Dynamic Resource Allocation in Systems-of-Systems Using a Heuristic-Based Interpretable Deep Reinforcement Learning

Qiliang Chen
Department of Mechanical and Industrial Engineering,
Multi-AGent Intelligent Complex Systems (MAGICS) Lab,
Northeastern University,
Boston, MA 02115
email: chen.qil@northeastern.edu

Babak Heydari[1]
Department of Mechanical and Industrial Engineering,
Multi-AGent Intelligent Complex Systems (MAGICS) Lab,
Institute of Experiential AI,
Northeastern University,
Boston, MA 02115
e-mail: b.heydari@northeastern.edu

Systems-of-systems (SoS) often include multiple agents that interact in both cooperative and competitive modes. Moreover, they involve multiple resources, including energy, information, and bandwidth. If these resources are limited, agents need to decide how to share resources cooperatively to reach the system-level goal, while performing the tasks assigned to them autonomously. This paper takes a step toward addressing these challenges by proposing a dynamic two-tier learning framework, based on deep reinforcement learning that enables dynamic resource allocation while acknowledging the autonomy of systems constituents. The two-tier learning framework that decouples the learning process of the SoS constituents from that of the resource manager ensures that the autonomy and learning of the SoS constituents are not compromised as a result of interventions executed by the resource manager. We apply the proposed two-tier learning framework on a customized OpenAI Gym environment and compare the results of the proposed framework to baseline methods of resource allocation to show the superior performance of the two-tier learning scheme across a different set of SoS key parameters. We then use the results of this experiment and apply our heuristic inference method to interpret the decisions of the resource manager for a range of environment and agent parameters. [DOI: 10.1115/1.4055057]

Keywords: artificial intelligence, machine learning, systems design, systems engineering, reinforcement learning, interpretable AI, resource allocation

1 Introduction

Systems-of-systems (SoS) often consist of multiple autonomous systems interacting at various levels of coordination, cooperation, and competition [1–3]. Agents in heterogeneous, dynamic environments make resource demand uncertain and dynamic, both over time and across agents [4,5]. As a result, one of the main functions of agent-to-agent interactions in an SoS is to facilitate dynamic resource allocation, either by sharing or competing for resources that can mean a variety of things according to the applications, ranging from information and energy to bandwidth, and processing power. This hybrid, *coopetitive* interactions among autonomous systems pose a fundamental challenge to the operation and governance of complex multi-agent systems (MAS), in general, and SoS, in particular. This is because the autonomous nature of these agents drives them to compete for resources in an inefficient manner. Furthermore, since individual systems cannot (and should not) be forced to efficiently share resources by a central planner due to their autonomy [6], making them different from many engineering systems with nonautonomous constituents, requiring fresh look at the command and control for these systems [7].

One way to address this challenge is to close the gap between the system-level goals and those of the individual systems, i.e., by creating system-level incentives that promote resource sharing that is beneficial to the entire SoS. Manipulating incentives have traditionally been realized in a direct way by engineering the pay-offs of strategic interactions between different agents, using implementation theory or mechanism design [8–10]. Complementary to this approach, it has been suggested that incentives for resource sharing can be steered indirectly by altering the structure of interactive network between different agents [11–13]. These schemes, although sufficient for some applications, have two problems: First, they are static in nature and not adaptable to the changes in the environment. Moreover, they do not take into account the long-term learning of agents in response to these manipulated incentives that would result in a shift in the behavior of autonomous agents [14,15]. Long-term learning means agents can adapt to the changes in the environment, especially when other agents are also learning at the same time. In other words, agents can always adapt and learn automatically for a long time or even during their life time. This long-term learning is especially important in the light of recent developments in artificial intelligence (AI), which calls for a more adaptable resource allocation scheme for systems-of-systems, by leveraging recent AI methods, in particular, reinforcement learning (RL).

Reinforcement learning has achieved significant success in several areas, including robots [16,17], Go [18], video games [19], and design and governance of engineering systems [20,21,4]. Moreover, using RL in MAS to induce cooperative behavior has attracted researchers' interests in recent years [22–27]. Most of those researches, however, assume that the environments are resource-unlimited and focus on how to design new learning frameworks or efficient communication protocols to improve the performance in MAS. However, the limited resource assumption is critical for many applications and can drastically change the optimal policies suggested by the RL agents. For example, in telecommunication system, bandwidth are limited, which means agents cannot send every bits they want in a crowded communication channel [28]; in multidrone system, drones have finite vision range, which means they cannot always observe the position of target in a rescue task [29]. Under these circumstances, using current MARL methods may lead to suboptimal or even detrimental policies.

[1]Corresponding author.

Contributed by the Design Automation Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received February 8, 2022; final manuscript received July 11, 2022; published online August 8, 2022. Assoc. Editor: Jitesh H. Panchal.

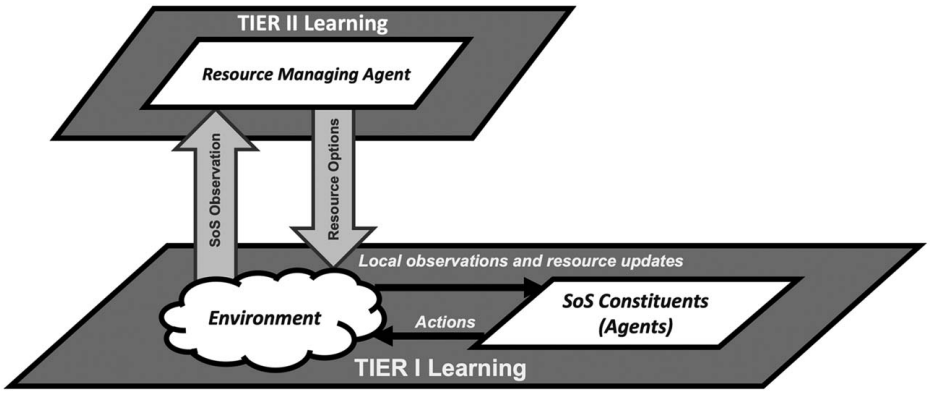

Fig. 1 High-level representation of the proposed two-tier learning framework for dynamic resource allocation in system-of-systems

This paper takes a step toward addressing these challenges by proposing a dynamic two-tier learning framework, as briefly described in Fig. 1 (see the methods section for more details about the framework), based on deep reinforcement learning that enables dynamic resource allocation, while acknowledging the autonomy of systems' constituents. The two-tier learning framework that decouples the learning process of the SoS constituents from the learning of an omniscient—but not omnipotent—resource manager ensures that the autonomy and learning of the SoS constituents are not compromised as a result of the resource manager's interventions. The agents in the first tier (SoS constituents) use deep deterministic policy gradient (DDPG) [30] to learn the basic skills like navigation and then fix their parameters. In specific situations, the resource manager can decide whether or not to assign additional resources (for a cost) to agents and which resources to assign at any point of time, and the SoS agents' behaviors will be shaped by the resources they get in conjunction to their pretrained skills, acquired in tier I. We use deep Q-learning [19] to train the manager to learn the efficient strategy of dynamic allocation of multiple costly resources. To create an RL solution, the environment is first modeled as a partial observable Markov decision process (POMDP) [31], which is a stochastic sequential decision process in which agents can only partially observe the environment state, make a decision in each time-step, and receive a reward, based on the state transition of the environment. The changes in the environment are modeled using transition probabilities, which are embedded into the model of the environment, but these transition probabilities and their associated rewards are generally unknown to the learning agents. By exploring the environment, RL agents *learn* optimal policies, i.e., how to take action based on their state observations, in order to maximize their expected total rewards. DDPG in first tier learning is an actor–critic RL framework [32], in which the actor takes actions based on its observations, while the critic *evaluates* the value of each observation–action pair. To make the learning scheme more realistic to decentralized systems with autonomous agents, our approach for TIER I learning is based on decentralized-training–decentralized-execution (DTDE), which means the actor and critic for each agent can only use the local observation in both training and decision making states. The resulting learning algorithms from this part can then help low-tier agents learn strategies or policies to solve the resource allocation task in real SoS.

A natural counterpart of the implemented two-tier learning is joint learning, in which all agents (i.e., the resource manager and SoS agents) learn their policies simultaneously, in a single tier framework. Although more flexible than the two-tier framework, joint learning can be quite unstable. This is due to the fact that from each agent's perspective, other learning agents are parts of the environment, which makes the environment unstationary, creating several theoretical and practical issues for agents' learning [33]. The two-tier framework also provides additional benefit when it comes to implementing some levels of specialization for the agents during the learning stage, something that is difficult to achieve in a joint learning scheme. Using two-tier learning, we can let SoS constituents learn their responsibilities during Tier I training process without imposing resource limits, then train them to coordinate with resource manager in the Tier II stage when resources are limited.

Assigning resource to an AI agent takes a high degree of trust, not only from users but also from stakeholders [34]. Trust is achieved, in part, by human-centric approaches that facilitate stakeholders' understanding of the process and interpretation of the outcomes. Having interpretable AI-generated decisions is particularly crucial for certain classes of complex sociotechnical systems (e.g., energy, mobility, healthcare) for which the stakeholders' desired attributes go beyond system efficiency, robustness, and resilience and include considerations such as equity and fairness [35–37]. As an alternative to the black-box approach, explainable AI is becoming increasingly important, and there have been several approaches designed to develop explainable AI in recent years [38,39], including in some system engineering applications [40]. Yet, most existing methods for explainable AI have focused on supervised and unsupervised learning, and less attention has been paid to the explainability of sequential decisions enabled by RL [41–43].

We focus in this paper on interpretability, a property that some authors consider to be an attribute of explainable AI and others distinguish between the two. Following recent recommendations by the National Institute of Standards and Technology (NIST) as suggested in few papers [44,45], we define interpretability as a property of the AI model that enables human to form a meaningful mental representation to make informed high-level decisions. This definition makes it different from another important property—one that gives a detailed *technical* description of the causal sequence that generates the model outcomes. In fact, as Lipton [46] emphasizes in his widely cited paper, an interpretation could be informative even if it does not shed light on the inner workings of a model. Consequently, and as the second contribution of this paper, we take a *post-hoc* approach toward RL interpretability to infer dynamic decision heuristics from the RL agent that is in charge of dynamic resource allocation. In addition to the trust-building benefit, these AI-generated decision heuristics can also be used by human decision-makers, even without any AI present. By extending this approach to more complex applications, such AI-generated heuristics can benefit a wide range of organizational and sociotechnical dynamic decision schemes and design problems [47–49] that are otherwise determined by human intuition, experience, or anecdotes [50–52]. The output of this part can be considered as a recommendation system for decision making. By using heuristics generated out of learned policies, we can provide recommendations to facilitate decision making for engineers in the real SoS resource allocation problems, even when the resource allocation is not handled by an AI agent.

We apply the proposed two-tier learning framework on a customized OpenAI Gym environment [53,23], which is a benchmark environment with continuous action space for MAS and set it such that it provides us with the minimum set of necessary components to build a multi-agent system-of-systems. We then compare the results of the proposed framework to baseline methods of resource allocation to show the superior performance of the two-tier learning scheme. We then use the results of this experiment and apply our heuristic inference method to interpret the decisions of the resource manager across different settings of environment and agents' parameters.

