Operations to decompose the complex tasks into a series of simple sub-tasks. We evaluate *SASS* through simulating static and dynamic tasks and comparing them with the state-of-the-art collision-aware task assignment method integrated into our framework.

Through further analyzing the diverse behaviors caused by the hierarchical needs of agents, we formalize the hierarchical needs-driven model and prove that the needs-driven cooperation in a heterogeneous robot system enables higher group utility than a homogeneous robot system from the theoretical analysis in Chapter 4. The results of simulations in robot-aided urban search and rescue (USAR) missions verify the proposed needs-driven collaboration and show that heterogeneous multi-robot cooperation can achieve better performance and increase system robustness by reducing uncertainty in task execution.

To achieve better cooperation in MAS/MRS, we proposed a general agent trust model based on *relative needs entropy (RNE)* to measure and analyze the trust levels between agents and groups, representing the similarity of their diverse needs in a specific situation for heterogeneous multi-robot cooperation (Chapter 5). To exemplify its utility, we implement and demonstrate our trust model through experiments simulating a heterogeneous multi-robot grouping task in a persistent urban search and rescue mission consisting of tasks at two levels of difficulty against the state-of-the-art.

Considering MAS/MRS cooperative working in adversarial or dangerous environments, we introduce a new *game-theoretic utility tree (GUT)* for multi-agent decision-making in Chapter 6. The GUT can decompose high-level strategies into executable low-level actions for cooperative MAS decisions and integrate a new payoff measure based on agent needs for real-time strategy games. We evaluate the GUT approach against state-of-the-art methods that greedily rely on rewards of the composite actions. Furthermore, we demonstrated the applicability of the GUT in the *Robotarium* on two different domains – Explore domain and Pursuit-Evasion domain, and the performances verified the effectiveness of the GUT.

For the agents' highest needs – learning, we discussed the hierarchical needs-driven systems learning from two perspectives – *single-agent systems* and *multi-agent systems*– based on the *agent needs hierarchy* in Chapter 7. Especially, organizing the agent's behaviors and actions to represent complex strategies is challenging to learn policies through interaction. Therefore, we introduce a novel agent strategy composition approach termed *Bayesian strategy network (BSN)* for achieving efficient deep reinforcement learning (DRL) methods. We integrate this approach into the state-of-the-art DRL method, soft actor-critic (SAC), and build the corresponding Bayesian soft actor-critic (BSAC) model by organizing each sub-policies as a joint policy. The results demonstrate the promising potential of the BSAC method in terms of training efficiency.

In the course of this research, the *SASS* as a novel DAI model, the planning and control govern the individual low-level *safety* and *basic* needs; *capability* and *teaming* needs are the preconditions and requirements of MAS cooperation in decision-making for achieving tasks; through *learning*, agents can upgrade by itself based on the experiences and lead to the self-evolution of the whole system. Especially, optimizing the communication architecture and adding decision learning levels into the individual agent hierarchical needs model, this improvement will cause the individual agents to present more complex group behaviors adapting various scenarios, to upgrade by themselves based on the learned experiences, and lead to the self-evolution of the whole system.

Then, it will be essential for future work to improve GUT from different perspectives, such as optimizing GUT structure through learning from different scenarios, designing appropriate utility functions, building suitable predictive models, and estimating reasonable parameters fitting the specific scenario. Besides, optimizing GUT structure through learning from different scenarios with reinforcement learning techniques is also an exciting and challenging area, helping us develop more robust computation models for MAS cooperation.

Moreover, how enable robots to learn and adapt to human needs and keep up trust and rapport between humans and robots is critical for task efficiency and safety improvement. Especially, it involves building a stable trust network and evaluating the reliability among agents' interactions, such as human-robot and robot-robot, when we consider the deeper and more complex cooperative relationships in MAS/MRS.

Specifically, *SASS* principles involve designing the basic system architecture and learning algorithms through machine learning, demonstrating them on AI agents (like robots) or sensor networks. These include system performance, robustness, adaptability, generality, and efficiency in scalability. This research also focus on how these basic and large components can be built in a scalable MAS while maintaining performance guarantees. Simultaneously, deeply understanding how to organize multi-layer relationships based on the agent needs hierarchy building in agents' interactions, presenting more complex behaviors or strategies adapting to various is a challenge and exciting topic.

Furthermore, though performance and scalability will remain key in MAS cooperation, we also intend to look at the issue – the robust agent trust network, which will become more relevant. Since compared with the natural intelligent agent, when the AI agent becomes more advanced and intelligent, it also represents more complex, multilayered, and diverse needs in evolution such as individual security, health, friendship, love, respect, recognition. When we consider intelligent agents, like robots, working as a team or cooperating with human beings, organizing their needs building certain reliable and stable relationships such as trust is a precondition for robot-robot and human-robot collaboration in complex and uncertain environments.

More importantly, the adaptive learning of *Human-Robot Interaction* will be considered as the following key factors: 1) adopting suitable formation to perceive and survey environments predicting threats; 2) reasonable path planning adaptation in various scenarios to avoid the collision, guaranteeing human security and decreasing human working environment interference; 3) combining the specific capabilities and needs of robots and humans, calculating sensible strategies to organize the entire group collaboration fulfilling corresponding mission efficiently. On the other hand, it also involves the topics such as AI safety and information security.

Generally, we envisage the field of MAS extended to DAI created from the ground up, building upon the foundations of several fields, including those mentioned above, which can help us model human needs and find a way to calculate the expected utility of the human member in the human-robot team. Finally, integrating human needs with AI agents' needs and capabilities in the proposed SASS framework will result in a harmonious teaming with heterogeneous agents (humans/robots) achieving common goals.

Bibliography

Agmon, N., Kaminka, G. A., & Kraus, S. (2011). Multi-robot adversarial patrolling: Facing a full-knowledge opponent. *Journal of Artificial Intelligence Research*, *42*, 887–916.

Agmon, N., Kraus, S., Kaminka, G. A., & Sadov, V. (2009). Adversarial uncertainty in multi-robot patrol. *Twenty-First International Joint Conference on Artificial Intelligence*.

Ahrenholz, J. (2010). Comparison of core network emulation platforms. *2010-Milcom 2010 Military Communications Conference*, 166–171.

Aknine, S., Pinson, S., & Shakun, M. F. (2004). An extended multi-agent negotiation protocol. *Autonomous Agents and Multi-Agent Systems*, *8*(1), 5–45.

Albrecht, S. V., & Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, *258*, 66–95.

Al-Jarrah, R., Shahzad, A., & Roth, H. (2015). Path planning and motion coordination for multi-robots system using probabilistic neuro-fuzzy. *IFAC-PapersOnLine*, *48*(10), 46–51.

Álvarez-Mozos, M., Van Den Brink, R., Van Der Laan, G., & Tejada, O. (2017). From hierarchies to levels: New solutions for games with hierarchical structure. *International Journal of Game Theory*, *46*(4), 1089–1113.

Amato, C., Chowdhary, G., Geramifard, A., Üre, N. K., & Kochenderfer, M. J. (2013). Decentralized control of partially observable markov decision processes. *52nd IEEE Conference on Decision and Control*, 2398–2405.

Amato, C., Oliehoek, F. A., & Shyu, E. (2013). Scalable bayesian reinforcement learning for multiagent pomdps. *Proc. 1st Multidiscipl. Conf. Reinforcement Learn. Decis. Making*, 188–192.

Andre, D., & Teller, A. (1998). Evolving team darwin united. *Robot Soccer World Cup*, 346–351.

Angeline, P. J., & Pollack, J. B. (1993). Competitive environments evolve better solutions for complex tasks. In S. Forrest (Ed.), *Proceedings of the 5th* *international conference on genetic algorithms, urbana-champaign, il, usa, june 1993* (pp. 264–270). Morgan Kaufmann.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, *34*(6), 26–38.

Asama, H., Matsumoto, A., & Ishida, Y. (1989). Design of an autonomous and distributed robot system: Actress. *Proceedings. IEEE/RSJ International Workshop on Intelligent Robots and Systems '. (IROS '89) 'The Autonomous Mobile Robots and Its Applications*, 283–290. https://doi.org/10.1109/IROS.1989.637920

Ashrafiuon, H., Muske, K. R., & McNinch, L. C. (2010). Review of nonlinear tracking and setpoint control approaches for autonomous underactuated marine vehicles. *Proceedings of the 2010 American Control Conference*, 5203–5211.

Åström, K. J. (1965). Optimal control of markov processes with incomplete state information. *Journal of mathematical analysis and applications*, *10*(1), 174–205.

Atınç, G. M., Stipanović, D. M., & Voulgaris, P. G. (2014). Supervised coverage control of multi-agent systems. *Automatica*, *50*(11), 2936–2942.

Aumann, R. J., & Dreze, J. H. (1974). Cooperative games with coalition structures. *International Journal of game theory*, *3*(4), 217–237.

Ayanian, N., & Kumar, V. (2010). Abstractions and controllers for groups of robots in environments with obstacles. *2010 IEEE International Conference on Robotics and Automation*, 3537–3542.

Balch, T. et al. (1997). Learning roles: Behavioral diversity in robot teams. *AAAI Workshop on Multiagent Learning*.

Balch, T. (1997). Social entropy: A new metric for learning multi-robot teams.

Balch, T. (1999). Reward and diversity in multirobot foraging. *IJCAI-99 Workshop on Agents Learning About, From and With other Agents*.

Balch, T., & Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. *IEEE transactions on robotics and automation*, *14*(6), 926–939.

Banzhaf, W. (2009). Self-organizing systems. *Encyclopedia of complexity and systems science*, *14*, 589.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., Tb, D., Muldal, A., Heess, N., & Lillicrap, T. (2018). Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*.

Bassett, R., & Deride, J. (2019). Maximum a posteriori estimators as a limit of bayes estimators. *Mathematical Programming*, *174*(1), 129–144.

