Toward multi-target self-organizing pursuit in a partially observable Markov game*

Lijun Sun[a,b], Yu-Cheng Chang[b], Chao Lyu[c,a], Ye Shi[d], Yuhui Shi[a,*], Chin-Teng Lin[b,*]

[a]*Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, China*
[b]*Centre for Artificial Intelligence, CIBCI Lab, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia*
[c]*Harbin Institute of Technology, China*
[d]*School of Information Science and Technology, ShanghaiTech University, China*

Abstract

The multiple-target self-organizing pursuit (SOP) problem has wide applications and has been considered a challenging self-organization game for distributed systems, in which intelligent agents cooperatively pursue multiple dynamic targets with partial observations. This work proposes a framework for decentralized multi-agent systems to improve intelligent agents' search and pursuit capabilities. We model a self-organizing system as a partially observable Markov game (POMG) with the features of decentralization, partial observation, and noncommunication. The proposed distributed algorithm–fuzzy self-organizing cooperative coevolution (FSC2) is then leveraged to resolve the three challenges in multi-target SOP: distributed self-organizing search (SOS), distributed task allocation, and distributed single-target pursuit. FSC2 includes a coordinated multi-agent deep reinforcement learning method that enables homogeneous agents to learn natural SOS patterns. Additionally, we propose a fuzzy-based distributed task allocation method, which locally decomposes multi-target SOP into several single-target pursuit problems. The cooperative coevolution principle is employed to coordinate distributed pursuers for each single-target pursuit problem. Therefore, the uncertainties of inherent partial observation and distributed decision-making in the POMG can be alleviated. The experimental results demonstrate that distributed noncommunicating multi-agent coordination with partial observations in all three subtasks are effective, and 2048 FSC2 agents can perform efficient multi-target SOP with an almost 100% capture rate.

Keywords: multi-target pursuit, noncommunication, observation uncertainty, interaction uncertainty, self-organization

1. Introduction

Self-organizing systems and multi-agent coordination without communication. Self-organization is a type of swarm intelligence that can be found in natural environments and animal behaviors: rippled sand dunes, synchronized flashing fireflies, fish schooling, flocking birds, etc [1]. It forms order and structure through purely internal and local interactions in a system, without any external controls. So, many researchers [2, 3, 4, 5] made efforts to understand the nature and create artificial self-organization systems that can be characterized by decentralization, partial observation, scalability, and emergent properties. However, in general multi-agent game setups, communication failures cannot be avoided due to communication attacks, varying protocols, blocked channels, physical distance, damage, energy conservation, etc. In such scenarios, multi-agent coordination degrades, since no commands, role assignments, conflicts elimination, information sharing, or other negotiations can be exchanged among agents. Therefore, more effective implicit coordination is expected for the more restricted self-organizing setup that does not rely on communication.

Background and application of the pursuit problem. This work investigates the multiple-target self-organizing pursuit (SOP) problem. It formulates general competitive and cooperative interactions among agents and thus can serve as a basic capability of agents in standardized problems and real-world applications. In warfare, agents may be any confrontational devices, such as fighters, bombers, and missiles [6]. In aerospace, one goal is to clean up space debris, inactive satellites, and military vehicles to ensure the safety of active space assets or aerial vehicles [7, 8, 9, 10]. The searchers, pursuers, targets, or evaders in the pursuit may also represent players in a football game, lions and humans in a bounded arena [11], searchers and lost spelunkers in a cave [12], cops and robbers in a city [13, 14, 15], pollutants and cleaning robots in the environment [16], creatures in biological systems [1], etc.

Self-organizing pursuit game setup in comparison with representative MARL pursuits. Comparing existing popular pursuit environments, MPE (multi-agent particle environments) pursuit [17] uses the occupying-based capture, where one pur-

*This work is partially supported by the Shenzhen Fundamental Research Program under Grant No. JCYJ20200109141235597, the National Science Foundation of China under Grant No. 61761136008, the Shenzhen Peacock Plan under Grant No. KQTD2016112514355531, the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant No. 2017ZT07X386, and the Australian Research Council (ARC) under Discovery Grant DP210101093 and DP220100803.

*Corresponding author: Yuhui Shi, Chin-Teng Lin.

Email addresses: Lijun.Sun@student.uts.edu.au (Lijun Sun), Yu-Cheng.Chang@uts.edu.au (Yu-Cheng Chang), 11849557@mail.sustech.edu.cn (Chao Lyu), shiye@shanghaitech.edu.cn (Ye Shi), shiyh@sustech.edu.cn (Yuhui Shi), Chin-Teng.Lin@uts.edu.au (Chin-Teng Lin)

suer occupies the same position of a target or their distance is smaller than a threshold. However, its global observation representation scales poorly with the number of agents. MAgent pursuit (battle) [18] uses the tag-based capture, where a target is attacked if it is tagged by pursuers. Besides, it provides the option to use the global information in the observation or not. The mean field pursuit [19] solves one mass capture problem, where the mass center of the pursuers matches that of the targets is called a capture. Although it can be applied in the large-scale pursuit, the one mass capture is totally different from the multi-target captures that are distributed in the whole space. Moreover, none of these environments consider the interagent collision avoidance problem.