In summary, this paper makes the following contributions to the literature on AI-driven resource allocation decision for systems-of-systems. (i) Introducing a two-tier reinforcement learning scheme for dynamic resource allocation problem in systems-of-systems. (ii) Interpreting RL-manager's behavior and deriving useful heuristic to establish trust by moving toward interpretable AI and improving decision heuristics used by human agents. (iii) Analyzing key trends in the learned behavior as a function of changes in the environmental variables. The outcome of this paper can be used in two interrelated ways. It can be used as a computational framework and learning algorithm for resource allocation problem in SoS, when the resource management is dynamically decided by an AI agent. It can also be used as a recommendation system for engineers during the system design state, even when RL is not a part of the system itself and when the resource allocation is not delegated to AI agents.

In the rest of the paper, we first provide some background materials for major methods and themes used in this paper, then we proceed to explain the methods and algorithms introduced and used in this work. This will be followed up by the explanation of the experiments and their results. Finally, we put the work in the context by drawing some conclusions.

2 Background

In this section, we reviewed some of the existing works which relate to our paper. We first reviewed the current research and main challenges about resource allocation in systems-of-systems, which is the main problem we would like to solve in this paper. Then, we reviewed the papers about RL-driven resource allocation, which is the core method we are using in this paper. Then some works about explainable AI especially in RL fields have been reviewed, we also analyze the core difference between the methods used in some of the existing works and our heuristics-generating process. Finally, we reviewed some papers about multi-agent RL, which is one of the most challenging fields for RL community in recent research, those papers also provide some evidences of the methods we are using especially in Tier I learning.

2.1 Resource Allocation in Systems-of-Systems.

Systems-of-systems: A shift in the paradigm of system design has been made to take advantage of the capabilities provided by distributed, autonomous, or semi-autonomous decision-making. Consequently, systems-of-systems have become popular in a variety of engineering design applications such as Internet of Things [54], autonomous vehicles [55], and urban air mobility vehicle design [56], smart grids [57,58] fractionated satellite systems in which detection, processing, and communication tasks are dynamically assigned to members of the satellite cluster [59,60]; sustainable manufacturing systems [61], communication networks in which frequency spectrum is dynamically allocated for efficient use [62]; and groups of unmanned, autonomous vehicles (such as aerial drones) that make dynamic assignment of tasks between them and each can make use of information gathered by other members of the group [63]. Although many definitions are offered in the literature for SoS (see Ref. [64] for a discussion of different definitions), the term loosely refers to a collection of heterogeneous, independent systems that collaborate in order to create capabilities beyond those of a single system. SoS is a special class of multi-agent systems [65], characterized by all or a subset of the following characteristics: autonomy, belonging, connectivity, diversity, and emergence, evolutionary development [3,66], which makes the behavior of the systems synergistic and adaptable. Synergistic collaboration between individual autonomous systems is an important aspect of a system-of-systems, which some authors have referred to as collaborative systems or systems-of-systems interchangeably [2]. Authors have also classified SoS into different types (directed, collaborative, virtual [2], and later, acknowledged [67,68]), depending on the power and scope of the central authority in setting the purpose or managing operations and the nature of collaboration among the systems.

SoS resource allocation: Systems-of-systems rely on a variety of local resources whose management is critical for ensuring the smooth operation of the system. Since they tend to operate in highly uncertain environments, it can be difficult to predict the demands for resources in various parts of the system at any moment in time; as a result, even if the total demands can be met, achieving an effective distribution of resources is not an easy task. It is important for a resource management scheme for a systems-of-systems to have a few key features. First, it needs to acknowledge the autonomy of the system's constituents and leverage such autonomy to balance centralized and decentralized decision-making. Additionally, it must be able to infer dynamic resource demand by relying on decentralized—and not necessarily reliable—local information provided by constituents of the system. Furthermore, the system should balance cooperation and competition between the autonomous agents within the system. Achieving this balance is difficult because too much competition results in inefficiency at the system level, and too much cooperation can reverse the advantages of autonomy at the constituent level. Finally, the scheme should account for tradeoffs that arise from the way one type of resource (e.g., energy) affects another type of resource (e.g., communication bandwidth). An approach of building such a scheme is the design of a suitable network architecture that governs the interaction of the constituents of different systems. The idea here is that by designing such systems, we can compensate for the inefficient allocation of resources at the beginning by designing selective pathways for resource sharing among different constituents. This approach was pursued by Mosleh et al. [6] to find the best way to connect the system components in order to enhance resource access in uncertain environments. The framework introduced in that paper explicitly incorporates costs of connection and the benefits that are received by direct and indirect access to resources, takes a strategic network formation perspective [69], and provides measures of the optimality of connectivity structures.

From an architecture perspective, the frameworks discussed earlier are dynamic; however, they are static when it comes to the behavior of components. This is because the framework does not take into account the complexities caused by goal-seeking behaviors of agents such as their learning over time and the way that impacts the strategic network formation model. This is especially true given recent developments in deep reinforcement learning and the anticipated expansion of its application to individual systems within an SOS. This can result in more complex and challenging behaviors—and disrupt static resource management programs—on one hand, while creating opportunity for establishing more sophisticated resource management schemes on the other. DRL is used in this paper to create a dynamic resource management scheme that goes beyond the previous approach by making use of both the autonomy and learning capabilities of system constituents by modifying the local parameters like information access ranges or resource cost functions.

2.2 Reinforcement Learning and Resource Allocation.

In the past, resource allocation has not been a primary application of RL; however, several applications of RL for resource allocation have emerged recently, especially in communication systems,

computing, and multi-agent robotics. Here, we review few examples from each of these lines.

RL-driven resource allocation in communication systems: The central focus here is on letting RL decide who to serve, by how much resources, and when. RL has been used in different communication systems, including 5G and satellite communication. For example, in Ref. [70], the authors equip each transmitter in wireless network with a deep RL agent, which can receive delayed observations from their associated users and their neighbors, and choose which user to serve and what transmit power to use. Resource scheduling is a key issue in the 5G networks, and authors in Ref. [71] suggest a method based on asynchronous actor–critic (A3C) RL to solve resource scheduling in 5G radio access network (RAN) slicing. At a system-level, RL has been used in satellite communication by Ferreira et al. [72] who propose a novel radio resource allocation algorithm based on multi-objective RL to manage available resources and conflicting mission-based goals in satellite communication.

RL-driven resource allocation in computing: Problems similar to those in the communication systems also emerge in cloud computing, where the question of how to allocate computing resource has been investigated using RL in a few studies, such as Ref. [73] who applies deep RL in conjunction with long short-term memory (LSTM). Trust is a key in many computing systems, and some authors have used RL to integrate trust-related considerations into resource allocation schemes. One such example is Ref. [74], who looks at resource allocation problem in edge computing by first modeling the service provisioning system as a Markov decision process (MDP), then uses Q-learning to find the strategy that can also maximize the services' trustworthiness.

RL in robotic resource allocation: Apart from using RL to control the robotic behavior, several researches focus on using RL to manage resource within a robotic team to improve the group performance. For example, in cloud robotic system, authors in Ref. [75] use RL to help cloud decide whether a request from robots about computing service should be accepted and how many resources are supposed to be allocated. Offloading problem in cloud robotic system has also been studied by Chinchali et al. [76] where they used deep RL to decide how and when should robots offload sensing tasks.

Balancing system-level tradeoffs can also be delegated to RL agents. For example, problem about balancing the performance gains and information exchange in unmanned aerial vehicles (UAVs) system has been studied in Ref. [77] using multi-agent reinforcement learning.

Our work is similar to several of these studies in computing and robotics, when it comes to using RL for resource allocation and tradeoff resolution. However, unlike those studies, we use a two-tier RL-based framework where we use RL to guide the system manager (Tier II), while having a more realistic model of autonomous lower tier agents (Tier I). This is in contrast to most other frameworks who use more static and compact models of agents behavior, which makes them applicable to narrower contexts. The two-tier framework, however, is more general and can be used in a wide range of contexts. This is because, unlike most existing works, the autonomous behavior of the lower tier agents is learned within the framework.

2.3 Explainable and Interpretable Artificial Intelligence. Delegating resource management of complex systems-of-systems in applications as diverse as medicine, mobility, defense, energy, and education has high stakes implications, not just from the technical perspective (efficiency, reliability, robustness, and resilience), but from the social perspective as well, in terms of equity, access, and fairness. In most of these cases, explanations are necessary for all stakeholder groups (users, designers, operators, and policymakers) to understand, trust, and view the AI agents as partners [38]. Although still an open challenge, several recent suggestions have been made to make DRL more explainable.

In machine learning, decision tree [78] is a popular method which can increase the explainability of the algorithm, because every node represents a rule which guides to make decisions. Some researches combine the tree structure with RL to improve the explainability. For example, a novel interpretable Q-learning algorithm has been proposed in Ref. [79] based on decision tree, which only increases tree size when the estimated discounted future reward of the overall policy increases by a sufficient amount. The authors in Ref. [80] further propose a method by using the proven universal approximation property with fuzzy predictions to translate RL agent's behavior to a simpler and easier set of if–then rules.

Attention mechanism is a powerful method which is widely used in natural language processing [81] and computer vision [82]. One of the benefits of using attention is that it helps build connections between parameters in the model and the corresponding effects, which can further generate causality and explainability. Some RL methods combine attention mechanisms to improve their explainability. A soft attention model has been applied to reinforcement learning domain in Ref. [83], which can show which parts in observation that is mainly used by agent to select its actions in each timestep. Similarly, some works use attention mechanism but focus on visual-based RL tasks, which shows clear connections between specific regions in the image and corresponding decisions. For example, a self-attention method has been applied in Ref. [84] incorporating neuroevolution concepts to explain learned behaviors from the image in video games like "CarRacing" and "DoomTakeCover." By introducing key-value memories, attention and reconstructable embeddings, models in Ref. [85], can easily invert influential causes toward actions from the latent space to input space, which can also show strong relations between specific areas in input image and output actions.

Languages and symbols contain plenty of information, it is also the most natural way to explain behaviors for human. Some papers try to take advantage of them in RL algorithm to improve the explainability. For example, RL policies can be translated into a high-level, domain-specific programming language using neurally directed program search (NDPS) in Ref. [41]. A symbolic deep reinforcement learning (SDRL) in Ref. [86] can decompose the high-dimensional sensory inputs into a sequence of symbolic tasks and solve them using the learned policies, where the sequence of symbolic tasks explain how RL agents solve the tasks. A natural-language-based method in Ref. [87] utilized programs, structured in a formal language, as a precise and expressive way to specify tasks, which can clearly explain policy for solving each task.

Most of those works focus on improving the explainability of RL algorithm in AI field, which needs to combine additional methods to explain RL policy. Although many of them show interesting results, but lots of efforts on method designing and large computational resources are required. This work focuses on augmenting the interpretability in deep reinforcement learning, which requires no additional methods but interpret RL agent's behavior directly. Specifically, our approach in this model takes a simpler form and focuses on generating static and dynamic heuristics that recommends *optimal* relative frequency of various resource allocation schemes as a function of the stage of the life-cycle of the SoS mission, or different combination of crucial cost parameters of the system.