Bayat, B., Crasta, N., Crespi, A., Pascoal, A. M., & Ijspeert, A. (2017). Environmental monitoring using autonomous vehicles: A survey of recent searching techniques. *Current opinion in biotechnology*, *45*, 76–84.

Beni, G., & Wang, J. (1993). Swarm intelligence in cellular robotic systems. *Robots and biological systems: Towards a new bionics?* (pp. 703–712). Springer.

Bonabeau, E., Theraulaz, G., Dorigo, M., Theraulaz, G., Marco, D. d. R. D. F., et al. (1999). *Swarm intelligence: From natural to artificial systems*. Oxford university press.

Boutilier, C. (2013). Learning conventions in multiagent stochastic domains using likelihood estimates. *arXiv preprint arXiv:1302.3561*.

Boyan, J., & Littman, M. (2000). Exact solutions to time-dependent mdps. *Advances in Neural Information Processing Systems*, *13*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym.

Browning, B., Bruce, J., Bowling, M., & Veloso, M. (2005). Stp: Skills, tactics, and plays for multi-robot control in adversarial environments. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *219*(1), 33–52.

Brunete, A., Hernando, M., Gambao, E., & Torres, J. E. (2012). A behaviour-based control architecture for heterogeneous modular, multi-configurable, chained micro-robots. *Robotics and Autonomous Systems*, *60*(12), 1607–1624.

Buro, M. (2003). Real-time strategy games: A new ai research challenge. *IJCAI*, *2003*, 1534–1535.

Byrne, C. M., Rodriguez, D. T., & Franklin, M. J. (2012). A study of the feasibility of autonomous surface vehicles.

Campbell, S., Naeem, W., & Irwin, G. W. (2012). A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres. *Annual Reviews in Control*, *36*(2), 267–283.

Cao, Y. U., Kahng, A. B., & Fukunaga, A. S. (1997). Cooperative mobile robotics: Antecedents and directions. *Robot colonies* (pp. 7–27). Springer.

Cao, Y., Yu, W., Ren, W., & Chen, G. (2012). An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, *9*(1), 427–438.

Cardona, G. A., & Calderon, J. M. (2019). Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations. *Applied Sciences*, *9*(8), 1702.

Cassandra, A. R. (1998). *Exact and approximate algorithms for partially observable markov decision processes*. Brown University.

Celli, A., & Gatti, N. (2018). Computational results for extensive-form adversarial team games. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1).

Chalkiadakis, G., & Boutilier, C. (2003). Coordination in multiagent reinforcement learning: A bayesian approach. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 709–716.

Chan, S. (2001). Complex adaptive systems. *ESD. 83 research seminar in engineering systems*, *31*, 1–9.

Chen, R., Bao, F., & Guo, J. (2015). Trust-based service management for social internet of things systems. *IEEE transactions on dependable and secure computing*, *13*(6), 684–696.

Chen, Y., & Sun, J. (2016). Distributed optimal control for multi-agent systems with obstacle avoidance. *Neurocomputing*, *173*, 2014–2021.

Cho, J.-H. (2014). Tradeoffs between trust and survivability for mission effectiveness in tactical networks. *IEEE transactions on cybernetics*, *45*(4), 754–766.

Cho, J.-H., Chan, K., & Adali, S. (2015). A survey on trust modeling. *ACM Computing Surveys (CSUR)*, *48*(2), 1–40.

Cho, J.-H., Swami, A., & Chen, R. (2010). A survey on trust management for mobile ad hoc networks. *IEEE communications surveys & tutorials*, *13*(4), 562–583.

Chung, T. H., Hollinger, G. A., & Isler, V. (2011). Search and pursuit-evasion in mobile robotics. *Autonomous robots*, *31*(4), 299.

Colledanchise, M., & Ögren, P. (2018). *Behavior trees in robotics and ai: An introduction*. CRC Press.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. MIT press.

Cover, T. M., & Thomas, J. A. (1991). Entropy, relative entropy and mutual information. *Elements of information theory*, *2*(1), 12–13.

Da Silva, F. L., & Costa, A. H. R. (2019). A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, *64*, 645–703.

Daskalakis, C., Goldberg, P. W., & Papadimitriou, C. H. (2009). The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, *39*(1), 195–259.

De Cubber, G., Doroftei, D., Serrano, D., Chintamani, K., Sabino, R., & Ourevitch, S. (2013). The eu-icarus project: Developing assistive robotic tools for search and rescue operations. *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 1–4.

Desaraju, V. R., & How, J. P. (2012). Decentralized path planning for multiagent teams with complex constraints. *Autonomous Robots*, *32*(4), 385–403.

Dias, M. B., & Stentz, A. (2000). A free market architecture for distributed control of a multirobot system.

Duan, H., & Qiao, P. (2014). Pigeon-inspired optimization: A new swarm intelligence optimizer for air robot path planning. *International journal of intelligent computing and cybernetics*.

El Oula Frihi, Z., Barreiro-Gomez, J., Eddine Choutri, S., & Tembine, H. (2020). Hierarchical structures and leadership design in mean-field-type games with polynomial cost. *Games*, *11*(3), 30.

Elkind, E., & Rothe, J. (2016). Cooperative game theory. *Economics and computation* (pp. 135–193). Springer.

Emery-Montemerlo, R., Gordon, G., Schneider, J., & Thrun, S. (2005). Game theoretic control for robot teams. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 1163–1169.

Ermon, S. (2021). Structure learning for bayesian networks. Retrieved 2021, from https://ermongroup.github.io/cs228-notes/learning/structure/

Etro, F. (2013). Stackelberg, heinrich von: Market structure and equilibrium.

Fahad, M., Guo, Y., Bingham, B., Krasnosky, K., Fitzpatrick, L., & Sanabria, F. A. (2017). Evaluation of ocean plume characteristics using unmanned surface vessels. *OCEANS 2017-Anchorage*, 1–7.

Farinelli, A., Iocchi, L., Nardi, D., & Ziparo, V. A. (2006). Assignment of dynamically perceived tasks by token passing in multirobot systems. *Proceedings of the IEEE*, *94*(7), 1271–1288.

Feddema, J. T., Lewis, C., & Schoenwald, D. A. (2002). Decentralized control of cooperative robotic vehicles: Theory and application. *IEEE Transactions on robotics and automation*, *18*(5), 852–864.

Ficici, S. G., & Pollack, J. B. (2000). A game-theoretic approach to the simple coevolutionary algorithm. *International Conference on Parallel Problem Solving from Nature*, 467–476.

Fierro, R., Das, A., Spletzer, J., Esposito, J., Kumar, V., Ostrowski, J. P., Pappas, G., Taylor, C. J., Hur, Y., Alur, R., et al. (2002). A framework and architecture for multi-robot coordination. *The International Journal of Robotics Research*, *21*(10-11), 977–995.

Fishburn, P. C. (1970). *Utility theory for decision making* (tech. rep.). Research analysis corp McLean VA.

Fornai, F., Ferri, G., Manzi, A., Ciuchi, F., Bartaloni, F., & Laschi, C. (2016). An autonomous water monitoring and sampling system for small-sized asvs. *IEEE Journal of Oceanic Engineering*, *42*(1), 5–12.

Fossen, T. I. (2000). A survey on nonlinear ship control: From theory to practice. *IFAC Proceedings Volumes*, *33*(21), 1–16.

Freedman, L. (2015). *Strategy: A history*. Oxford University Press.

Fudenberg, D., Drew, F., Levine, D. K., & Levine, D. K. (1998). *The theory of learning in games* (Vol. 2). MIT press.

Fullam, K. K., Klos, T. B., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, K. S., Rosenschein, J. S., Vercouter, L., & Voss, M. (2005). A specification of the agent reputation and trust (art) testbed: Experimentation and competition for trust in agent societies. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 512–518.

Gal, Y., & Pfeffer, A. (2008). Networks of influence diagrams: A formalism for representing agents' beliefs and decision-making processes. *Journal of Artificial Intelligence Research*, *33*, 109–147.

Gazi, V., & Passino, K. M. (2003). Stability analysis of swarms. *IEEE transactions on automatic control*, *48*(4), 692–697.

Geihs, K. (2020). Engineering challenges ahead for robot teamwork in dynamic environments. *Applied Sciences*, *10*(4), 1368.

Gilles, R. P., Owen, G., & van den Brink, R. (1992). Games with permission structures: The conjunctive approach. *International Journal of Game Theory*, *20*(3), 277–293.

Glorennec, P.-Y. (1997). Coordination between autonomous robots. *International Journal of Approximate Reasoning*, *17*(4), 433–446.

Gmytrasiewicz, P. J., & Doshi, P. (2004). Interactive pomdps: Properties and preliminary results. *International Conference on Autonomous Agents: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-*, *3*, 1374–1375.

Goldberg, D., Cicirello, V., Dias, M. B., Simmons, R., Smith, S., Smith, T., & Stentz, A. (2002). A distributed layered architecture for mobile robot coordination: Application to space exploration. *The 3rd International NASA Workshop on Planning and Scheduling for Space*.

Gonzalez, J. E. (2007). Inference in bayesian networks. Retrieved 2007, from http://www.cs.cmu.edu/~guestrin/Class/10701/recitations/r9/var_elim_recitation_unanimated.pdf

Guerrero-González, A., Garcıéa-Córdova, F., Ortiz, F. J., Alonso, D., & Gilabert, J. (2016). A multirobot platform based on autonomous surface and underwater vehicles with bio-inspired neurocontrollers for long-term oil spills monitoring. *Autonomous Robots*, *40*(7), 1321–1342.

Haarnoja, T. (2018). *Acquiring diverse robot skills via maximum entropy deep reinforcement learning*. University of California, Berkeley.

Haarnoja, T., Tang, H., Abbeel, P., & Levine, S. (2017). Reinforcement learning with deep energy-based policies. *International Conference on Machine Learning*, 1352–1361.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International conference on machine learning*, 1861–1870.

Hamann, H. (2018). Swarm robotics: A formal approach.