Compared with the above capture definitions, the surrounding-based capture in most literatures is more general in terms of multi-agent behaviors and challenging in terms of coordination, where a target cannot move only if it is surrounded by pursuers. Therefore, this work builds the multi-target self-organizing pursuit (SOP) environment and consider a more practical and challenging multi-agent setting: large-scale partially observable pursuers coordinate without explicit communications, search, chase, and surround multiple distributed dynamic targets until all targets are found and captured without collisions in a grid world. It is worth noting that we consider and report the multi-agent collision avoidance performance, which is a crucial metric concerning the safety in deploying the multi-agent system (MAS) but is rarely reported especially in general MARL literatures.

Related work and MARL coordination solution. Since Isaacs [6], the differential games are used to formulate the pursuit problem, which look for saddle-point strategies and model the dynamics in games with differential equations [7, 8, 9, 10, 20]. However, most such works relate to two-agent zero-sum games. Another seminal work proposed by Benda et al. [21] explores the pursuit problem to investigate the optimal communication structure of agents. In addition to the conventional communication with predefined communication topology, message content, and transmission frequency, selective communications, including dynamic event-triggered communications, are studied [22, 23]. When there are no communications, agents have no ways to get a bigger picture of the world by actively exchanging information. Therefore, it is more challenging to make decision only with agents' own partial observations and the uncertain behaviors of other agents, i.e., the interaction uncertainty [24]. Finally, although many previous works consider the surrounding-based capture, they model the pursuit domain with the Markov decision process (MDP), where each agent can fully observe all agents' positions.

On the other hand, in terms of the partially observable multi-agent settings, many general multi-agent reinforcement learning (MARL) algorithms are tested in the pursuit domain. Coordinated agents can outperform fully independent agents [25]. In particular, the centralized training and decentralized execution (CTDE) is a general framework of coordinated learning for decentralized multi-agent systems. MADDPG [17], COMA [26], and QMIX [27] are representative CTDE MARL algorithms that have centralized action value functions. A common issue of such centralized action-value function learning is that the computational complexity increases with the number of agents involved and the trained agents are heterogeneous, which hinder the large-scale deployment. Besides, since these centralized action value functions are optimized with a fixed number of agents, when the agent swarm size changes, the policy is hard to guarantee its optimality and new training is needed.

In contrast, other CTDE MARL algorithms use the concept of parameter sharing [28] to train shared (value or policy) models from the experiences of all homogeneous agents, which better suit large-scale applications. For example, DGN (graph convolutional reinforcement learning) [29] represents the interactions between agents by the graph network and trains end-to-end by the deep Q-network method. It achieves the cooperation of agents through local communications and interchanges the outputs of its convolutional layers that contain the information of agents within the multi-hop scope. Mean field reinforcement learning [30] tackles large scale multi-agent problems by simplifying the interactions between agents to the interplay between an agent and the mean effect of its neighborhood, i.e., the virtual mean agent, through the mean field approximation. However, by employing the mean field theory, it explicitly ignores the detailed interactions between real agents, which means it cannot deal with the collision avoidance between agents, i.e., the safety RL issues. Besides, most MARL works tested regarding pursuit are subject to several or all of the constraints: small-scale, fully observable pursuers, single-target pursuit, occupying-based capture, with communications, and permitted collisions (see also [31]).

Our work and contribution. Based on the above discussions, the main contributions of this paper are:

- To facilitate the study of implicit multi-agent coordination without communications, this work fills the current literature gap in the self-organizing pursuit (SOP) setup featured by large-scale, decentralization, partial observation, no communication, no interagent collision, multiple distributed targets, and surrounding-based capture. To enable this study, we build the SOP environment [1].

- To address the severer interaction uncertainty [24] and observation uncertainty due to no communications, this work proposes the distributed hierarchical framework called the fuzzy self-organizing cooperative coevolution (FSC2) for the multi-target SOP, as shown in Figure 1. Through analysis, it decomposes the SOP into three sub-problems that can be well formulated and can thus utilize the strengths of fuzzy logic, MARL, and evolutionary computation (EC).

Further, the innovations of the proposed FSC2 framework can be summarized as follows.

- The first module of FSC2: fuzzy based task allocation overcomes the consensus issue of the distributed clustering in two folds by the fuzzy logic with introduced memory. First, to improve the consensus between independent

[1] All code is available at https://github.com/LijunSun90/pursuitFSC2.