2.4 Multiagent Reinforcement Learning. A multi-agent system is a framework for describing components of complex systems-of-systems. It is difficult to train agents in multi-agent systems not only because of the curse of dimension but also because of the problem of unstationarity caused by agents' simultaneous learning. One way to extend single-agent reinforcement learning to multi-agent scenarios is to use a large learning model to control the actions of all agents. This method has two problems, however, first it could be in contrast to the autonomous nature of agents with respect to one another; moreover, as the number of

agents increases, the state space and action space increase exponentially, which makes this method not feasible in most practical applications. Alternatively, one could naively degenerate the model into a set of independent RL agents, that is to equip each agent with the single-agent reinforcement learning method [88]. This method violates the assumption of a stable environment in reinforcement learning, because from the perspective of each agent, other agents with changing policies become part of the environment, making the model unstationary. Despite the fact that independent RL introduce unstationary to the system, it can achieve decent performance in some application [89,90].

There have been several methods proposed for solving these challenges in multi-agent reinforcement learning. The most popular model of learning is central-training–decentralized-execution (CTDE) [23,24], where during training, the critics have access to the observations of all agents, while in execution, actors only have access to their local observations; in contrast, critics in DTDE can only get access to their local observations to estimate the value functions, and actors in execution can also just have access to their local observations to make decisions. In recent years, some papers have examined the theoretical analysis of CTDE, and they concluded CTDE introduces high variance for policy gradient process in training phase, but it has low bias and can learn a better value function than DTDE [91,92]. Bias in here means the accuracy of the estimated value function; while the variance measures the uncertainty of the learned stochastic policy. Consequently and like most learning algorithms, there is a variance-bias tradeoff for different tasks and scenarios. In our application, due to resource limitations in the low-level training process and since the environment is only partial observable, the environment becomes highly uncertain from the perspective of each individual agent. Consequently, CTDE will further destabilize training, resulting in poor performance in practice. Therefore, while acknowledging certain merits and advantages of CTDE, we chose DTDE as our learning method.

3 Method and Proposed Framework

We developed a two-tier learning framework for dynamic resource allocation in systems-of-systems that uses multiagent reinforcement learning based on a partial observable Markov game model [31] of multi-agent systems interactions. To make the framework more interpretable, we implemented a post-hoc analysis step that infers decision heuristics from the learned behaviors of the SoS agents. Below, we explain different ingredients of the framework and review the proposed algorithms.

3.1 Partial Observable Markov Games. The multi-agent RL that we use in this paper is based on the framework of Markov games [31], sometimes also called stochastic games as formulated back in 1954 by Lloyd Shapley [93]. In order to model the fact that SoS agents only have local information access, we use the partial observability assumption for the Markov game. A partial observable stochastic games (POSG) can be described using a tuple $< I, S, s^0, A_i, O_i, T, R_i >$, where

I is a set of agents index from $1 \ldots n$.
S is a set of possible states.
s^0 is the initial state of the environment, where $s^0 \in S$.
A_i is a set of available actions for agent i.
A is joint action space for all agents, $A_1 \times \ldots \times A_n$.
O_i is observation perceived from agent i.
O is joint observation space for all agents, $O_1 \times \ldots \times O_n$.
T is transition function, where T: $O \times A \rightarrow O$.
R_i is reward function for agent i, where $O_i \times A_i \rightarrow R_i$.

The objective for each agent in POSG is to learn the optimal policy π_{θ_i}: $O_i \times A_i \rightarrow [0, 1]$, which can maximize the expected return: $\sum_{t=0}^{T} \gamma^t r_i^t$ where γ is a discount factor and T is the time horizon. The learning of the policy is achieved by DRL, as we explain next.

3.2 Deep Q-Learning and Deterministic Policy Gradient. Deep Q-learning [19] is a very popular algorithm and used commonly in many RL tasks. The agent receives the observation o of the environmental state s, then makes use of the action value function for policy π as $Q^{\pi}(o, a) = E(R \mid o = o^t, a = a^t)$ and always takes the action greedily which can maximize action value. Q-learning learns the action value function corresponding to optimal policy by minimizing the following cost:

$$L_\theta = E_{o,a,r,o'}[(Q(o, a \mid \theta) - y)^2] \tag{1}$$

where $y = r + \gamma \max_{a'} Q'(o', a')$. The Q function is represented using deep neural network, θ is the parameters of deep neural network. o' is the observation in next time-step after taking action a and a' is the action which was taken in next time-step. Function Q' is the target Q function which can help stabilize the learning process, it is a copy of the learning Q function and will be updated after certain rounds. The details are in Ref. [19].

Policy gradient method [94] is another very popular RL algorithm. The core idea is directly adjusting parameters of θ of the policy to maximize the objective $J(\theta) = E_{\pi_\theta}[R]$ by taking steps in the direction of $\nabla_\theta J(\theta)$. The gradient of the policy can be written as

$$\nabla_\theta J(\theta) = E_{o,a}[\nabla_\theta \log \pi_\theta(a \mid o) Q^{\pi}(o, a)] \tag{2}$$

$Q^{\pi}(o, a)$ can be updated using the method described in Q-learning section, this method becomes the actor–critic algorithm [81]. If we extend the policy to deterministic policies $\mu_\theta : O \rightarrow A$, we get DDPG [28], the gradient of the objective changes as

$$\nabla_\theta J(\theta) = E_{o,a}[\nabla_\theta \mu_\theta(a \mid o) \nabla_a Q^{\mu}(o, a) \mid a = \mu_\theta(o)] \tag{3}$$

Also, the representation of $\mu_\theta(a \mid s)$ is using deep neural network, the deep neural network is parameterized by θ.

Deep Q-learning and DDPG are model-free reinforcement learning algorithms, which means the methods will not estimate the environment dynamic (transition function and reward function), only use rewards to optimized their policy. In contrast, model-based RL methods will first estimate a predictive model of environment dynamics, and then based on the estimated model to do planning which can optimize the objective [95].

3.3 Two-Tier Learning Framework. For a framework that can address the challenges described in the previous section regarding dynamic resource allocation, it must be dynamic and adaptive and take into account not simply the changes in the SoS environment, but the changes in the SoS itself as the result of learning, strategic behavior, and self-optimizing behaviors by the agents. The framework should also consider some fundamental tradeoffs of multi-agent systems, such as the tradeoff between system-level (and not just agent-level) exploration and exploitation [96,97], and cooperation and competition [12]. Critically, the framework should be able to foster the sharing of resources as a form of cooperation when it benefits the system, while recognizing the autonomy of the SoS agents. Our two-tier learning framework for dynamic resource allocation was designed with these requirements in mind.

The overall idea of the proposed framework is represented in Fig. 1. The framework consists of a lower level learning (Tier I) in which the SoS agents interact with the environment, and a higher level (Tier II) in which the resource managing agent interacts with the environment. In Tier II, managers assign resources—through the environment—based on the states of all SoS agents at each time-step. For the Tier I task, on the other hand, SoS agents rely on their local observations and updated resources to make decisions and solve the problem.

Corresponding to these two layers, learning occurs twice. First, the SoS agents are pretrained in the original environment, assuming no resource limit, they can learn how to solve the task in an ideal unconstrained context. The outcome of this tier is a set of agents that can match the environment states to their goal-seeking actions. The agents' policies learned at this stage are then fixed to

be used in the next stage. Since the action space is continuous in general (e.g., physical movement, amount of resources to share, level of information exchange, and so on), we choose DDPG [30] as the learning algorithm. It is an off-policy method and uses two separate pairs of actor–critic neural networks to concurrently learn the Q function and policy for continuous action space. More details on this method can be found in Ref. [30], as well as in a previous work by the authors in Ref. [21].

As described in Sec. 2.4, we use the DTDE multi-agent learning scheme, in which the agents are trained independently, which is useful for reducing variance in training, especially in this highly uncertain environment. As [91] argues, although using this scheme aggravates the bias-variance tradeoff due to limited samples and less correct value functions, in practice, it regularly gives more robust performance since stable policy gradients turn out to be more crucial than stable value functions in most applications.

Second, the SoS manager will learn in the second stage. The SoS agents have been pretrained in the first stage as we mentioned, and they already possess the ability to seek the landmarks if given appropriate resources. Thus, the outcome of this tier is a SoS manager that can use minimum total resource and allocate them in the most efficient way, which can maximize the performance of SoS agents. Because the action space is discrete in general (different options of resources), we choose Q-learning [19] as the learning algorithm. It is an off-policy method and uses two separate pairs of neural networks to learn the Q function and generate policy using the learned Q function for discrete action space. More details on this method can be found in Ref. [19].

Algorithm 1 Tier I: Learning process for SoS agents

Initialize Replay Buffer D_i, actor μ_i and critic $Q^i(o, a)$ for each agent.
for *E-poch* $= 1, 2, \ldots$ **do**
  for *agent* $= 1, 2, \ldots, N$ **do**
    Run policy μ_θ in environment for T time-steps
    Store each agent's history $o_i^t, a_i^t, r_i^t, {o'}_i^t$ to D_i
  end for
  In training process, each agent i will sample experience o_i, a_i, r_i, o_i' randomly from D_i
  Update actor μ_θ using Eq. (3)
  Update critic $Q^i(o, a)$ using Eq. (1)
end for

Tier I Learning: The learning algorithm of this stage works as follow and is summarized in the pseudo-code presented in 1. For agent i, the algorithm first initializes a replay buffer D_i to record history when interacting with environment, a random policy μ_i (actor) to carry out exploration at early episode, and a critic function $Q^i(o, a)$ to evaluate the state–action pair which can facilitate learning. Exploration involves adding noise to the actions that decays with learning, as suggested by Lillicrap et al. [30]. Each agent will observe the current state of the environment and use its old policy μ_i to make a decision, the environment will then use its transition function to update its state and output reward to each agent. The actor in each agent will observe the current state of the environment and use its old policy μ_i to make a decision, the critic will estimate the value function for observation–action pair based on the rewards they get and the old estimated value function, the environment will then use its transition function to update its state and use reward function to output reward to each agent. This RL algorithm is model-free [95] and the agents learn from their observations, actions, and rewards only. Iteration will last T time-steps. Observations, actions, rewards, and new observations are then stored in D_i.

During the training process, each agent will randomly sample experience from its replay buffer. Then, each agent will update their policy μ_i using Eq. (3), and update their critic $Q^i(o, a)$ using Eq. (1). Since we assume partial observability on behalf of the SoS agents, in our actor–critic model we use agents' observations instead of the true state of the environment. During the learning process, the process will iterate until the learning is complete.

Tier II Learning: The high-level manager needs to learn how to assign appropriate resources in each time-step during the game. The system can observe the concatenation of local observations of SoS agents and select the preferred resource. The training algorithm we use for managers is Q-learning [19]. In the algorithm, we began with an initialized replay buffer D and a random state–action function $Q(o, a)$. During the training phase, the resource manager uses the epsilon-greedy method [19] to make decisions. The idea is that when choosing action, the manager has a ε probability of selecting action at random from the available action set (exploration) and a $1 - \varepsilon$ probability of selecting the best action given the current value function. With the training process, the epsilon will always start at 1 and gradually decrease. The resource manager always takes the best action during the execution phase based on their learned policy (no exploration). Resource manager determines action using the epsilon-greedy method, while environment determines new state and reward based on its transition function and reward function. Like Tier I, the RL model here is also model-free.