Hamann, H., & Wörn, H. (2008). A framework of space–time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, *2*(2-4), 209–239.

Hancock, P. A., Billings, D. R., Schaefer, K. E., Chen, J. Y., De Visser, E. J., & Parasuraman, R. (2011). A meta-analysis of factors affecting trust in human-robot interaction. *Human factors*, *53*(5), 517–527.

Haoran Tang, T. H. (2017). Learning diverse skills via maximum entropy deep reinforcement learning. https://bair.berkeley.edu/blog/2017/10/06/soft-q-learning/

Hasselt, H. (2010). Double q-learning. *Advances in neural information processing systems*, *23*.

Hausknecht, M., & Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. *2015 aaai fall symposium series*.

Hayes, A. (2022). What is game theory? https://www.investopedia.com/terms/g/gametheory.asp

Haynes, T., Lau, K., & Sen, S. (1996). Learning cases to compliment rules for conflict resolution in multiagent systems. *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, 51–56.

Haynes, T., & Sen, S. (1995). Evolving behavioral strategies in predators and prey. *International Joint Conference on Artificial Intelligence*, 113–126.

Hengst, B. (2012). Hierarchical approaches. *Reinforcement learning* (pp. 293–323). Springer.

Hernandez-Leal, P., Kartal, B., & Taylor, M. E. (2019). A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, *33*(6), 750–797.

Heylighen, F. (2017). Distributed intelligence technologies: Present and future applications. *The Future Information Society: Social and Technological Problems*, 179–212.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Ichikawa, S., Hara, F., & Hosokai, H. (1993). Cooperative route-searching behavior of multi-robot system using hello-call communication. *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, *2*, 1149–1156.

Ilyas, A., Engstrom, L., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., & Madry, A. (2018). Are deep policy gradient algorithms truly policy gradient algorithms?

Ismail, Z. H., Sariff, N., & Hurtado, E. (2018). A survey and analysis of cooperative multi-agent robot systems: Challenges and directions. *Applications of mobile robots* (pp. 8–14). IntechOpen.

Jansen, T., & Wiegand, R. P. (2003). Exploring the explorative advantage of the cooperative coevolutionary (1+ 1) ea. *Genetic and Evolutionary Computation Conference*, 310–321.

Jaynes, E. T. (1957a). Information theory and statistical mechanics. *Physical review*, *106*(4), 620.

Jaynes, E. T. (1957b). Information theory and statistical mechanics. ii. *Physical review*, *108*(2), 171.

Jiang, A. X., & Leyton-Brown, K. (2009). A tutorial on the proof of the existence of nash equilibria. *University of British Columbia Technical Report TR-2007-25. pdf*, *14*.

Jorge, V. A., Granada, R., Maidana, R. G., Jurak, D. A., Heck, G., Negreiros, A. P., dos Santos, D. H., Gonçalves, L. M., & Amory, A. M. (2019). A survey on unmanned surface vehicles for disaster robotics: Main challenges and directions. *Sensors*, *19*(3), 702.

Jun, M., & D'Andrea, R. (2003). Path planning for unmanned aerial vehicles in uncertain and adversarial environments. *Cooperative control: Models, applications and algorithms* (pp. 95–110). Springer.

Jung, D., & Zelinsky, A. (1999). An architecture for distributed cooperative planning in a behaviour-based multi-robot system. *Robotics and Autonomous Systems*, *26*(2-3), 149–174.

Kaminka, G. A., & Frenkel, I. (2005). Flexible teamwork in behavior-based robots. *Proceedings Of The National Conference On Artificial Intelligence*, *20*(1), 108.

Karni, E. (2017). Savage's subjective expected utility model. *The new palgrave dictionary of economics* (pp. 1–5). Palgrave Macmillan UK. https://doi.org/10.1057/978-1-349-95121-5_2467-1

Keidar, O., & Agmon, N. (2018). Safe navigation in adversarial environments. *Annals of Mathematics and Artificial Intelligence*, *83*(2), 121–164.

Kennedy, J. (2006). Swarm intelligence. *Handbook of nature-inspired and innovative computing* (pp. 187–219). Springer.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95-international conference on neural networks*, *4*, 1942–1948.

KhanAcademy. (2016). What is the maxwell-boltzmann distribution? Retrieved 2016, from https://www.khanacademy.org/science/physics/thermodynamics/temp-kinetic-theory-ideal-gas-law/a/what-is-the-maxwell-boltzmann-distribution

Khavas, Z. R., Ahmadzadeh, S. R., & Robinette, P. (2020). Modeling trust in human-robot interaction: A survey. *International Conference on Social Robotics*, 529–541.

Kiener, J., & Von Stryk, O. (2010). Towards cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots. *Robotics and Autonomous Systems*, *58*(7), 921–929.

Kim, J. (2018). Multirobot exploration while building power-efficient sensor networks in three dimensions. *IEEE transactions on cybernetics*, *49*(7), 2771–2778.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kloder, S., & Hutchinson, S. (2006). Path planning for permutation-invariant multirobot formations. *IEEE Transactions on Robotics*, *22*(4), 650–665.

Kochenderfer, M. J. (2015). *Decision making under uncertainty: Theory and application*. MIT press.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT press.

Koller, D., & Milch, B. (2003). Multi-agent influence diagrams for representing and solving games. *Games and economic behavior*, *45*(1), 181–221.

Kolling, A., & Carpin, S. (2010). Multi-robot pursuit-evasion without maps. *2010 IEEE International Conference on Robotics and Automation*, 3045–3051.

Konda, V. R., & Tsitsiklis, J. N. (2003). Onactor-critic algorithms. *SIAM journal on Control and Optimization*, *42*(4), 1143–1166.

Kothari, M., Manathara, J. G., & Postlethwaite, I. (2014). A cooperative pursuit-evasion game for non-holonomic systems. *IFAC Proceedings Volumes*, *47*(3), 1977–1984.

Kruijff-Korbayová, I., Colas, F., Gianni, M., Pirri, F., de Greeff, J., Hindriks, K., Neerincx, M., Ögren, P., Svoboda, T., & Worst, R. (2015). Tradr project: Long-term human-robot teaming for robot assisted disaster response. *KI-Künstliche Intelligenz*, *29*(2), 193–201.

Kuniyoshi, Y., Rickki, J., Ishii, M., Rougeaux, S., Kita, N., Sakane, S., & Kakikura, M. (1994). Vision-based behaviors for multi-robot cooperation. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, *2*, 925–932.

Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, *10*(2), 141–160.

Lattanzi, D., & Miller, G. (2017). Review of robotic infrastructure inspection systems. *Journal of Infrastructure Systems*, *23*(3), 04017004.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Lee, J. D., & See, K. A. (2004). Trust in automation: Designing for appropriate reliance. *Human factors*, *46*(1), 50–80.

Levin, S. A. (1998). Ecosystems and the biosphere as complex adaptive systems. *Ecosystems*, *1*(5), 431–436.

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, *17*(1), 1334–1373.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., & Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, *37*(4-5), 421–436.

Lewicki, R. J., Tomlinson, E. C., & Gillespie, N. (2006). Models of interpersonal trust development: Theoretical approaches, empirical evidence, and future directions. *Journal of management*, *32*(6), 991–1022.

Li, J., & Li, J. (2015). Iterative learning control approach for a kind of heterogeneous multi-agent systems with distributed initial state learning. *Applied Mathematics and Computation*, *265*, 1044–1057.

Li, W., & Shen, W. (2011). Swarm behavior control of mobile multi-robots with wireless sensor networks. *Journal of Network and Computer Applications*, *34*(4), 1398–1407.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, E. S., Agmon, N., & Kraus, S. (2019). Multi-robot adversarial patrolling: Handling sequential attacks. *Artificial Intelligence*, *274*, 1–25.

Liu, D., Luan, X., Zhang, F., Jin, J., Guo, J., & Zheng, R. (2016). An usv-based laser fluorosensor for oil spill detection. *2016 10th International Conference on Sensing Technology (ICST)*, 1–4.

Liu, R., Nageotte, F., Zanne, P., de Mathelin, M., & Dresp-Langley, B. (2021). Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review. *Robotics*, *10*(1), 22.

Liu, S., Lever, G., Wang, Z., Merel, J., Eslami, S., Hennes, D., Czarnecki, W. M., Tassa, Y., Omidshafiei, S., Abdolmaleki, A., et al. (2021). From motor control to team play in simulated humanoid football. *arXiv preprint arXiv:2105.12196*.

Lopez-Franco, M., Sanchez, E. N., Alanis, A. Y., Lopez-Franco, C., & Arana-Daniel, N. (2015). Decentralized control for stabilization of nonlinear multi-agent systems using neural inverse optimal control. *Neurocomputing*, *168*, 81–91.

Luke, S., & Spector, L. (1996). Evolving teamwork and coordination with genetic programming. *Genetic Programming*, *96*, 150–56.

Luo, W., Chakraborty, N., & Sycara, K. (2016). Distributed dynamic priority assignment and motion planning for multiple mobile robots with kinodynamic constraints. *2016 American Control Conference (ACC)*, 148–154.

Ma, H., Hönig, W., Cohen, L., Uras, T., Xu, H., Kumar, T. S., Ayanian, N., & Koenig, S. (2017). Overview: A hierarchical framework for plan generation and execution in multirobot systems. *IEEE Intelligent Systems*, *32*(6), 6–12.

Magid, E., Pashkin, A., Simakov, N., Abbyasov, B., Suthakorn, J., Svinin, M., & Matsuno, F. (2020). Artificial intelligence based framework for robotic search and rescue operations conducted jointly by international teams. *Proceedings of 14th International Conference on Electromechanics and Robotics "Zavalishin's Readings"*, 15–26.

Makkapati, V. R., & Tsiotras, P. (2019). Optimal evading strategies and task allocation in multi-player pursuit–evasion problems. *Dynamic Games and Applications*, 1–20.