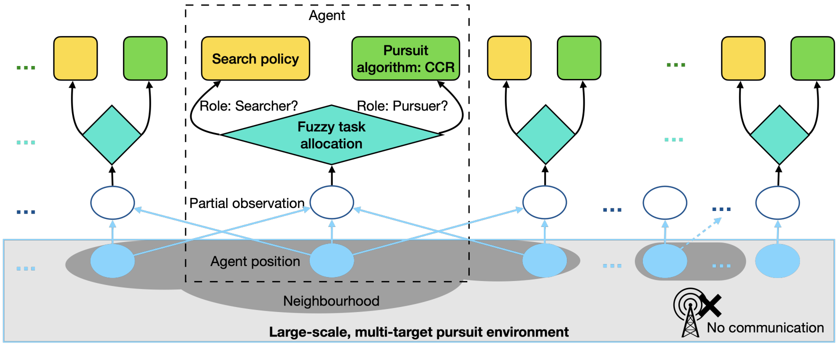


Figure 1: FSC2 (fuzzy self-organizing cooperative coevolution) is a distributed framework for the self-organized pursuit (SOP) that consists of three modules: fuzzy clustering, search policy, and pursuit algorithm–CCR. Note that, since the multi-target pursuit environment is dynamic where all agents and targets are moving, the role of an agent is not fixed and may alternate between a searcher and a pursuer.

agents without communications, we utilize the fuzziness of fuzzy clustering in identifying the cluster memberships of agents. Second, to keep a consistent clustering decision of a single agent in the time scale, we introduce an incremental agent memory in the distributed fuzzy clustering.

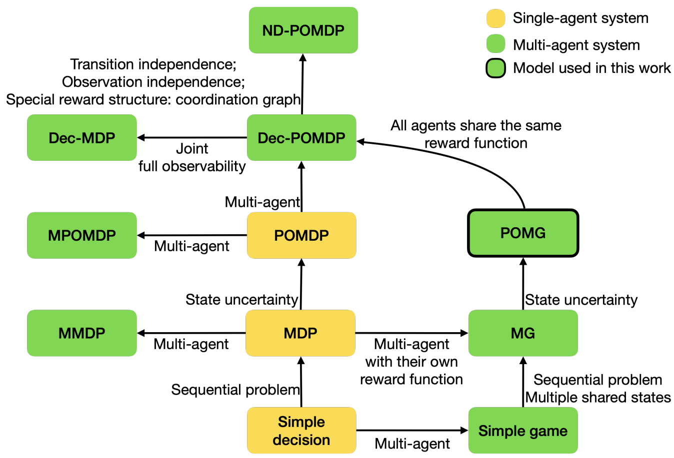
- The second module of FSC2 for search learns reasonable and explainable behaviors for large-scale homogeneous agents by the CTDE actor-critic algorithm. By formulating as the partially observable Markov game (POMG) and designing the reward function, the unknown self-organization mechanism can be learned that maps the local search policy to the coordinated global space exploration without communications.
- The third module of FSC2: pursuit algorithm–CCR proposes a distributed coordination mechanism that can ensure the safety of multi-agent collision avoidance in the target pursuit within clusters. To conquer the partial observation uncertainty and the limit of no communications, the coevolutionary coevolution scheme is used for the online planning in balancing the individual and swarm interests, while the lexicographic convention is adopted for close coordination with the introduced concept of certain partial observation.

The organization of the paper is as follows. First, the problem formulation of self-organizing pursuit is given in Section 2. Second, the proposed approaches are given in Section 3. Third, the experimental results, analyses, and discussions are given in Section 4. Finally, the conclusions, limitations, and future work are given in Section 5.

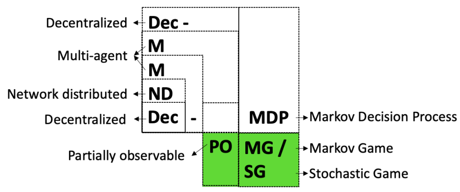
2. Problem formulation

2.1. Multi-agent formulation of self-organization systems

A multi-agent system (MAS) can be seen as a decision-making system in which each agent is a decision maker. It can be formulated in terms of the following four factors: (1) the number of agents: a single agent or multiple agents; (2) state transitions: present (sequential problem) or not; (3) the uncertainty of observability: full observability, joint full observability, or partial observability; and (4) the reward function: each agent has an individual reward function, all agents share the same reward function, or different groups of agents have separate reward functions. Based on the above four dimensions, various models have been proposed and investigated, as shown in Figure 2a [24], and the common nomenclature for the model name abbreviations is presented in Figure 2b.



(a) Common multi-agent problem formulations [24].



(b) Common nomenclature for multi-agent models.

Figure 2: Common multi-agent problem formulations and their nomenclature.