Resource manager will then store observations, actions, rewards, and new observations to replay buffer D. When updating the policy, resource manager will randomly sample the previous experience from D, and update its action value function $Q(o, a)$ using Eq. (1). We are replacing real state with observation because resource manager is in a partially observable environment. There, y contains a value function Q', it can be represented using the same value function, but the problem is the model will be difficult to converge [19]. Therefore, we use a target Q function which copies parameters from the original Q function every specific learning epoch. Learning will continue for a certain number of iterations. The pseudo-code can be seen in Algorithm 2.

Algorithm 2 Tier II: Learning process for the SoS resource manager

Initialize Replay Buffer D, action value function $Q(o, a)$
for E *poch* $= 1, 2, \ldots$ **do**
  for time-step $= 1, 2, \ldots, T$ **do**
    Choose action a at each time-step using
    $$a = \begin{cases} \arg\max_{a^t} Q(o^t, a^t), & \text{probability } 1 - \epsilon \\ \textit{random action}, & \text{probability } \epsilon \end{cases}$$
    Store each manager's history o^t, a^t, r^t, o'^t to D
  end for
  In training process, manager will sample experience o, a, r, o' randomly from D
  Update action value function $Q(o, a)$ using Eq. (3)
  Decay ϵ until minimum value
end for

3.4 Explainability and Heuristic Inference. While the proposed framework and RL algorithm can help us solve the resource allocation problem of SoS in a specific environment, we would also like to infer decision heuristics based on the learned policy of the RL agent to not only gain more trust to delegating crucial decisions to the AI agents, but also possibly use these inferred heuristics as recommendations to human agents who are in charge of resource allocation in dynamic multi-agent environment where both cooperation and competition are important.

Our approach to interpret the behaviors of the RL agents is twofolded. First, we have a static, yet parameter-dependent analysis of the relative frequency of using each decision from the action space by the resource managing agent. We do this in two dimensions, by changing the relative ratio of the relative cost of resources to the *importance* of the SoS mission, and the relative cost of the two resources with respect to one another. We then proceed to infer a

dynamic set of heuristics, that is what types of decisions the resource manager should use at various stages of the life-cycle of the mission (or the system). We do this by dividing the span of the mission into a number of distinct stages according to transitions in the RL agents' behaviors, then providing intuitive interpretations on those behaviors. Given the relatively simple setup of our experiments, few of these interpretations come as real surprises; however, it is reassuring that the RL agent can learn dynamic decisions that match well with our expectations and intuitions in most cases. Even in these simple setups, we will show a few examples where the heuristics inferred from the behaviors of the RL agents are not immediately obvious, especially in the case of varying relative cost of the two resources.

4 Experiment and Results

4.1 Environment Settings. As mentioned earlier, our goal here is to create a dynamic resource management scheme that goes beyond the previous approaches and takes into account both the autonomy and learning capabilities of system constituents by manipulating the local parameters such as the information accessibility range or the cost function of resource consumption. We further need a framework that incorporates both cooperation and competition dynamics among agents. Furthermore, it needs to be adapted to the two-tier framework we described in the previous section. The environment we use for the experiment here is also motivated by a common application of SoS that involves a multirobot resource allocation problem [55,75–77], which requires a reasonable policy for allocating resources so robots can navigate to the appropriate location and complete the task. In such cases, one of the key performance metrics is having the autonomous SoS agents to maximize their coverage of the potential targets as quickly as possible and in a resource efficient manner.

To build such an environment, we customize the "Cooperative Navigation" environment, first proposed in Ref. [98] and used by a variety of other applications, as our test bed. The environment (shown in Fig. 2) is simple but contains all key components of a SoS, such as multiple agents, multiple resources, and cooperative objectives. The environment consists of continuous state space, action space, and discrete time. Crucially, agents are heterogeneous in function and policies. This means that they do not have identical observations, and they do not act according to the same policies. To keep the experiment traceable, we use three landmarks and three agents, although the model can be extended to larger systems.

The agents, representing the SoS constituents, have limited observation range. Depending on the application, it can be related to the available surveillance energy, the amount of output power (wireless receivers), or the size and shape of the antenna or camera. In any case, we assume that they can accurately observe the landmarks and other agents within their available *vision* range. In particular, the observation from each agent contains its own position and velocity, relative distance from other agents within its visual range, and relative distance from landmarks within its visual range. These will create a 14 dimensional observation vector for each agent. Agents can move and their action space is composed of horizontal and vertical movements, a two-dimensional vector. The goal of the system is to maximize the surveillance of all the targets (landmarks), thus agents are rewarded inversely proportional to their distance from landmarks. They are also penalized if they collide with one another. Speed of the task also matters, so the goal is to spread agents to every landmark as quickly as possible. Target landmarks are not preassigned to the agents, which makes coordination and collision-avoidance another challenge, which also has key implications for our results.

To accrue both the benefits of centralized resource allocation and decentralized mission-related decisions, the resource managing agent that we include in the tier II learning is *omniscient*, but not omnipotent. This means that it has a full observability of the entire SoS, but cannot directly change the behavior of the agents. Its observation consists of the concatenation of all of the SoS constituents' observations (a 42-dimensional vector in our experiment). Its action indicates which resource it is choosing for the system at every period. Resources are costly to the resource managing agent with tunable cost ratio for different types of resources. The manager also receives benefits proportional to the sum of rewards of individual SoS agents. Hence, the objective of resource manager is to optimize the overall resource usage during a span of time (that can mean the life-cycle of the system, or the span of a mission, depending on the context and application) to optimally balance between the performance of the SoS and resource cost. This is clearly a dynamic allocation problem where the optimal decision depends on time and the stage of the life-cycle of the SoS or stages of its mission. The environment is rich enough to capture this dependency well. We consider two types of resources: vision and communication. We assume that if an agent is within the visual field of another agent, it can share its information by using some communication resource. In practice, the communication resource can translate into bandwidth, transmission power, information processing power, or memory storage.

We made a few assumptions for the environment: First, the manager is assumed to know the total sum of all agents' local observations at every time-step. Agents, however, only have their own local information that can be gained, either by direct observation within their visual range (the vision resource) or through communication with other agents, if those communication channels are

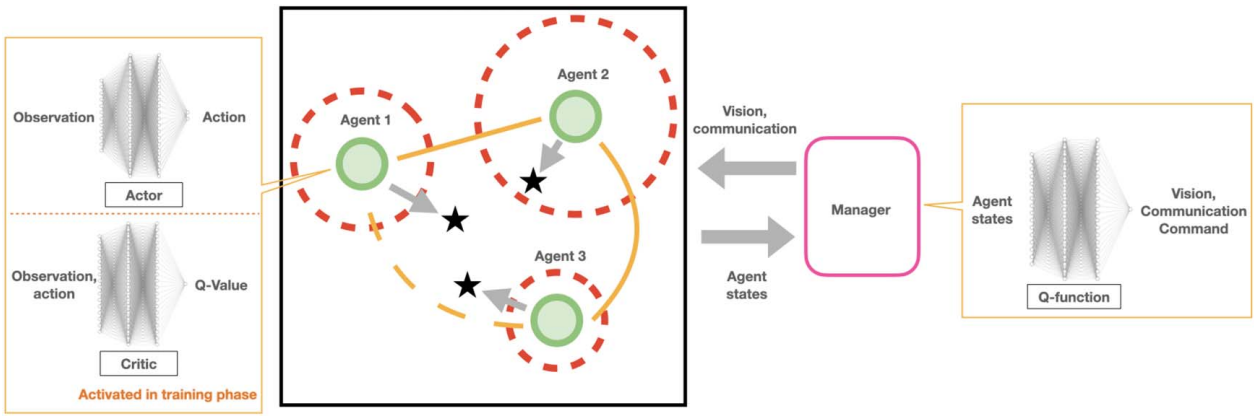

Fig. 2 Agents should spread on the appropriate landmarks (star object) with less collisions to get the best total rewards. The dotted circles show the visual range of each agent, the linking lines represent the communication channel between agents, the channel can also be deactivated as the dotted line represented. The detailed technical structure is represented in the left and right blocks.

Table 1 Details to describe the observation space, action space, and rewards for agents and resource manager

	Symbol	Agents	Resource manager
Observation	O	The position and velocity of themselves and positions of other agents and landmarks within their visual range	The position and velocity of all agents and positions of all landmarks
Action	a	Forces they would like to impose on x-axis and y-axis, limited in $[-1, 1]$	No use: use no additional resource; additional vision: increase visual range of all agents from 0.8 to 1.2; allowing communication: allow agents to communicate their observation with other agents; use both: provide additional vision and communication at same time
Rewards	R	Regrets which are proportional to the distance between each agent and its closest landmark, plus a penalty of colliding with other agents	Agents regrets plus resource cost

enabled (the communication resource). We also assume that the signals that the manager and agents received about the current state of the system are fully reliable (no noisy signal is assumed), although this assumption can be added to the framework by adding more model complexity and without any fundamental changes in the framework. Second, we assume that the environment is evolving, and the state transitions are uncertain and are modeled using POMDP, as described in Sec. 3. In our current implementation, the evolution of the environment is due to changes caused by agents' actions, although other dimensions of evolution (e.g., in resources, targets) can be integrated in this framework with additional model complexity. Finally, the manager cannot directly control the movements of agents, and any control needs to be exerted via changes in the capacities of agents through dynamic allocation of visual and communication resources. In each game, all entities will be randomly initialized within the 2× 2 square, the basic visual range for workers will be 0.8, when additional visual resources are used, the workers will have 1.2 visual range. By using communication, agents can update their own observations by using the information of others, for example, if agent 2 observes a landmark, agent 1 will calculate its position based on agent 2's observation and their relative distance. The metrics used in evaluating agents' performance is regret, which is a negative value that is proportional to the distance between each agent and its closest landmark, plus a penalty of colliding with other agents. So if all agents spread on each landmark without collision, the regret is 0, which is the highest performance they can get. In each time-step, managers have four options to choose from, which are no additional resource, using additional vision, using communication, and using both. To keep the analysis easier in the next step, we set 0.0, 0.2, 0.1, and 0.3 as default values so that choosing the same option during the game will provide similar overall performance in different four cases. The metrics used in evaluating resource manager's performance is the sum of agents' regrets and the resource costs. All details are summarized in Table 1.