Marconi, L., Leutenegger, S., Lynen, S., Burri, M., Naldi, R., & Melchiorri, C. (2013). Ground and aerial robots as an aid to alpine search and rescue:

Initial sherpa outcomes. *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 1–2.

Marques, F., Lourenço, A., Mendonça, R., Pinto, E., Rodrigues, P., Santana, P., & Barata, J. (2015). A critical survey on marsupial robotic teams for environmental monitoring of water bodies. *OCEANS 2015-Genova*, 1–6.

Martinoli, A. (1999). *Swarm intelligence in autonomous collective robotics: From tools to the analysis and synthesis of distributed control strategies* (Doctoral dissertation). Citeseer.

Maslow, A. H. (1943). A theory of human motivation. *Psychological review*, *50*(4), 370.

Maslow, A. H. (1981). *Motivation and personality*. Prabhat Prakashan.

Mataric, M. J. (1993). Designing emergent behaviors: From local interactions to collective intelligence. *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, 432–441.

Mataric, M. J. (1994). Learning to behave socially. *From animals to animats*, *3*, 453–462.

Matignon, L., Laurent, G. J., & Le Fort-Piat, N. (2012). Independent reinforcement learners in cooperative markov games: A survey regarding coordination problems. *The Knowledge Engineering Review*, *27*(1), 1–31.

McKelvey, R. D., McLennan, A. M., & Turocy, T. L. (2006). *Gambit: Software tools for game theory*. Version 0.2006. 01.20.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Mitchell, M., Crutchfield, J. P., Das, R., et al. (1996). Evolving cellular automata with genetic algorithms: A review of recent work. *Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96)*, *8*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *International conference on machine learning*, 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529–533.

Mohammadi, V., Rahmani, A. M., Darwesh, A. M., & Sahafi, A. (2019). Trust-based recommendation systems in internet of things: A systematic literature review. *Human-centric Computing and Information Sciences*, *9*(1), 1–61.

Mongin, P. (1998). Expected utility theory.

Mongin, P., & d'Aspremont, C. (1998). Utility theory and ethics.

Moud, H. I., Shojaei, A., & Flood, I. (2018). Current and future applications of unmanned surface, underwater, and ground vehicles in construction. *Proceedings of the Construction Research Congress*, 106–115.

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT press.

Murphy, R. R. (2014). *Disaster robotics*. MIT press.

Murphy, R. R., Tadokoro, S., & Kleiner, A. (2016). Disaster robotics. In B. Siciliano & O. Khatib (Eds.), *Springer handbook of robotics* (pp. 1577–1604). Springer International Publishing. https://doi.org/10.1007/978-3-319-32552-1_60

Murray, R. M. (2007). Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, *129*(5), 571–583.

Myerson, R. B. (2013). *Game theory*. Harvard university press.

Nadarajah, S., & Sundaraj, K. (2013). A survey on team strategies in robot soccer: Team strategies and role description. *Artificial Intelligence Review*, *40*(3), 271–304.

Nash, J. (1951). Non-cooperative games. *Annals of mathematics*, 286–295.

Nash, J. F. et al. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, *36*(1), 48–49.

Nourbakhsh, I. R., Sycara, K., Koes, M., Yong, M., Lewis, M., & Burion, S. (2005). Human-robot teaming for search and rescue. *IEEE Pervasive Computing*, *4*(1), 72–79.

Nowak, M. A., & Sigmund, K. (1998). Evolution of indirect reciprocity by image scoring. *Nature*, *393*(6685), 573–577.

Oleiwi, B. K., Al-Jarrah, R., Roth, H., & Kazem, B. I. (2015). Integrated motion planning and control for multi objectives optimization and multi robots navigation. *IFAC-PapersOnLine*, *48*(10), 99–104.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*, *51*(3), 401–420.

Otte, M., Kuhlman, M. J., & Sofge, D. (2019). Auctions for multi-robot task allocation in communication limited environments. *Autonomous Robots*, 1–38.

Panait, L., & Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, *11*(3), 387–434.

Panait, L., Wiegand, R. P., & Luke, S. (2004). A visual demonstration of convergence properties of cooperative coevolution. *International Conference on Parallel Problem Solving from Nature*, 892–901.

Parasuraman, R., Kim, J., Luo, S., & Min, B.-C. (2018). Multipoint rendezvous in multirobot systems. *IEEE transactions on cybernetics*.

Parasuraman, R., Pagala, P., Kershaw, K., & Ferre, M. (2012). Energy management module for mobile robots in hostile environments. *Conference Towards Autonomous Robotic Systems*, 430–431.

Parker, L. E. (1994a). Alliance: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots. *Proceedings of IEEE/RSJ international conference on intelligent robots and systems (IROS'94)*, *2*, 776–783.

Parker, L. E. (1994b). *Heterogeneous multi-robot cooperation* (tech. rep.). Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab.

Parker, L. E. (1998). Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE transactions on robotics and automation*, *14*(2), 220–240.

Parker, L. E. (2007). Distributed intelligence: Overview of the field and its application in multi-robot systems. *AAAI Fall Symposium: Regarding the Intelligence in Distributed Intelligent Systems*, 1–6.

Parker, L. E. (2008). Multiple mobile robot systems. *Springer Handbook of Robotics*, 921–941.

Parker, L. E., & Tang, F. (2006). Building multirobot coalitions through automated task solution synthesis. *Proceedings of the IEEE*, *94*(7), 1289–1305.

Pateria, S., Subagdja, B., Tan, A.-h., & Quek, C. (2021). Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, *54*(5), 1–35.

Paulos, J., Chen, S. W., Shishika, D., & Kumar, V. (2019). Decentralization of multiagent policies by learning what to communicate. *arXiv preprint arXiv:1901.08490*.

Pham, D. T., Awadalla, M. H., & Eldukhri, E. E. (2006). Fuzzy and neuro-fuzzy based co-operative mobile robots. *Intelligent production machines and systems* (pp. 578–583). Elsevier.

Pickem, D., Glotfelter, P., Wang, L., Mote, M., Ames, A., Feron, E., & Egerstedt, M. (2017). The robotarium: A remotely accessible swarm robotics research testbed. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1699–1706.

Polydoros, A. S., & Nalpantidis, L. (2017). Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, *86*(2), 153–173.

Posadas, J. L., Poza, J. L., Simó, J. E., Benet, G., & Blanes, F. (2008). Agent-based distributed architecture for mobile robot control. *Engineering Applications of Artificial Intelligence*, *21*(6), 805–823.

Puterman, M. L. (2014). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons.

Quinn, M. (2001). A comparison of approaches to the evolution of homogeneous multi-robot teams. *Proceedings of the 2001 Congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, *1*, 128–135.

Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., & Whiteson, S. (2018). QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. *Proceedings of the 35th International Conference on Machine Learning*.

Rathour, S. S., Kato, N., Tanabe, N., Senga, H., Yoshie, M., & Tanaka, T. (2015). Sea experiments on autonomous tracking of oil spill using a robotic platform. *Proceedings of the 23rd Ocean Engineering Symposium, Tokyo, Japan*.

Ray, P. P. (2016). Internet of robotic things: Concept, technologies, and challenges. *IEEE access*, *4*, 9489–9500.

Ren, W., & Cao, Y. (2011). *Distributed coordination of multi-agent networks: Emergent problems, models, and issues* (Vol. 1). Springer.

Reynolds, C. W. (1987). *Flocks, herds and schools: A distributed behavioral model* (Vol. 21). ACM.

Rizk, Y., Awad, M., & Tunstel, E. W. (2018). Decision making in multiagent systems: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, *10*(3), 514–529.

Rizk, Y., Awad, M., & Tunstel, E. W. (2019). Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, *52*(2), 1–31.

Rosa, L., Cognetti, M., Nicastro, A., Alvarez, P., & Oriolo, G. (2015). Multi-task cooperative control in a heterogeneous ground-air robot team. *IFAC-PapersOnLine*, *48*(5), 53–58.

Russell, S., & Norvig, P. (2002). Artificial intelligence: A modern approach.

Salustowicz, R., Wiering, M., & Schmidhuber, J. (1997). Learning team strategies with multiple policy-sharing agents: A soccer case study. *Technical Report IDSIA*, *29*, 1–20.

Sałustowicz, R. P., Wiering, M. A., & Schmidhuber, J. (1998). Learning team strategies: Soccer case studies. *Machine Learning*, *33*(2), 263–282.

Sanghvi, N., Nagavalli, S., & Sycara, K. (2017). Exploiting robotic swarm characteristics for adversarial subversion in coverage tasks. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Santa Fe, I. (2021). When is a basin of attraction like an octopus? www.sciencedaily.com/releases/2021/11/211104115355.htm

Savage, L. J. (1972). *The foundations of statistics*. Courier Corporation.

Schiaretti, M., Chen, L., & Negenborn, R. R. (2017). Survey on autonomous surface vessels: Part i-a new detailed definition of autonomy levels. *International Conference on Computational Logistics*, 219–233.

Schulman, J., Chen, X., & Abbeel, P. (2017). Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. *International conference on machine learning*, 1889–1897.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shao, K., Tang, Z., Zhu, Y., Li, N., & Zhao, D. (2019). A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:1912.10944*.

Shapira, Y., & Agmon, N. (2015). Path planning for optimizing survivability of multi-robot formation in adversarial environments. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4544–4549.

Shen, J., Zhang, X., & Lesser, V. (2004). Degree of local cooperation and its implication on global utility. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 546–553.

Sherchan, W., Nepal, S., & Paris, C. (2013). A survey of trust in social networks. *ACM Computing Surveys (CSUR)*, *45*(4), 1–33.

Sherrington, D., & Kirkpatrick, S. (1975). Solvable model of a spin-glass. *Physical review letters*, *35*(26), 1792.

Shivam, S., Kanellopoulos, A., Vamvoudakis, K. G., & Wardi, Y. (2019). A predictive deep learning approach to output regulation: The case of collaborative pursuit evasion. *arXiv preprint arXiv:1909.00893*.