The actors, critics, and Q value function are represented by fully connected neural networks. For the resource manager, we estimate the Q value function with a 256*128 neural network with fully connected neural network. We use Adam optimizer with 0.0001 learning rate, the training for resource manager contains 30,000 epochs. The SoS constituents use 512*512*256 fully connected neural networks to represent critics, and 256*128 fully connected neural networks to represent actors. Importantly, we only use critics during the training phase. This way, during the execution

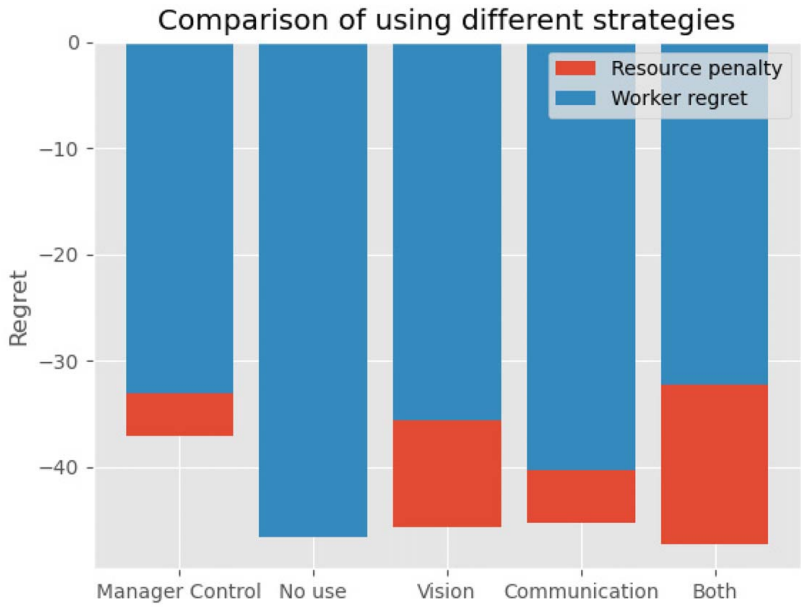

Fig. 3 Performance comparison between different strategies: "no use" means that manager always uses no resource during the game, "vision" means that manager always uses additional vision capacity, "communication" means that manager always allows communication, "both" means that manager always uses both resources. "Manager control" is using the learned manager to assign resources in different situations. Delegating resource allocation to the RL manager achieves comparable regret to the lavish case of using both resources all the time, but the method pays substantially less resource penalty.

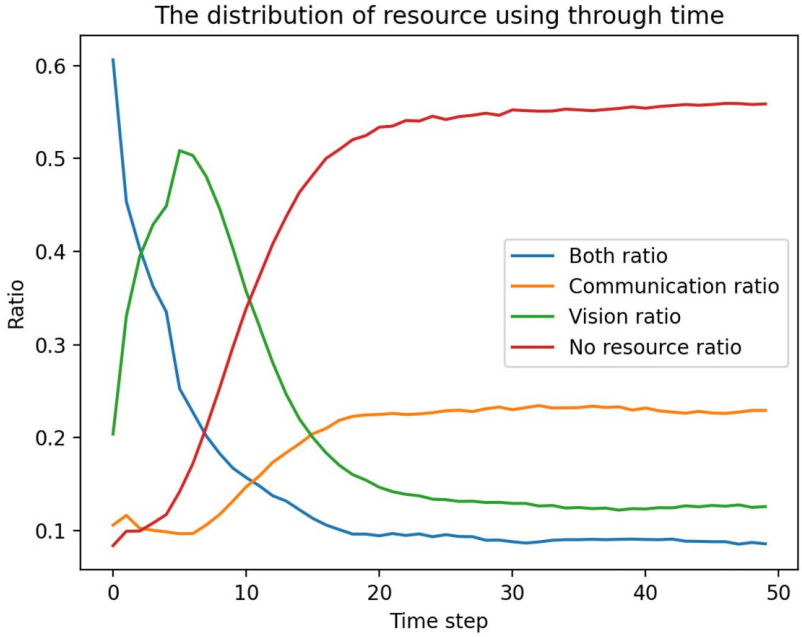

Fig. 4 Manager behavior during a game by showing the probability (ratio) of using each of the strategies at any time-step. The sum of all the ratios at each time is 1. "No resource" means how often the manager chooses to use no resource at any time-step, "vision ratio" shows how often the manager chooses to use vision capacity resource at any given time-step, "communication ratio" measures how often the manager chooses to use communication capacity at any given time-step, and "both ratio" shows how often the manager chooses to use both resources at any time-step.

phase, only actors are active and participating in making decisions. We use Adam optimizer with 0.001 learning rate for critics training and Adam optimizer with 0.0001 learning rate for actors training. The training for SoS constituents consists of 30,000 epochs. In our experiments, each game has 50 time-steps, and the following results have been tested on 1000 new games. An overview of the environment is presented in Fig. 2.

4.2 Experimental Results

4.2.1 Analysis for Manager's Performance and Behavior. The first natural question is how much we benefit from having the resource manager included in the system, so we first compare the overall performance of the system with the manager and with different schemes of static resource allocation. The static resource allocation means always choosing one resource option during the whole task. For example, static resource allocation of vision means: at each time-step, manager will always allocate additional vision capacity during the entire task. The result of this experiment is shown in Fig. 3. The figure compares the performance of the dynamic manager (the proposed framework) in the first column marked as manager control, to four other cases in which the two resources are either not used at all, are used separately, or used both all along (static resource allocation). The top portion of each graph shows the share of workers scores (regret) and the bottom portion shows the cost of resources paid by the SoS. Dividing the score of each scheme into two parts is illustrative to see the benefits of the RL method. While the dynamic scheme outperforms single-resource schemes in both resource penalty and agents' performance, it is a winner against the static, bi-resource scheme because of its lower resource cost score. Notably, in this experiment, the resource managing agent learns to reduce this cost without compromising on the actual performance of the SoS measured by regret.

While Fig. 3 gives us an overall sense of the performance of the proposed framework during the entire span of the mission, we are also interested in seeing the dynamic behavior of resource managing agent over the course of the task. This dynamic representation can help us with the intuitive explanation of the RL-suggested policies and can be used to infer dynamic decision heuristics for human agents, as discussed in the previous sections. To do this, we plotted the resource usage distribution through 50 time-steps in Fig. 4 for the four schemes described earlier. At each time-step, the value in the *y*-axis means the distribution of resource usage from 1000 games with random environmental states. The first observation here is that there is a clear variation in the ratio of various resource allocation as a function of time, suggesting that early optimal allocation schemes are quite different from later schemes. To better understand this, we can divide the game into three phases based on the results. We can see that the generous allocation scheme (allocating both resources to all agents) is selected very early on when the system is trying to conduct a broad exploration of the landscape, but quickly fades away as the mission progresses further. At this point the usage of the vision-only scheme is increasing, whereas the other two schemes (no resources and communication-only) are still low. This makes sense since at the earlier stages of the game vision is more important than communication. This is because there is a high likelihood that the agents will not be able to observe any landmarks, and communication is of little benefit if most agents do not have anything useful to communicate. So, the managing agents has learned to rely on agents autonomy (and their competition) to allocate more vision resource to increase the overall *information* of the system as a whole. We expect this to change as the agents gather more decentralized information from different parts of the system to the extent that information sharing and cooperation becomes beneficial. This indeed happens in the second phase(up to 20 time-steps), in which the managers' reliance on both costly schemes (all-resource and vision-resource) decreases. This is because after phase 1, the agents have gathered enough information about the environment, and the manager switches to other resources like communication, or even does not use them to minimize costs. We can consider this shift in the managers' performance as a transition for system-level exploration

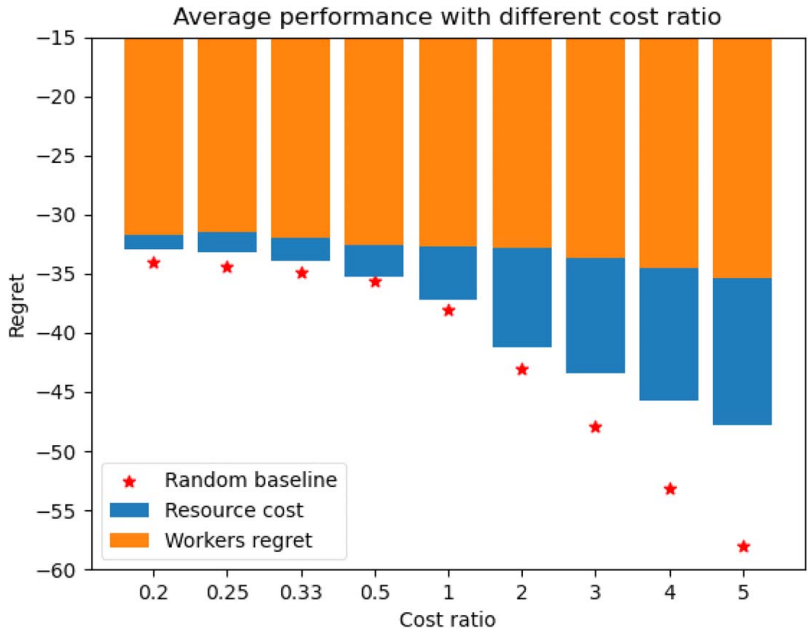

Fig. 5 The performance of learned models when the weight ratio between workers regrets and resource cost changes. The star markers represent the baseline performance when manager randomly picks resource options at each time-step.

mode (independent information gathering) to system-level exploitation (cooperative information sharing) and we can see that the resource manager has learned this transition by adding Tier II learning in our framework. Finally, in the third phase (20–50 time-steps), most agents have already approached the targets. At this stage, we see that cheaper resource communication is sometimes used, primarily to avoid a collision.

To understand how environmental settings will influence the behavior of managers, two additional experiments were conducted.

4.2.2 Influence of Ratio Between Task Scores and Resource Costs. We expect the behavior of the resource managing agent to be a function of the relative cost of the resources with respect to the benefits of better SoS performance. To study this dependency,

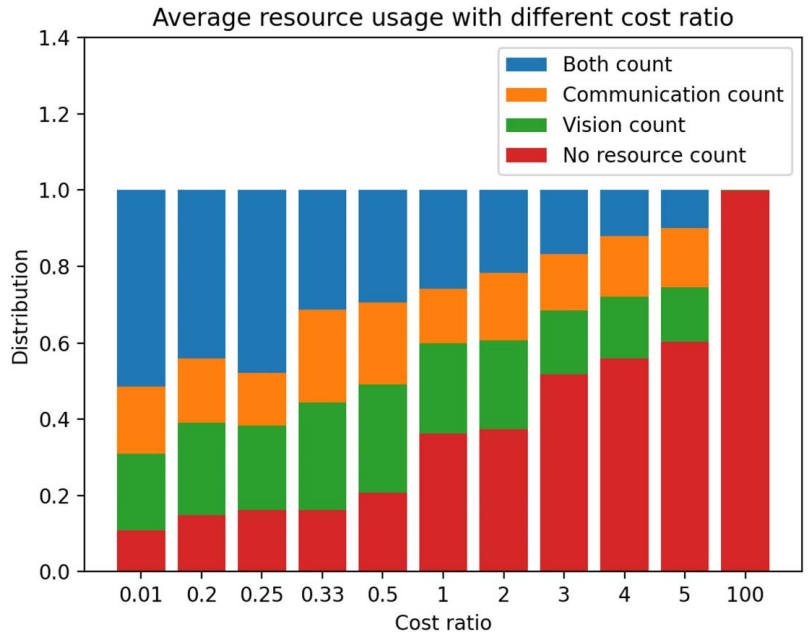

Fig. 6 The distribution for usage of different resource options when the weight ratio between workers' regrets and resource cost changes. Under different weight ratio between workers' regrets and resource cost, "both count" means the proportion of manager using both resources during the game, "communication count" means the proportion of manager using communication resources during the game, "vision count" means the proportion of manager using vision capacity resources during the game, "no resource count" means the proportion of manager using no resources during the game.

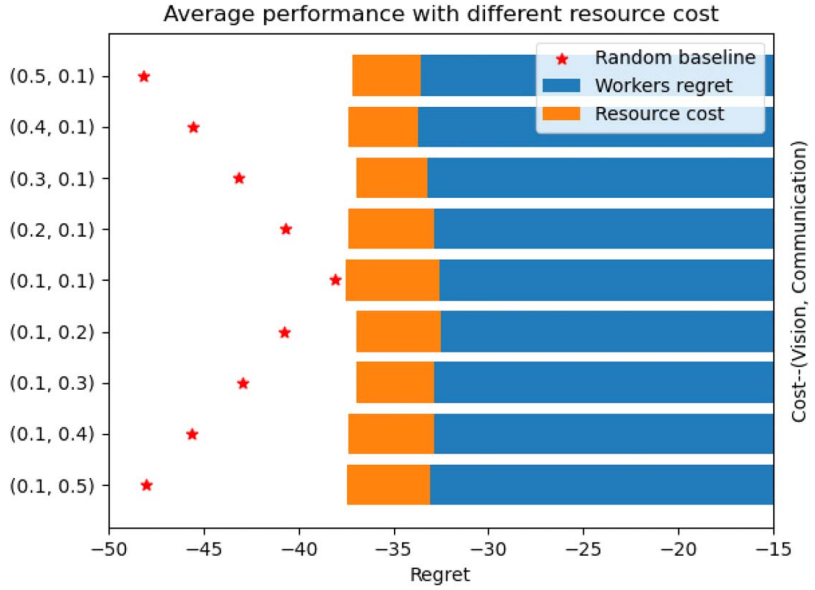

Fig. 7 The performance of learned models when vision resource cost and communication resource cost changes. The star markers represent the baseline performance when manager randomly picks resource options at each time-step.

we created a series of different environments (11 total) that only differ in the relative cost of the two resources (i.e., vision and communication). We then retrained these environments as before and tested them on 1000 rounds of mission.

Figure 5 illustrates the results of evaluating the performance of learned models under different environmental settings. To see the relative benefit of having a resource manager according to our two-tier framework, we compare the performance to a random baseline. This is a reasonable choice since what we care about is the relative benefit compared to a well-defined baseline that can be used in all the environments. The star markers represent this baseline performance in which the manager selects a command at random every step. As expected, we can see that our learned models perform better than the baseline for all environmental settings.

To provide an intuitive interpretation of the performance of the RL manager, we then proceed to see the relative usage of the four schemes introduced in the previous section for each of these environments. To do this, we plot the mean distribution of resource usage across all the 1000 rounds, as depicted in Fig. 6. The weight ratio is the importance between the original score and the resource cost. Due to the manager's need to trade off between these two parts, the different weight ratio will lead the manager to change its prioritization and allocation strategies. The results show reasonable trends, when the weight ratio changed from very

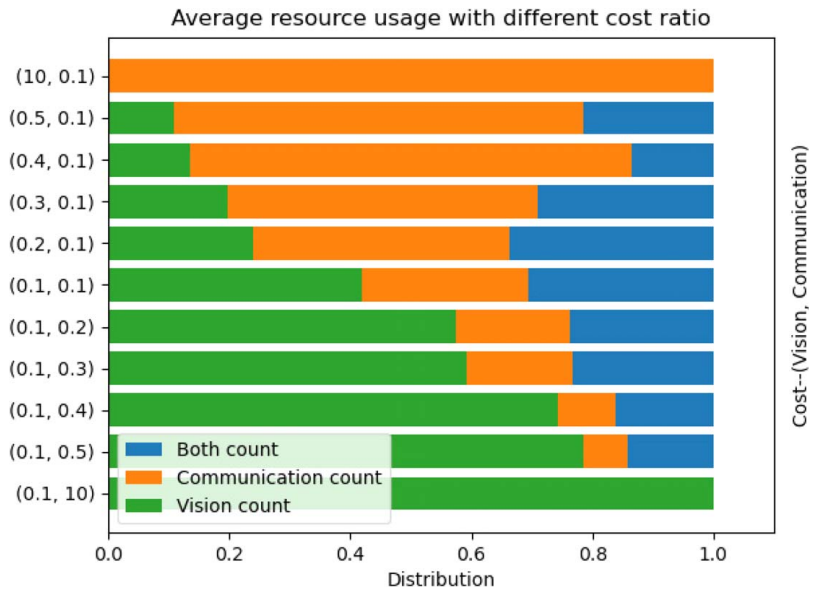

Fig. 8 The distribution for usage of different resource option when vision resource cost and communication resource cost changes. Under different weight ratio between vision resource cost and communication resource cost, "both count" means the proportion of manager using both resources during the game, "communication count" means the proportion of manager using communication resources during the game, "vision count" means the proportion of manager using vision capacity resources during the game.

small (caring much less about resource cost, compared to mission scores) to very large (having very expensive resources compared to what the SoS performance is worth), we can see that manager learn to use less and less resources when resource cost becomes more and more expensive, and when it becomes extremely expensive, manager learns to let go of resources altogether and rely instead on the status-quo performance of autonomous SoS agents.

4.2.3 Influence of Different Resource Costs. We also expect that the manager's policy changes as a function of the relative cost of the two resources (vision versus communication). To study this, we designed a new experiment and vary the cost of these two resources. We created 11 environments with only differences in resource costs per unit (e.g., (10,0.1), (0.5,0.1), (0.4,0.1),…, (0.1,0.5), (0.1, 10), re-trained the model in three runs and tested them for 1000 new games. Figure 7 shows the performance of learned models under different settings.

Once again, as expected, we see that our model always performs significantly better than the random baseline for all different environments. However, the relative benefit increases substantially as the costs of the two resources diverge from one another. This is because an asymmetric cost increases the benefit of an informed decision (i.e., only use the expensive resource when it is worth it) much bolder compared to an uninformed, random decision.

Like the previous two experiments, and in order to better explain the learned behavior, we plot the mean distribution of resource usage per game, as shown in Fig. 8. Since we are focusing on the usage of two resources, we exclude the "no use" portion and normalize the remaining three parts. We can see that in the two extreme cases, when the relative cost is 10 times higher for one resource, the manager solely relies on the affordable resource, as expected. However, even for cost ratio of 5, we can see that all three schemes have been used by the manager. The relative share of the schemes is not symmetric however, due to different functions of the two resources. Notably, as the cost of communication increases, we see that the manager sees more benefit in spending the expensive communication resources in conjunction with the vision resource (generous spending) than using it independently. This means that the hybrid exploration/exploitation (or competition/cooperation) modes are seen as more beneficial when the cost of communication and coordination is relatively high. This is an interesting observation and possibly a useful heuristic that needs to be explored further in future studies.

5 Conclusion

Dynamic resource sharing is a fundamental challenge in systems-of-systems in which autonomous agents (subsystems) need to balance their competition for different resources with cooperation to use those resources more effectively. While most existing methods rely on static incentive structures and interaction topology to balance these two modes of interactions, we argued in this work that reinforcement learning can be effectively used for moving the governance of such systems from static to dynamic.

To this end, we proposed a two-tier learning framework based on a decentralized scheme of multi-agent reinforcement learning. The framework includes SoS constituents and a resource manager who uses a holistic information about the state of all the SoS agents to indirectly lead them toward a balanced exploration versus exploitation on the one hand, and competition versus cooperation on the other. The experimental results on "Cooperative Navigation" environment show that the framework can help agents use minimum resources and reach a near-optimal performance compared to the environment with unlimited resources. Also, we used first level analysis of the learned policies across time and also across different environment parameters to infer dynamic decision heuristics. We do so to establish trust in delegating crucial resource management decisions to AI agents, as well as creating recommendations to human agents who are in charge of resource allocation in dynamic multi-agent environment where both cooperation and competition are important.

While we show the merits of the proposed two-tier learning framework for hybrid dynamic environments and take a few first steps toward explainability for this framework, our paper has a few limitations: First, the notion of cooperation and competition is modeled in a simple form of location information sharing. In practice, cooperation can be much more complex, even in a decentralized target surveillance task and includes possibility of establishing subclusters of cooperation, emergence of information sharing network structures, pair-wise trust building, and counter-measures such as gaming the system for agent-level benefits. While our framework can take into account many of these aspects, our experiments are too simple and ignore most of them. We can also extend the current system to include more heterogeneous agents that different in the types and functions (not just policies and resource access), something that is more realistic and can give rise to some of these complex emergent patterns of *coopetition*.

Moreover, while we believe that the DTDE scheme is the right choice for the multi-agent RL at hand, a more complete analysis would include a comparison of allocating policies (and subsequent heuristics) across different MARL frameworks. Finally, the assumption of an omniscient manager, while plausible for many applications, is not feasible for some complex sociotechnical systems, especially those where decentralized agents do not all belong to the same legal entity. In such scenarios, the proposed framework needs to be modified, possibly by including several semi-centralized allocation managers who also compete and cooperate with each other at the second tier.

It is also natural to wonder about the scalability of this framework. First, the proposed framework is scalable in principle. Due to the DTDE scheme used in the framework, each agent has its own independent actors and critics, which limits the growth rate of the joint action space, which would otherwise become unsustainable. As a matter of fact, this scalability advantage is one of the main reasons we prefer a two-tiered learning scheme over a single-tiered method. So this work can be scaled up in its current form—using more computational resources—to systems with tens of agents. Scaling up further can, however, pose some challenges, since scalability remains one of the key challenges of the multi-agent reinforcement learning community and an active area of research [33]. Our framework can be scaled further by making use of some of the recently suggested solutions in the literature. As an example, the mean-field approach used in MARL [26] can be scaled up to scenarios with hundreds of agents through a new method of multi-agent learning between each agent and a virtual agent, which represents the mean-field effect of all other agents. Moreover, by learning a few sets of parameters and sharing them with other agents of the same type, parameter sharing in MARL [99] can also mitigate the problem of scalability.

Several types and classes of SoS have been proposed over the past two decades, so one might wonder where the framework might be most helpful. In his widely cited paper [2], Maier distinguishes between different types of SoS: directed SoS in which the component systems maintain an ability to operate independently, but their normal operational mode is subordinated to the centrally managed purpose; and collaborative SoS, where the central management organization does not have coercive power and its control can only be exercised by publishing standards, after which the "component systems" must more or less voluntarily collaborate in order to accomplish the agreed-upon central purposes. By extending the idea of setting standards to establishing dynamic adaptive governance schemes, this paper's framework is applicable naturally to collaborative SoS. The dynamic adaptive governance scheme reduces the need for strict enforcing mechanisms in the directive class of SoS and can assist in balancing the tradeoff between central control and system autonomy.