Shoham, Y., & Leyton-Brown, K. (2008). *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

Simonin, O., & Grunder, O. (2009). A cooperative multi-robot architecture for moving a paralyzed robot. *Mechatronics*, *19*(4), 463–470.

Singh, B., Kumar, R., & Singh, V. P. (2021). Reinforcement learning in robotic applications: A comprehensive survey. *Artificial Intelligence Review*, 1–46.

Smith, R. G., & Davis, R. (1981). Frameworks for cooperation in distributed problem solving. *IEEE Transactions on systems, man, and cybernetics*, *11*(1), 61–70.

Soysal, O., & Şahin, E. (2006). A macroscopic model for self-organized aggregation in swarm robotic systems. *International Workshop on Swarm Robotics*, 27–42.

Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, *8*(3), 345–383.

Sugawara, K., & Sano, M. (1997). Cooperative acceleration of task performance: Foraging behavior of interacting multi-robots system. *Physica D: Nonlinear Phenomena*, *100*(3-4), 343–354.

Suryadi, D., & Gmytrasiewicz, P. J. (1999). Learning models of other agents using influence diagrams. *Um99 user modeling* (pp. 223–232). Springer.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Swinth, R. L. (1967). The establishment of the trust relationship. *Journal of conflict resolution*, *11*(3), 335–344.

Tambe, M. (1997). Towards flexible teamwork. *Journal of artificial intelligence research*, *7*, 83–124.

Tambe, M., Pynadath, D. V., Chauvat, N., Das, A., & Kaminka, G. A. (2000). Adaptive agent integration architectures for heterogeneous team members. *Proceedings Fourth International Conference on MultiAgent Systems*, 301–308.

Tardioli, D., Parasuraman, R., & Ögren, P. (2019). Pound: A multi-master ros node for reducing delay and jitter in wireless multi-robot networks. *Robotics and Autonomous Systems*, *111*, 73–87.

Tavakoli, A., Pardo, F., & Kormushev, P. (2018). Action branching architectures for deep reinforcement learning. *AAAI Conference on Artificial Intelligence*, 4131–4138.

Teodorovic, D., & Dell'Orco, M. (2005). Bee colony optimization–a cooperative learning approach to complex transportation problems. *Advanced OR and AI methods in transportation*, *51*, 60.

Thomas, M., & Joy, A. T. (2006). *Elements of information theory*. Wiley-Interscience.

Thompson, D. W., & Thompson, D. W. (1942). *On growth and form* (Vol. 2). Cambridge university press Cambridge.

Thompson, F., & Guihen, D. (2019). Review of mission planning for autonomous marine vehicle fleets. *Journal of Field Robotics*, *36*(2), 333–354.

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., & Tassa, Y. (2020). Dm_control: Software and tasks for continuous control. *Software Impacts*, *6*, 100022.

Turing, A. M. (1990). The chemical basis of morphogenesis. *Bulletin of mathematical biology*, *52*(1), 153–197.

Twu, P., Mostofi, Y., & Egerstedt, M. (2014). A measure of heterogeneity in multi-agent systems. *2014 American Control Conference*, 3972–3977.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. *Proceedings of the AAAI conference on artificial intelligence*, *30*(1).

Vanam, L. (2017). Information theory: Statistics for deep learning. https://medium.com/machine-learning-bootcamp/demystifying-information-theory-e21f3af09455

Vasilijevic, A., Nad, D., Mandic, F., Miskovic, N., & Vukic, Z. (2017). Coordinated navigation of surface and underwater marine robotic vehicles for ocean sampling and environmental monitoring. *IEEE/ASME transactions on mechatronics*, *22*(3), 1174–1184.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354.

Von Neumann, J., & Morgenstern, O. (2007). *Theory of games and economic behavior*. Princeton university press.

von Stengel, B., & Koller, D. (1997). Team-maxmin equilibria. *Games and Economic Behavior*, *21*(1-2), 309–321.

Wagner, G., & Choset, H. (2015). Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, *219*, 1–24.

Wallkötter, S. (2019). The hidden potential of nao and pepper – custom robot postures in naoqi v2.4. https://sebastianwallkoetter.wordpress.com/

2019/04/05/the-hidden-potential-of-nao-and-pepper-custom-robot-postures-in-naoqi-v2-4/

Wang, H., & Yeung, D.-Y. (2020). A survey on bayesian deep learning. *ACM Computing Surveys (CSUR)*, *53*(5), 1–37.

Wang, J., Jing, X., Yan, Z., Fu, Y., Pedrycz, W., & Yang, L. T. (2020). A survey on trust evaluation based on machine learning. *ACM Computing Surveys (CSUR)*, *53*(5), 1–36.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. *International conference on machine learning*, 1995–2003.

Wehrl, A. (1978). General properties of entropy. *Reviews of Modern Physics*, *50*(2), 221.

Weiss, G. (1999). *Multiagent systems: A modern approach to distributed artificial intelligence*. MIT press.

Weyns, D., Boucké, N., Holvoet, T., & Demarsin, B. (2007). Dyncnet: A protocol for dynamic task assignment in multiagent systems. *First International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2007)*, 281–284.

Wiegand, R. P. (2004). *An analysis of cooperative coevolutionary algorithms*. George Mason University.

Wilson, S., Glotfelter, P., Wang, L., Mayya, S., Notomista, G., Mote, M., & Egerstedt, M. (2020). The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems. *IEEE Control Systems Magazine*, *40*(1), 26–44.

Winfield, A. F., Sa, J., Fernández-Gago, M.-C., Dixon, C., & Fisher, M. (2005). On formal specification of emergent behaviours in swarm robotic systems. *International journal of advanced robotic systems*, *2*(4), 39.

Wolpert, D. H., & Tumer, K. (2002). Optimal payoff functions for members of collectives. *Modeling complexity in economic and social systems* (pp. 355–369). World Scientific.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John wiley & sons.

Wu, F., Varadharajan, V. S., & Beltrame, G. (2019). Collision-aware task assignment for multi-robot systems. *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 30–36.

Yan, Z., Jouandeau, N., & Cherif, A. A. (2013). A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, *10*(12), 399.

Yang, G.-Z., Bellingham, J., Dupont, P. E., Fischer, P., Floridi, L., Full, R., Jacobstein, N., Kumar, V., McNutt, M., Merrifield, R., et al. (2018). The grand challenges of science robotics. *Science robotics*, *3*(14), eaar7650.

Yang, Q. (2021). Self-adaptive swarm system (sass) [Doctoral Consortium]. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 5040–5041. https://doi.org/10.24963/ijcai.2021/722

Yang, Q., Luo, Z., Song, W., & Parasuraman, R. (2019). Self-reactive planning of multi-robots with dynamic task assignments. *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 89–91.

Yang, Q., & Parasuraman, R. (2020a). A game-theoretic utility network for cooperative multi-agent decisions in adversarial environments. *arXiv preprint arXiv:2004.10950*.

Yang, Q., & Parasuraman, R. (2020b). Hierarchical needs based self-adaptive framework for cooperative multi-robot system. *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2991–2998.

Yang, Q., & Parasuraman, R. (2020c). Needs-driven heterogeneous multi-robot cooperation in rescue missions. *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 252–259.

Yang, Q., & Parasuraman, R. (2021). How can robots trust each other for better cooperation? a relative needs entropy based robot-robot trust assessment model. *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2656–2663.

Yates, F. E. (2012). *Self-organizing systems: The emergence of order*. Springer Science & Business Media.

Yehoshua, R., & Agmon, N. (2015). Adversarial modeling in the robotic coverage problem. *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 891–899.

Yoshida, E., Arai, T., Ota, J., & Miki, T. (1994). Effect of grouping in local communication system of multiple mobile robots. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, *2*, 808–815.

Yu, H., Shen, Z., Leung, C., Miao, C., & Lesser, V. R. (2013). A survey of multi-agent trust management systems. *IEEE Access*, *1*, 35–50.

Yu, L., Song, J., & Ermon, S. (2019). Multi-agent adversarial inverse reinforcement learning. *Proceedings of the 36th International Conference on Machine Learning*.

Zeng, Y., Doshi, P., & Chen, Q. (2007). Approximate solutions of interactive dynamic influence diagrams using model clustering. *Proceedings of the National Conference on Artificial Intelligence*, *22*(1), 782.

Zhang, Z., Ong, Y.-S., Wang, D., & Xue, B. (2019). A collaborative multiagent reinforcement learning method based on policy gradient potential. *IEEE transactions on cybernetics*.

Zhao, W., Queralta, J. P., & Westerlund, T. (2020). Sim-to-real transfer in deep reinforcement learning for robotics: A survey. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 737–744.

Zheng, H., Panerati, J., Beltrame, G., & Prorok, A. (2019). An adversarial approach to private flocking in mobile robot teams. *arXiv preprint arXiv:1909.10387*.

Ziebart, B. D. (2010). *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.

Zuo, S., & Yue, D. (2020). Resilient output formation containment of heterogeneous multigroup systems against unbounded attacks. *IEEE Transactions on Cybernetics*.

Appendix A

Game-theoretic Utility Tree (GUT) Experiment Details

Considering cross-platform, scalability, and efficiency of the simulations, we chose the "Unity" game engine to simulate the *Explorers and Aliens Game* and selected Gambit McKelvey et al., 2006 toolkit[45] for calculating each level's Nash Equilibrium.

Each interaction (trial) in the *Explorers and Aliens Game* lasts about 40 to 50 minutes in the simulated experiments on a laptop with Intel i7 Processor, GeForce GTX 1050 Ti GPU, and 16GB DDR4 RAM running on Ubuntu 18.04.

In our experiments, we assume that explorers and aliens have the same moving speed, and aliens can not share information. It is worth noting that an alien attacking capability is three times that of an explorer in all the experiments, which means that aliens represent higher capabilities (who act based on their self-interests) in preventing explorers from their tasks.