Finally, the way that the proposed framework can be applied to real problems depend on whether the system relies on online or

offline learning. Online learning can be implemented on a SoS at the operation stage, where the learning happens in real time by experimenting with the environment. In this case, the proposed algorithm can be implemented as described, but instead of interacting with a simulated environment, as was the case in the paper, the RL agent will interact with the real world. This type of implementation is simple, but risky, because RL includes a trial-and-error process, which may cause significant losses—or depending on the applications, catastrophic consequences—when making mistakes in real online system, although some techniques such as transfer learning can be used to reduce the number of trial-and-error and the likelihood of costly mistakes. Alternatively, the framework can be implemented using offline learning [100]. In offline learning, we already have access to a set of historical data of the real SoS system performance for which we want to set up a dynamic resource allocation scheme. In this case, we need to first create a customized simulation environment (a digital twin of the real SoS) that integrates the SoS historical data in its design; train the model using the customized environment; and finally use transfer learning to transfer the learned policy to the real system [21]. Although the two methods of implementing RL-based framework to real applications are promising, efficient and safe implementations of RL methods in real multi-agent systems is an active area of research.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

References

[1] Ackoff, R. L., 1971, "Towards a System of Systems Concepts," Manag. Sci., **17**(11), pp. 661–671.
[2] Maier, M. W., 1998, "Architecting Principles for Systems-of-Systems," Syst. Eng.: J. Int. Council Syst. Eng., **1**(4), pp. 267–284.
[3] Jamshidi, M., 2008, "System of Systems Engineering—New Challenges for the 21st Century," IEEE Aerosp. Electron. Syst. Mag., **23**(5), pp. 4–19.
[4] Caputo, C., and Cardin, M.-A., 2022, "Analyzing Real Options and Flexibility in Engineering Systems Design Using Decision Rules and Deep Reinforcement Learning," ASME J. Mech. Des., **144**(2), p. 021705.
[5] Heydari, B., Mosleh, M., and Dalili, K., 2016, "From Modular to Distributed Open Architectures: A Unified Decision Framework," Syst. Eng., **19**(3), pp. 252–266.
[6] Mosleh, M., Ludlow, P., and Heydari, B., 2016, "Resource Allocation Through Network Architecture in Systems of Systems: A Complex Networks Framework," 2016 Annual IEEE Systems Conference (SysCon), Orlando, FL, Apr. 18–21, IEEE, pp. 1–5.
[7] Raz, A. K., Blasch, E., Cruise, R., and Natarajan, S., 2019, "Enabling Autonomy in Command and Control Via Game-Theoretic Models and Machine Learning With a Systems Perspective," AIAA Scitech 2019 Forum, San Diego, CA, Jan. 7–11, p. 0381.
[8] Jackson, M. O., 2001, "A Crash Course in Implementation Theory," Soc. Choice Welfare, **18**(4), pp. 655–708.
[9] Roughgarden, T., 2010, "Algorithmic Game Theory," Commun. ACM, **53**(7), pp. 78–86.
[10] Nishimura, D. G., 2010, *Principles of Magnetic Resonance Imaging*, Stanford University, Stanford, CA.
[11] Nowak, M. A., 2006, "Five Rules for the Evolution of Cooperation," Science, **314**(5805), pp. 1560–1563.
[12] Gianetto, D. A., and Heydari, B., 2013, "Catalysts of Cooperation in System of Systems: The Role of Diversity and Network Structure," IEEE Syst. J., **9**(1), pp. 303–311.
[13] Xiao, Y., and Sha, Z., 2022, "Robust Design of Complex Socio-Technical Systems Against Seasonal Effects: A Network Motif-Based Approach," Design Sci., **8**(2), p. e2.
[14] Rahwan, I., Cebrian, M., Obradovich, N., Bongard, J., Bonnefon, J.-F., Breazeal, C., Crandall, J. W., Christakis, N. A., Couzin, I. D., Jackson, M. O., and Jennings, N. R., 2019, "Machine Behaviour," Nature, **568**(7753), pp. 477–486.
[15] Heydari, B., and Pennock, M. J., 2018, "Guiding the Behavior of Sociotechnical Systems: The Role of Agent-Based Modeling," Syst. Eng., **21**(3), pp. 210–226.
[16] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., 2017, "Proximal Policy Optimization Algorithms," preprint arXiv:1707.06347.
[17] Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S., 2020, "Meta-World: A Benchmark and Evaluation for Multi-task and Meta Reinforcement Learning," 3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan, Oct. 30–Nov. 1, 2019, PMLR, pp. 1094–1100.
[18] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., and Chen, Y., 2017, "Mastering the Game of Go Without Human Knowledge," Nature, **550**(7676), pp. 354–359.
[19] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., and Petersen, S., 2015, "Human-Level Control Through Deep Reinforcement Learning," Nature, **518**(7540), pp. 529–533.
[20] Lee, T. H., 2003, *The Design of CMOS Radio-Frequency Integrated Circuits.*, Cambridge University Press, Cambridge, UK.
[21] Chen, Q., Heydari, B., and Moghaddam, M., 2021, "Leveraging Task Modularity in Reinforcement Learning for Adaptable Industry 4.0 Automation," ASME J. Mech. Des., **143**(7), p. 071701.
[22] Sukhbaatar, S., Szlam, A., and Fergus, R., 2016, "Learning Multiagent Communication With Backpropagation," Advances in Neural Information Processing Systems 29 (NIPS 2016), Barcelona, Spain, Dec. 5–10.
[23] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I., 2017, "Multi-Agent Actor–Critic for Mixed Cooperative-Competitive Environments," preprint arXiv:1706.02275.
[24] Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S., 2018, "Counterfactual Multi-Agent Policy Gradients," The Thirty-Second AAAI Conference on Artificial Intelligence, Vol. 32, New Orleans, LA, Feb. 2–7.
[25] Jiang, J., and Lu, Z., 2018, "Learning Attentional Communication for Multi-Agent Cooperation," preprint arXiv:1805.07733.
[26] Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., and Wang, J., 2018, "Mean Field Multi-Agent Reinforcement Learning," Proceedings of the 35th International Conference on Machine Learning, Stockholmsmässan, Stockholm, Sweden, July 10–15, PMLR, pp. 5571–5580.
[27] Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y., 2019, "Qtran: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning," Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, June 9–15, PMLR, pp. 5887–5896.
[28] Al-Tam, F., Correia, N., and Rodriguez, J., 2020, "Learn to Schedule (leasch): A Deep Reinforcement Learning Approach for Radio Resource Scheduling in the 5g Mac Layer," IEEE Access, **8**, p. 108088.
[29] Pham, H. X., La, H. M., Feil-Seifer, D., and Nefian, A., 2018, "Cooperative and Distributed Reinforcement Learning of Drones for Field Coverage," preprint arXiv:1803.07250.
[30] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., 2015, "Continuous Control With Deep Reinforcement Learning," preprint arXiv:1509.02971.
[31] Littman, M. L., 1994, "Markov Games As a Framework for Multi-Agent Reinforcement Learning," Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, July 10–13, Elsevier, pp. 157–163.
[32] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K., 2016, "Asynchronous Methods for Deep Reinforcement Learning," Proceedings of The 33rd International Conference on Machine Learning, New York, June 20–22, PMLR, pp. 1928–1937.
[33] Zhang, K., Yang, Z., and Başar, T., 2021, *Handbook of Reinforcement Learning and Control*, Springer Nature, Cham, Switzerland, pp. 321–384.
[34] Winfield, A. F., and Jirotka, M., 2017, "The Case for An Ethical Black Box," Towards Autonomous Robotic Systems 18th Annual Conference, TAROS 2017, Guildford, UK, July 19–21, Springer, pp. 262–273.
[35] Selbst, A. D., Boyd, D., Friedler, S. A., Venkatasubramanian, S., and Vertesi, J., 2019, "Fairness and Abstraction in Sociotechnical Systems," FAT* '19: Conference on Fairness, Accountability, and Transparency, Atlanta, GA, Jan. 29–31, pp. 59–68.
[36] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A., 2021, "A Survey on Bias and Fairness in Machine Learning," ACM Comput. Surv. (CSUR), **54**(6), pp. 1–35.
[37] Mosleh, M., and Heydari, B., 2017, "Fair Topologies: Community Structures and Network Hubs Drive Emergence of Fairness Norms," Sci. Rep., **7**(1), pp. 1–9.
[38] Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., and Yang, G.-Z., 2019, "XAI—explainable Artificial Intelligence," Sci. Rob., **4**(37), p. eaay7120.
[39] Preuer, K., Klambauer, G., Rippmann, F., Hochreiter, S., and Unterthiner, T., 2019, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, Springer Nature, Cham, Switzerland.
[40] Dachowicz, A., Mall, K., Balasubramani, P., Maheshwari, A., Raz, A. K., Panchal, J. H., and DeLaurentis, D. A., 2022, "Mission Engineering and Design Using Real-Time Strategy Games: An Explainable AI Approach," ASME J. Mech. Des., **144**(2), p. 021710.
[41] Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri, S., 2018, "Programmatically Interpretable Reinforcement Learning," Proceedings of the 35th International Conference on Machine Learning, Stockholmsmässan, Stockholm, Sweden, July 10–15, PMLR, pp. 5045–5054.
[42] Madumal, P., Miller, T., Sonenberg, L., and Vetere, F., 2020, "Explainable Reinforcement Learning Through a Causal Lens," The Thirty-Fourth AAAI Conference on Artificial Intelligence, New York, Feb. 7–12, Vol. 34, pp. 2493–2500.