A.0.1 Utility Values Computation

We implement the agent needs hierarchy through the following levels and parameters:

- The teaming needs are represented using the team *winning* expectation, which is used in the Level 1 payoff matrix of the GUT (Table 6.1) and implemented using the calculations in Section A.0.1 ;

[45] **Gambit** is an open-source collection of tools for doing computation in game theory which can build, analyze, and explore game models

- The basic needs are represented using the (battery) *Energy* level metric, which is used in the Level 2 payoff matrix of the GUT (Table 6.2) and implemented using the calculations in Section A.0.1 ;
- The safety needs are represented using the Health Points (*HP*) status metric, which is used in the Level 3 payoff matrix of the GUT (Table 6.3) and implemented using the calculations in Section A.0.1.

The capability needs, and the learning needs as per Section 3.2 are not implemented in this work.

Below, we list the terms and notations to be used in the below utility equations.

- AT and BT represent the action space of group A and B correspondingly;
- n and m represent the number of Explorers and Aliens respectively;
- d represents the group average distance between two opponents;
- v represents the agent's velocity;
- i and j represent the times of attacks and being attacked;
- w represents Explorers' communication times;
- f and q represent the unit attacking energy cost of both sides agents respectively;
- t_{ev} and t_{mv} represent average attacking ability levels of both sides respectively;
- r_{ev} and r_{mv} represent average defending ability levels of both sides respectively;
- t_e and t_m represent specific agent's attacking ability levels of both sides respectively;
- r_e and r_m represent specific agent's defending ability levels of both sides respectively;
- ϕ_e and ϕ_m represent individual agent's size;
- k represents the number of Explorers' attacking simultaneously;
- g represents the number of Aliens' attacking simultaneously;

- a, b, c, ρ, $\gamma_{e/m}$ and $\delta_{e/m}$ represent corresponding coefficient;
- e_e and e_m represent the current energy level of Explorer and Alien;
- h represents the current HP level of agent;
- p represents the probability values of a specific variable.

Winning Utility Expectation

We consider the *Winning Utility* following *Bernoulli Distribution* to represent individual high-level expected utility (teaming & cooperation needs) in the first level (Eq. (A.1)). The values of the coefficient $a_{1...5}$ are based on different situations in the game.

$$W(t_{ev}, t_{mv}, r_{ev}, r_{mv}, n, m) = (a_1 \frac{a_2 t_{ev} + a_3 r_{ev}}{a_4 t_{mv} + a_5 r_{mv}})^{\frac{m}{n}}; \tag{A.1}$$

Energy Utility Expectation

The second level's utility can be described as the relative *Expected Energy Cost* Eq. (A.2), which consists of three parts of energy costs: $moving$, $attacking$, and $communication$. And we assume that the energy of $moving$, $attacking$, and $communication$ are proportional to the distance between agents Eq. (A.3), times of attacks and being attacked Eq. (A.4), and the communication times Eq. (A.5), respectively.

$$\begin{aligned}
&E(d, v, f, q, n, m, \phi_e, \phi_m) = b_0 + \\
&b_1 \int_{-\infty}^{+\infty} (n - m) e_d(x) p_d(x, d) \mathrm{d}x \\
&+ b_2 (\sum_{i=1}^{+\infty} n e_{a_e}(i, f) p_{a_m}(j, m\phi_m) - \\
&\sum_{j=1}^{+\infty} m e_{a_m}(j, q) p_{a_e}(i, n\phi_e)) \\
&+ b_3 \sum_{w=1}^{+\infty} n e_c(w) p_c(w, \frac{d}{v});
\end{aligned} \tag{A.2}$$

$$e_d(x) = b_{11} x; \tag{A.3}$$

$$e_a(x, y) = b_{12} x y; \tag{A.4}$$

$$e_c(x) = b_{13} x \tag{A.5}$$

In the attack-defend process, we also assume that individual action distance (Eq. (A.6)), times of attacks (Eq. (A.7)) and being attacked (Eq. (A.8)), and the communication times (Eq. (A.9)) follow *Normal Distribution*, *Poisson Distribution* correspondingly.

$$p_d(x, d) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-d)^2}{2}}; \tag{A.6}$$

$$p_{a_e}(x, \lambda_e) = \frac{e^{-\lambda_e} \lambda_e^x}{x!}; \tag{A.7}$$

$$p_{a_m}(y, \lambda_m) = \frac{e^{-\lambda_m} \lambda_m^y}{y!}; \tag{A.8}$$

$$p_{a_m}(x, \frac{d}{v}) = \frac{e^{-\frac{d}{v}} (\frac{d}{v})^x}{x!}; \tag{A.9}$$

Then, we can simplify the Eq. (A.2) as Eq. (A.10).

$$\begin{aligned} &E(d, v, f, q, n, m, \phi_e, \phi_m) = \\ &b_0 + b_1 b_{11}(n - m)d + \\ &b_2 b_{12} nm(f\phi_m - q\phi_e) + \\ &b_3 b_{13} n\frac{d}{v}; \end{aligned} \tag{A.10}$$

HP Utility Expectation

At the lowest level, we use the expected HP cost to describe the expected utility Eq. (A.11), which is the relative HP cost between explorer and alien. And we assume that the unit HP cost is associated with times of being attacked and individual unit attacking and resistant level Eq. (A.12). Also, individual unit attacking Eq. (A.13) and resistant Eq. (A.14)) level is proportional to the agent's current energy level, correspondingly.

$$\begin{aligned} &H(k, t_e, t_m, r_e, r_m, g, \phi_e, \phi_m) = \\ &c_0 + c_1(\sum_{i=1}^{+\infty} kh(t_e, r_e, i) p_{h_m}(i, \phi_m) - \\ &\sum_{j=1}^{+\infty} gh(t_m, r_m, j) p_{h_e}(j, \phi_e)); \end{aligned} \tag{A.11}$$

$$h(x, y, z) = \rho z(x + y) \tag{A.12}$$

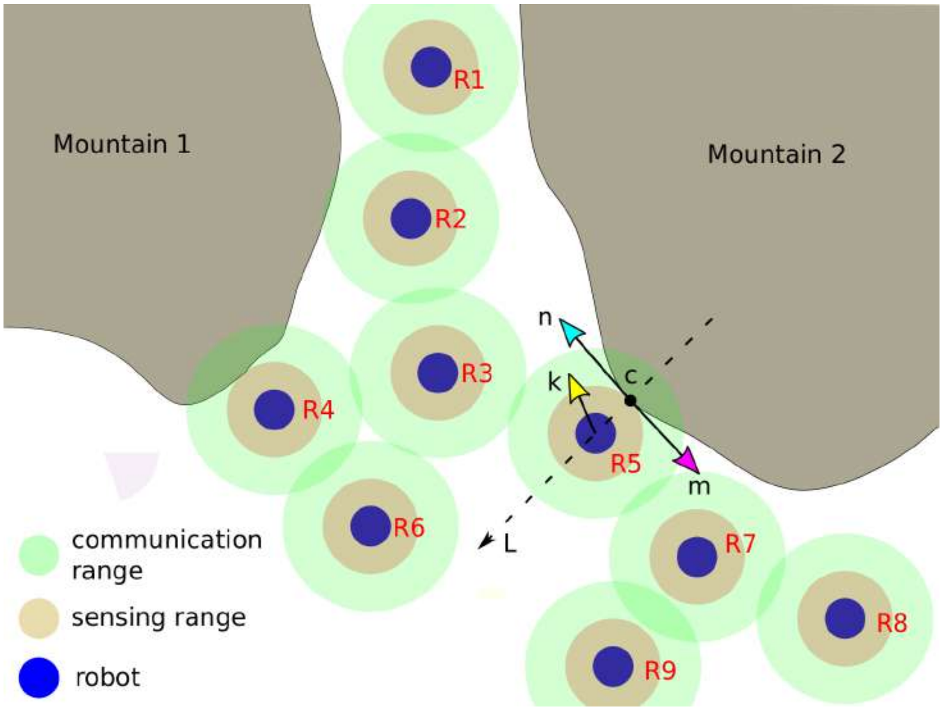


Figure A.1: Illustration of "Adapt The Edge" algorithm for tackling (unintentional) obstacles

$$t_{e/m}(e_{e/m}) = \gamma_{e/m} e_{e/m}; \tag{A.13}$$

$$r_{e/m}(e_{e/m}) = \delta_{e/m} e_{e/m}; \tag{A.14}$$

Here, we assume that $p_{h_e}(i, \phi_e)$ and $p_{h_m}(j, \phi_m)$ also follow *Poisson Distribution*. Then, through simplifying the Eq. (A.11), we finally get Eq. (A.15).

$$\begin{aligned} &H(k, t_e, t_m, r_e, r_m, g, \phi_e, \phi_m) = \\ &c_0 + c_1 \rho [k \phi_m e_e (\gamma_e + \delta_e) - \\ &g \phi_e e_m (\gamma_m + \delta_m)]; \end{aligned} \tag{A.15}$$

A.0.2 Obstacle Avoidance Algorithm

When explorers perceive the obstacles (which are unintentional adversaries), they utilize limited information by sharing the perceiving information among agents for cooperative collision avoidance. In the current implementation, robots can $R5$ detect the obstacles using their onboard proximity sensors.

An example scenario with nine robots forming an ally agent team and two mountains representing obstacles on the way to a treasure is depicted in Fig. A.1. To avoid a collision, the robot team needs to switch the moving direction k based on the tangent's direction of the nearest collision point c. There are two directions n and m. According to the current status, $R5$ will select the direction n, which has potentially more non-collision robots.

Specifically, L is a straight line passing through the tangent point c and perpendicular to n and m. There are four robots $R1$, $R2$, $R3$, and $R6$ without collision in the direction n comparing with three non-collision robots $R7$, $R8$, and $R9$ in the direction m in current situation. So $R5$ will move Δd in direction n, then adjust the direction to the goal point moving forward until there is no unintentional adversaries in its route through iterating the process. Therefore, the ally robot team adapts the edge of their formation shape dynamically to avoid obstacles. Alg. 11 presents the decision process of the proposed obstacle avoidance strategy.

Algorithm 11: Obstacle avoidance algorithm to adapting the edge of the team formation shape.

```
Input: Explorers' and Obstacles' states
Output: moving direction $r$ and distance $\Delta d$
while The nearest collision point $c\,!= Null$ do
    calculate the number $n$ and $m$ of non-collision agents in both side
      of the line $l$ passing through $c$ and perpendicular $c$'s tangent;
    if $n > m$ then
        $r$ = $n$ side in line $l$;
        $\Delta d$ = one step of agent's movement;
    else if $n == m$ then
        agent stop;
    else if $n < m$ then
        $r$ = $m$ side in line $l$;
        $\Delta d$ = one step of agent's movement;
return $r$ = current position to goal point, r, $\Delta d$
```

Appendix B

Applying GUT on the Pursuit Domain

To demonstrate the generality of the GUT in multiagent systems in different domains, we choose the *Pursuit* domain Chung et al., 2011 implementation and extend our experimental analysis. The Pursuit-Evasion is a family of problems that have been extensively studied using a wide variety of approaches and has many instantiations that can be used to illustrate different multiagent system scenarios Stone and Veloso, 2000. Assuming there are k-number of pursuers and m-number of evaders, the pursuit problem is ensuring the eventual capture of all the evaders by the pursuers efficiently.

Recently, Makkapati and Tsiotras, 2019 studied this pursuit problem by considering two different strategies for the pursuers; they either follow a constant bearing (CB) or a pure pursuit (PP) strategy to capture the evader. Each evader is assigned to a set of pursuers based on the instantaneous positions of all the players. More specifically, in the case of a CB strategy, the bearing angle between a pursuer and the evader remains constant until the time of capture. But in the PP strategy, the velocity vector of the pursuer is aligned along the line of sight. From the cooperative capture angle, Kothari et al., 2014 discussed an approximate solution that the pursuers minimize the safe-reachable area (the set of points where an evader can travel without being caught) of the evader.

B.0.1 Experiment Setup

We implement this problem in the Robotarium platform Pickem et al., 2017, where the pursuers' strategy is computed using GUT and compare its performance with the standard state-of-the-art pursuit strategies: Constant Bearing and Pure Pursuit. Our experiments design a pursuit-evasion game with three pursuers and one evader scenario shown in Fig. B.2(a). We implement three

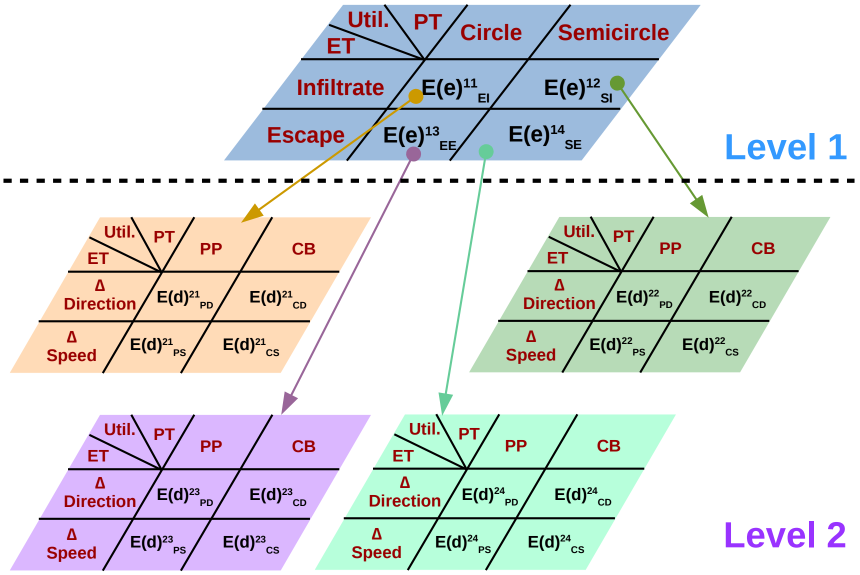


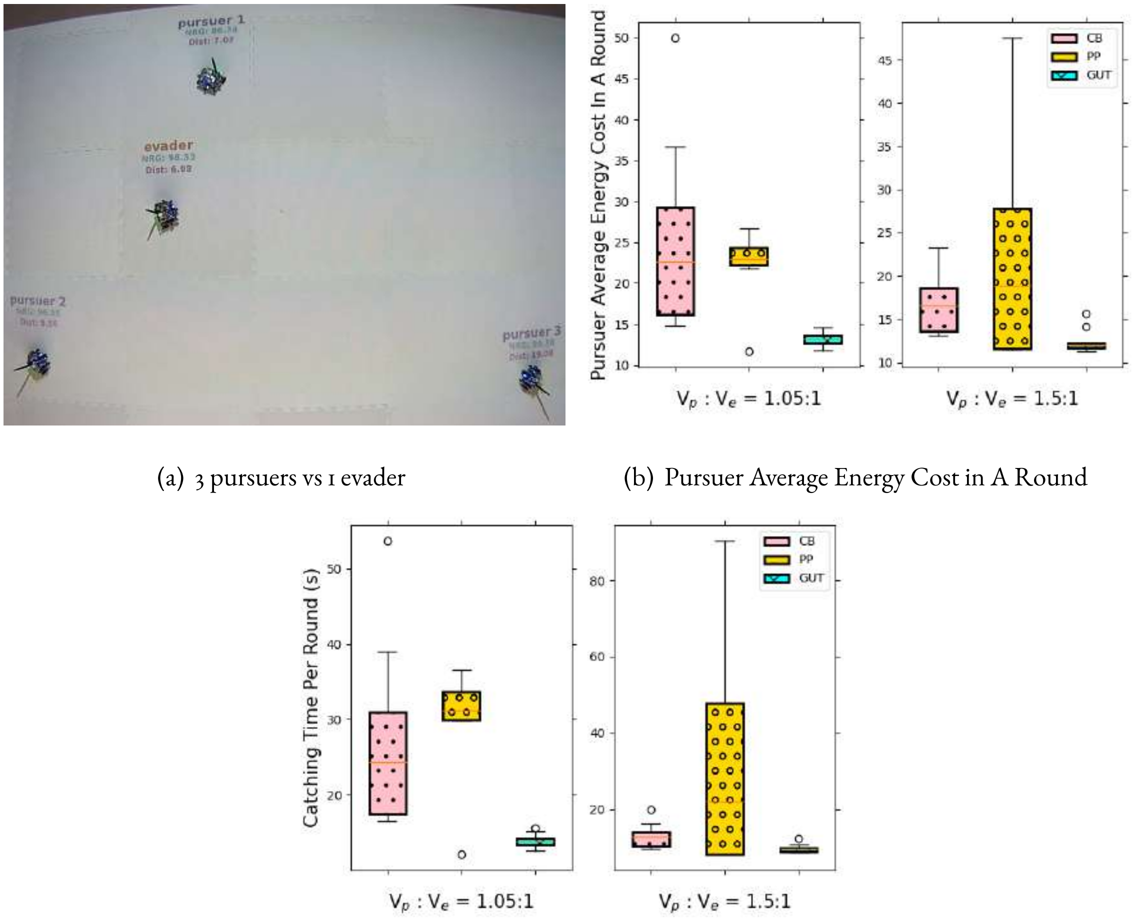
Figure B.1: Illustration of two level game-theoretic utility tree (GUT) for multi-robot cooperative pursuit

pursuer's approaches to capture the evader: *constant bearing (CB)*, *pure pursuit (PP)*, and the *GUT*. [46]

For the GUT, we built a structure of the two-level pursuer and evader tactics payoff matrices represented in the Tables B.1 and B.2. They calculate the utilities based on the expected energy E_{xx} of the pursuers to capture the evader (Level 1) and the distance D_{xx} between the pursuer and the evader (Level 2). Figure B.1 illustrates the two-level GUT for multi-robot cooperative pursuit in the *Pursuit-Evasion* game.

In Level 1, the pursuer team will decide which tactic to choose to capture the evader and searches the area correspondingly. For this level, we design two kinds of strategies for the team of pursuers – *Circle* and *Semicircle* – for achieving the cooperative capture of evaders based on the study in Kothari et al., 2014. Here, the *Circle* tactic is that pursuers center on the evader and surround it gradually until the evader can not move in any direction. In the *Semicircle*, the pursuers encompass the evader in a semicircle, forcing it to a dead end. On the other hand, the evader can select the strategies – *Infiltrate* and *Escape* – based on the current situation correspondingly.

[46] The video demonstration of the experiments showing sample trials of experiments implementing GUT in *Pursuit* domain is available at the link https://youtu.be/KBZCZbuPKso.



(a) 3 pursuers vs 1 evader

(b) Pursuer Average Energy Cost in A Round

(c) Time To Catch Evader Per Round

Figure B.2: The Performance of Robotarium Experiments with Different Speed Proportion in Pursuit-Evasion Domain

In Level 2, we design the specific approaches to execute level one's decision by choosing how to approach the evader. Here, we directly implement the methods of *CB* and *PP* Makkapati and Tsiotras, 2019 as the pursuers' tactics. The evader can execute either the *Changing Direction* (Δ Direction) or *Changing Speed* (Δ Speed) adapting to the current scenario.

In this pursuit-evasion game, we assumed that the energy utility expectation in the Level-1 tactics payoff matrix follow the Eq. (B.1).

$$E(e_{cir/sem}, \alpha_{\Delta inf/\Delta esc}) = \alpha_{\Delta inf/\Delta esc}.d_{cir/sem} \tag{B.1}$$

Table B.1: Level 1 pursuit-evader tactics payoff matrix

Utility PT ET	Circle	Semicircle
Infiltrate	E_{EI}	E_{SI}
Escape	E_{EE}	E_{SE}

Table B.2: Level 2 pursuit-evader tactics payoff matrix

Utility PT ET	PP	CB
ΔDirection	D_{PD}	D_{CD}
ΔSpeed	D_{PS}	D_{CS}

Here, $d_{cir/sem}$ are the distances to the goal point (position of the evader) using circle or semicircle methods. $\alpha_{\Delta inf/\Delta esc}$ are the coefficients of evader with the strategies of infiltrating or escaping, respectively.