[43] Heuillet, A., Couthouis, F., and Díaz-Rodríguez, N., 2021, "Explainability in Deep Reinforcement Learning," Knowl.-Based Syst., **214**, p. 106685.
[44] Phillips, P. J., Hahn, C. A., Fontana, P. C., Broniatowski, D. A., and Przybocki, M. A., 2021, *Four Principles of Explainable Artificial Intelligence*, National Institute of Standards and Technology, Gaithersburg, MD.
[45] Broniatowski, D. A., 2021, "Psychological Foundations of Explainability and Interpretability in Artificial Intelligence," NIST: National Institute of Standards and Technology, US Department of Commerce.
[46] Lipton, Z. C., 2018, "The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability Is Both Important and Slippery," Queue, **16**(3), pp. 31–57.
[47] Hassannezhad, M., Cantamessa, M., Montagna, F., and Clarkson, P. J., 2019, "Managing Sociotechnical Complexity in Engineering Design Projects," ASME J. Mech. Des., **141**(8), p. 081101.
[48] ElSayed, K. A., Bilionis, I., and Panchal, J. H., 2021, "Evaluating Heuristics in Engineering Design: A Reinforcement Learning Approach," International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 85383, American Society of Mechanical Engineers, Paper No. V03AT03A022.
[49] Rahman, M. H., Yuan, S., Xie, C., and Sha, Z., 2020, "Predicting Human Design Decisions With Deep Recurrent Neural Network Combining Static and Dynamic Data," Design Sci., **6**, p. e15.
[50] Simon, H. A., 1977, *Models of Discovery*, D. Reidel Publishing Company, Dordrecht, The Netherlands, pp. 154–175.
[51] Artinger, F., Petersen, M., Gigerenzer, G., and Weibler, J., 2015, "Heuristics As Adaptive Decision Strategies in Management," J. Organ. Behav., **36**(1), pp. S33–S52.
[52] Meluso, J., and Austin-Breneman, J., 2018, "Gaming the System: An Agent-Based Model of Estimation Strategies and Their Effects on System Performance," ASME J. Mech. Des., **140**(12), p. 121101.
[53] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W., 2016, "Openai gym," preprint arXiv:1606.01540.
[54] Kopetz, H., 2011, *Real-Time Systems*, Prentice Hall PTR, Upper Saddle River, NJ, pp. 307–323.
[55] Alighanbari, M., 2007, "Robust and Decentralized Task Assignment Algorithms for UAVs," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.
[56] Prakasha, P. S., Ratei, P., Naeem, N., Nagel, B., and Bertram, O., 2021, "System of Systems Simulation Driven Urban Air Mobility Vehicle Design," AIAA AVIATION 2021 FORUM, Virtual Event, Aug. 2–6, p. 3200.
[57] Moradian, M., Tabatabaei, F. M., and Moradian, S., 2013, "Modeling, Control & Fault Management of Microgrids," Smart Grid and Renewable Energy, **4**(1), p. 28141.
[58] Saad, W., Han, Z., and Poor, H. V., 2011, "A Game Theoretic Approach for Multi-Hop Power Line Communications," 2nd International ICST Conference, GameNets 2011, Shanghai, China, Apr. 11–18, Springer, pp. 546–561.
[59] Brown, O., Eremenko, P., and Collopy, P., 2009, "Value-Centric Design Methodologies for Fractionated Spacecraft: Progress Summary From Phase I of the Darpa System F6 Program," AIAA SPACE 2009 Conference & Exposition, Pasadena, CA, Sept. 14–17, p. 6540.
[60] Mosleh, M., Dalili, K., and Heydari, B., 2014, "Optimal Modularity for Fractionated Spacecraft: The Case of System F6," Procedia Comput. Sci., **28**, pp. 164–170.
[61] Westkämper, E., 2008, "Manufuture and Sustainable Manufacturing," The 41st CIRP Conference on Manufacturing Systems, Tokyo, Japan, May 26–28, Springer, pp. 11–14.
[62] Mitola, J., and Maguire, G. Q., 1999, "Cognitive Radio: Making Software Radios More Personal," IEEE Pers. Commun., **6**(4), pp. 13–18.
[63] Madni, A. M., Sievers, M. W., Humann, J., Ordoukhanian, E., Boehm, B., and Lucero, S., 2018, *Disciplinary Convergence in Systems Engineering Research*, Springer Nature, Cham, Switzerland.
[64] Dahmann, J., and Henshaw, M., 2016, "Introduction to Systems of Systems Engineering," Insight, **19**(3), pp. 12–16.
[65] Wooldridge, M., 2009, *An Introduction to Multiagent Systems*, John Wiley & Sons Ltd, West Sussex, UK.
[66] Boardman, J., and Sauser, B., 2006, "System of Systems—The Meaning of of," 2006 IEEE/SMC International Conference on System of Systems Engineering, Los Angeles, CA, Apr. 24–26, IEEE, p. 6.
[67] Dahmann, J. S., and Baldwin, K. J., 2008, "Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering," 2008 2nd Annual IEEE Systems Conference, Montreal, QC, Canada, Apr. 7–10, IEEE, pp. 1–7.
[68] Agarwal, S., Pape, L. E., Kilicay-Ergin, N., and Dagli, C. H., 2014, "Multi-Agent Based Architecture for Acknowledged System of Systems," Procedia Comput. Sci., **28**, pp. 1–10.
[69] Jackson, M. O., and Wolinsky, A., 1996, "A Strategic Model of Social and Economic Networks," J. Econ. Theory, **71**(1), pp. 44–74.
[70] Naderializadeh, N., Sydir, J. J., Simsek, M., and Nikopour, H., 2021, "Resource Management in Wireless Networks Via Multi-Agent Deep Reinforcement Learning," IEEE Trans. Wirel. Commun., **20**(6), pp. 3507–3523.
[71] Yan, M., Feng, G., Zhou, J., Sun, Y., and Liang, Y.-C., 2019, "Intelligent Resource Scheduling for 5g Radio Access Network Slicing," IEEE Trans. Veh. Technol., **68**(8), pp. 7691–7703.
[72] Ferreira, P. V. R., Paffenroth, R., Wyglinski, A. M., Hackett, T. M., Bilén, S. G., Reinhart, R. C., and Mortensen, D. J., 2018, "Multiobjective Reinforcement Learning for Cognitive Satellite Communications Using Deep Neural Network Ensembles," IEEE J. Sel. Areas Commun., **36**(5), pp. 1030–1041.
[73] Du, B., Wu, C., and Huang, Z., 2019, "Learning Resource Allocation and Pricing for Cloud Profit Maximization," The Thirty-Third AAAI Conference on Artificial Intelligence, Honolulu, HI, Jan. 27–Feb. 1, Vol. 33, pp. 7570–7577.
[74] Deng, S., Xiang, Z., Zhao, P., Taheri, J., Gao, H., Yin, J., and Zomaya, A. Y., 2020, "Dynamical Resource Allocation in Edge for Trustable Internet-of-Things Systems: A Reinforcement Learning Method," IEEE Trans. Ind. Inform., **16**(9), pp. 6103–6113.
[75] Liu, H., Liu, S., and Zheng, K., 2018, "A Reinforcement Learning-Based Resource Allocation Scheme for Cloud Robotics," IEEE Access, **6**, pp. 17215–17222.
[76] Chinchali, S., Sharma, A., Harrison, J., Elhafsi, A., Kang, D., Pergament, E., Cidon, E., Katti, S., and Pavone, M., 2021, "Network Offloading Policies for Cloud Robotics: A Learning-Based Approach," Auton. Rob., **45**, pp. 997–1012.
[77] Cui, J., Liu, Y., and Nallanathan, A., 2019, "Multi-Agent Reinforcement Learning-Based Resource Allocation for UAV Networks," IEEE Trans. Wirel. Commun., **19**(2), pp. 729–743.
[78] Safavian, S. R., and Landgrebe, D., 1991, "A Survey of Decision Tree Classifier Methodology," IEEE Trans. Syst. Man Cybern., **21**(3), pp. 660–674.
[79] Roth, A. M., Topin, N., Jamshidi, P., and Veloso, M., 2019, "Conservative Q-Improvement: Reinforcement Learning for an Interpretable Decision-Tree Policy," preprint arXiv:1907.01180.
[80] Nageshrao, S., Costa, B., and Filev, D., 2019, "Interpretable Approximation of a Deep Reinforcement Learning Agent As a Set of If–Then Rules," 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, Dec. 16–19, IEEE, pp. 216–221.
[81] Hu, D., 2019, "An Introductory Survey on Attention Mechanisms in NLP Problems," Proceedings of the 2019 Intelligent Systems Conference (IntelliSys), London, UK, Sept. 5–6, Springer, pp. 432–448.
[82] Hafiz, A. M., Parah, S. A., and Bhat, R. U. A., 2021, "Attention Mechanisms and Deep Learning for Machine Vision: A Survey of the State of the Art," preprint arXiv:2106.07550.
[83] Mott, A., Zoran, D., Chrzanowski, M., Wierstra, D., and Rezende, D. J., 2019, "Towards Interpretable Reinforcement Learning Using Attention Augmented Agents," preprint arXiv:1906.02500.
[84] Tang, Y., Nguyen, D., and Ha, D., 2020, "Neuroevolution of Self-Interpretable Agents," GECCO '20: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, Cancún, Mexico, July 8–12, pp. 414–424.
[85] Annasamy, R. M., and Sycara, K., 2019, "Towards Better Interpretability in Deep Q-networks," The Thirty-Third AAAI Conference on Artificial Intelligence, Honolulu, HI, Jan. 27–Feb. 1, Vol. 33, pp. 4561–4569.
[86] Lyu, D., Yang, F., Liu, B., and Gustafson, S., 2019, "SDRL: Interpretable and Data-Efficient Deep Reinforcement Learning Leveraging Symbolic Planning," The Thirty-Third AAAI Conference on Artificial Intelligence, Honolulu, HI, Jan. 27–Feb. 1, Vol. 33, pp. 2970–2977.
[87] Sun, S.-H., Wu, T.-L., and Lim, J. J., 2020, "Program Guided Agent," Eighth International Conference on Learning Representations, Addis Ababa, Ethiopia, Apr. 26–May 1.
[88] Tan, M., 1993, "Multi-Agent Reinforcement Learning: Independent Vs. Cooperative Agents," Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, June 27–29, pp. 330–337.
[89] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., and Vicente, R., 2017, "Multiagent Cooperation and Competition With Deep Reinforcement Learning," PLoS One, **12**(4), p. e0172395.
[90] Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T., 2017, "Multi-Agent Reinforcement Learning in Sequential Social Dilemmas," preprint arXiv:1702.03037.
[91] Lyu, X., Xiao, Y., Daley, B., and Amato, C., 2021, "Contrasting Centralized and Decentralized Critics in Multi-Agent Reinforcement Learning," preprint arXiv:2102.04402.
[92] Lyu, X., Baisero, A., Xiao, Y., and Amato, C., 2022, "A Deeper Understanding of State-Based Critics in Multi-Agent Reinforcement Learning," preprint arXiv:2201.01221.
[93] Shapley, L. S., 1953, "Stochastic Games," Proc. Natl. Acad. Sci. USA, **39**(10), pp. 1095–1100.
[94] Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y., 1999, "Policy Gradient Methods for Reinforcement Learning With Function Approximation," NIPS'99: Proceedings of the 12th International Conference on Neural Information Processing Systems, Denver, CO, Nov. 29–Dec. 4, pp. 1057–1063.
[95] Huang, Q., 2020, "Model-Based Or Model-Free, A Review of Approaches in Reinforcement Learning," 2020 International Conference on Computing and Data Science (CDS), Stanford, CA, Aug. 1–2, IEEE, pp. 219–221.
[96] March, J. G., 1991, "Exploration and Exploitation in Organizational Learning," Organ. Sci., **2**(1), pp. 71–87.
[97] Leonardos, S., Piliouras, G., and Spendlove, K., 2021, "Exploration–exploitation in Multi-Agent Competition: Convergence With Bounded Rationality," NeurIPS 2021: Thirty-fifth Conference on Neural Information Processing Systems, Virtual, Dec. 6–14, Vol. 34, pp. 26318–26331.
[98] Mordatch, I., and Abbeel, P., 2018, "Emergence of Grounded Compositional Language in Multi-Agent Populations," Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, Feb. 2–7, Vol. 32.
[99] Gupta, J. K., Egorov, M., and Kochenderfer, M., 2017, "Cooperative Multi-Agent Control Using Deep Reinforcement Learning," Autonomous Agents and Multiagent Systems AAMAS 2017 Workshops, São Paulo, Brazil, May 8–12, Springer, pp. 66–83.
[100] Levine, S., Kumar, A., Tucker, G., and Fu, J., 2020, "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," preprint arXiv:2005.01643.