We introduce the expected distance (Eq. (B.2)) between the pursuer's current position and the evader to present the Level-2 tactics payoff matrix expected utility .

$$E(d_{cb/pp}, \beta_{\Delta dir/\Delta spd}) = \beta_{\Delta dir/\Delta spd} \cdot d_{cb/pp} \tag{B.2}$$

Here, $d_{cb/pp}$ represents the current distance between pursuer and capture position, executing the CB or the PP tactic. $\beta_{\Delta dir/\Delta spd}$ describe the coefficients of evader implementing changing direction or speed strategy following normal distributions with different expectations correspondingly.

Our experiments assume that the pursuers will capture the evader if all the distances between each pursuer and evader are less than a threshold. We apply the three approaches (CB, PP, and GUT) in the Robotarium with real robots and analyze the results of ten simulation trials in the Robotarium simulator.

Further, we consider two different speed ratios between the pursuer and the evader: 1) $V_p = 1.05 * V_e$ meaning that the pursuers can travel 5% higher speed than the evader; 2) $V_p = 1.5 * V_e$ meaning that that pursuer has 50% higher speed than the evader. Note, the speed of the pursuer is always assumed to be higher than the evader to guarantee an eventual capture using standard pursuit approaches Kothari et al., 2014. Other experiment settings are similar to the *Explore* domain experiments in Robotarium described in Section 6.6.

We consider two performance metrics: the pursuer's energy costs and the efficiency (time to capture). An ideal approach is the one that results in the least energy cost and least time.

B.0.2 Results and Discussion

Fig. B.2(b) and B.2(c) describe the pursuer average energy cost per round and capture time for two different speed proportions, respectively. As we can see, if the pursuer and evader velocities are very close ($V_p : V_e$ = 1.05:1), the GUT has a more distinct performance than the larger difference in the velocity ($V_p : V_e$ = 1.5:1) compared with the standard CB and the PP strategies. It means that if there is a considerable advantage in capabilities (like the speed) between individuals and opponents, there would not be a significant difference in the performance of cooperative strategy (GUT) and independent strategies (CB or PP). That is, when every group member has outstanding abilities compared to their adversaries, the tasks are less challenging, and the cooperation becomes less significant in the real world. However, when the agents have fewer advantages against their adversaries, cooperation between group members becomes critical to ensure a successful mission.

B.0.3 Conclusion

In the *Pursuit* Domain experiments, we demonstrate the generality of the GUT-based decision-making in multiagent systems under adversarial scenarios. GUT helps organize group behaviors to present more complex group strategies. The GUT can enable the multiagent system to adapt to various situations and improve overall system performance and achieve global team mission successfully. Furthermore, we prove that effective cooperation strategies can help agents with fewer advantages achieve higher performance in the tasks, which effectively organize the system resource and fully take advantage of the capabilities of each group member.

Appendix C

Bayesian SAC (BSAC) Proofs and Experiment Details

C.1 Proofs

We follow the original SAC approach Haarnoja et al., 2018 for proving the policy evaluation, improvement, and training in the BSAC.

Lemma 1 (Bayesian Soft Policy Evaluation)**.** *Consider the soft Bellman backup operator $\mathcal{T}^{\pi}$ in the Eq.* (7.6) *and define $Q^{k+1} = \mathcal{T}^{\pi}Q^{k}$. The sequence Q^{k} will converge to the soft Q-value of the joint policy π for each sub-policy π_i as $k \to \infty$, $i \in m$.*

Proof. Due to the additivity of the entropy Wehrl, 1978, we can rewrite the entropy augmented rewards and the update rule as Eq. (C.1) and (C.2), respectively.

$$r_{\pi}(s_t, \mathcal{A}_t) \triangleq r(s_t, \mathcal{A}_t) + \mathbb{E}_{s_{t+1} \sim p}\left[\frac{1}{m}\sum_{i=1}^{m}\mathcal{H}(\pi_i(\cdot|s_{t+1}))\right] \tag{C.1}$$

$$\begin{aligned} Q(s_t, \mathcal{A}_t) \leftarrow r_{\pi}(s_t, \mathcal{A}_t) + \\ \frac{\gamma}{m} \cdot \sum_{i=1}^{m} \mathbb{E}_{s_{t+1} \sim p, a_{i_{t+1}} \sim \pi_i}\left[Q(s_{t+1}, a_{i_{t+1}})\right] \end{aligned} \tag{C.2}$$

Then, through the policy evaluation of the standard convergence results Sutton and Barto, 2018, the assumption $|\mathcal{A}| < \infty$ guarantees the BSAC entropy augmented reward to be bounded. Here, we assume that each sub-policy π_i has the same weight in the joint policy π. □

Lemma 2 (Bayesian Soft Policy Improvement). *Let the joint policy π_{new} optimize the minimization problem defined in Eq.* (7.8) *and* $\pi_{old} \in \Pi$, *then* $Q^{\pi_{new}}(s_t, \mathcal{A}_t) \geq Q^{\pi_{old}}(s_t, \mathcal{A}_t)$ *for all* $(s_t, \mathcal{A}_t) \in \mathcal{S} \times \mathcal{A}$ *with* $|\mathcal{A}| < \infty$.

Proof. Supposing the joint policy is $\pi_{old} \in \Pi$, its soft state joint action value and soft state value are $Q^{\pi_{old}}$ and $V^{\pi_{old}}$, respectively. By defining π_{new} as Eq. (7.8) and (C.3), we can deduct the Eq. (C.4).

$$J_{\pi_{old}}(\pi_{new}(\cdot|s_t)) \leq J_{\pi_{old}}(\pi_{old}(\cdot|s_t)) \tag{C.3}$$

$$\begin{aligned} \mathbb{E}_{\mathcal{A}_t \sim \pi_{new}} & \left[Q^{\pi_{old}}(s_t, \mathcal{A}_t) - \frac{1}{m} \sum_{i=1}^{m} \log \pi_{i_{new}}(a_{i_t}|s_t)) \right] \\ & \geq V^{\pi_{old}}(s_t). \end{aligned} \tag{C.4}$$

Then, through applying the soft Bellman equation and the bound in Eq. (C.4), we get Eq. (C.5). Finally, it will converge to $Q^{\pi_{new}}$.

$$\begin{aligned} Q^{\pi_{old}}(s_t, \mathcal{A}_t) &= r_\pi(s_t, \mathcal{A}_t) + \gamma \cdot \mathbb{E}_{s_{t+1} \sim p} \left[V^{\pi_{old}}(s_{t+1}) \right] \\ &\leq r_\pi(s_t, \mathcal{A}_t) + \gamma \cdot \mathbb{E}_{s_{t+1} \sim p} [\mathbb{E}_{a_{i_{t+1}} \sim \pi_{i_{new}}} \\ &[Q^{\pi_{old}}(s_t, \mathcal{A}_t) - \frac{1}{m} \sum_{i=1}^{m} \log \pi_{i_{new}}(a_{i_t}|s_t))]] \\ &\vdots \\ &\leq Q^{\pi_{new}}(s_t, \mathcal{A}_t). \end{aligned} \tag{C.5}$$

□

Theorem 1 (Bayesian Soft Policy Iteration). *Iterating the Bayesian soft policy evaluation and Bayesian soft policy improvement from any joint policy* $\pi \in \Pi$ *converges to a joint policy* π^* *for* $Q^{\pi^*}(s_t, \mathcal{A}_t) \geq Q^{\pi}(s_t, \mathcal{A}_t)$ *and* $(s_t, \mathcal{A}_t) \in \mathcal{S} \times \mathcal{A}$, *assuming* $|\mathcal{A}| < \infty$.

Proof. According to the Lemma 2, the sequence of Q^{π_k} is monotonically increasing and converges to the specific π^*. Here, k is the iteration less than infinity. Furthermore, considering for all the joint policy $\pi \in \Pi, \pi \neq \pi^*$ at convergence having Eq. (C.6), we can get a soft value $Q^*(s_t, a_t)$ which is larger than the other soft value of any joint policy in Π. Hence the joint policy π^* is optimal in Π.

$$J_{\pi^*}(\pi^*(\cdot|s_t)) < J_{\pi^*}(\pi(\cdot|s_t)) \tag{C.6}$$

□

C.2 Experiment details

Tables C.1, C.2, C.3 and C.4 provide the technical details and hyperparameters used to train the agents in our experiments.

Table C.1: BSAC and SAC Hyperparameters.

Parameter	Value
optimizer	Adam Kingma and Ba, 2014
learning rate	3×10^{-4}
discount (γ)	0.99
replay buffer size	10^6
number of hidden layers in all networks	2
number of hidden units per layer	256
number of sample per mini-batch	256
nonlinearity	ReLU
target smoothing coefficient (τ)	0.005
target update interval	1
gradient steps	1

Table C.2: DDPG Hyperparameters.

Parameter	Value
optimizer	Adam Kingma and Ba, 2014
learning rate	2.5×10^{-4}
discount (γ)	0.99
replay buffer size	10^6
number of hidden layers in all networks	2
number of hidden units per layer	256
batch size	64
nonlinearity	ReLU

Table C.3: PPO Hyperparameters.

Parameter	Value
optimizer	Adam Kingma and Ba, 2014
learning rate	3×10^{-4}
discount (γ)	0.99
number of hidden layers in all networks	2
number of hidden units per layer	256
batch size	64
nonlinearity	ReLU

Table C.4: Environment-specific parameters.

Environment	Action Dimensions	Reward Scale
Hopper-v2	3	5
Walker2d-v2	6	5
Humanoid-v2	17	20