A definition of self-organization is given in [1]: global level patterns unexpectedly emerge solely from the distributed decentralized local nonlinear interactions of components of the system under behavioral rules (of thumb) with local information and no external directing influences. In terms of these features, we can formulate a self-organizing system as a POMG [24]: $\langle \gamma, \mathcal{I}, \mathcal{S}, \mathcal{A}, O, P, O, R \rangle$. γ is the discounted factor for return; $\mathcal{I} = \{1, ..., n\}$ represents all total n agents; $\mathcal{S} = \{s\}$ is the true state space; $\mathcal{A} = \mathcal{A}^1 \times ... \times \mathcal{A}^n = \{\vec{a}\}$ is the joint action space;

$O = O^1 \times ... \times O^n = \{\vec{o}\}$ is the joint observation space; $P(s'|s, \vec{a})$ is the transition function from the current state s to the next state s' given the joint action $\vec{a}$; $O(s) = \{o^1(s), ..., o^n(s)\}$ is the joint observation function; and $R(s, \vec{a}) = \{R^1(s, \vec{a}), ..., R^n(s, \vec{a})\}$ is the joint reward function and each agent maximizes its own accumulated reward.

The reason we use the POMG rather than the Dec-POMDP (decentralized partially observable Markov decision process) to represent a self-organizing system is that in the Dec-POMDP, all agents are fully cooperative in that they aim to maximize a collective reward $R(s, \vec{a})$, while in a general self-organizing system, even collaborative agents have unequal rewards and need to balance the swarm benefits and their own benefits. Therefore, POMG is more similar to the natural swarm intelligence.

2.2. The problem of self-organized search and pursuit

A typical multi-target search and pursuit scenario is illustrated in Figure 3. Due to the partial observation and communication limitation of agents, we distinguish the self-organizing search (SOS) and self-organizing pursuit (SOP) as two different but related multi-agent problems, where the search policy in SOS is taken as a basic capability of agents in the SOP.

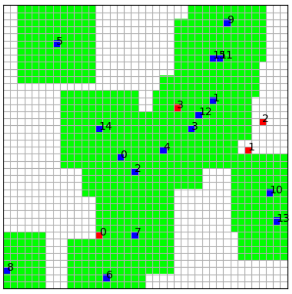


Figure 3: A screenshot of self-organized search and pursuit in a bounded grid world, where red squares are targets or evaders, blue squares are searchers or pursuers, and green background around each agent shows its perception range with an inf-norm radius of 5.

- **Self-organizing search (SOS):** A search is considered successful when a searcher occupies the same position of a target, and the target will then disappear. The SOS terminates when all targets in the environment disappear or the maximum time is reached.
- **Self-organizing pursuit (SOP):** A capture is considered as successful when a target is encircled by four pursuers and cannot move further. However, the target will not disappear after it is captured in the SOP. The game terminates when all targets are found and captured or the maximum time is reached.

Note that, the SOS task is only used to train the search policy in Figure 1, i.e., the space exploring ability that will be used in the SOP task. We design the SOS task harder than the search requirement in the SOP to better train the search policy. First, the SOS task uses multiple static targets since searching for static targets are sometimes harder than dynamic ones in bounded environments, as the agent has no chance to wait for the target coming. In addition, in the SOS task, a target is designed to disappear after being searched to make the search task harder and harder with time, especially when there is no communication and information exchange between agents. Last, in the actual pursuit, agents are not expected to collide with the targets or evaders. However, in the SOS task, we specially define a successful search as that a searcher occupies a target rather than a target appears in the agent's local view, which also only serves the purpose of training. This is because, in the pursuit where the search policy is applied, more than one agents are expected to find and approach the same target simultaneously in order to finally capture it.

In the following, we investigate the coordination strategies for agents constrained by: (1) the observation range of an agent is the scope of radius 5 according to the inf-norm, i.e., an 11×11 square centered at the agent; (2) communication between agents is limited that they can only see the positions of targets and other agents in their own local views, and no other information exchange is allowed; (3) the available movements of all agents are 5 discrete actions {up, down, right, left, still} in the grid world. Therefore, the inf-norm is used in the agent's perception, and the 1-norm (Manhattan distance) is used in the agent's movement, which are widely adopted in MAS.

3. Proposed approach for self-organizing pursuit (SOP)

In this section, we introduce in detail the proposed distributed hierarchical framework–fuzzy self-organizing cooperative coevolution (FSC2) in Figure 1. FSC2 is a distributed algorithm for homogeneous swarm of agents that each agent consists of three modules: (1) fuzzy clustering; (2) search policy; and (3) pursuit algorithm–CCR. Its main idea and motivation is to decompose the distributed self-organizing pursuit (SOP) problem into sub-tasks that are more intuitive and simpler to be well defined and solved.

In the multi-target pursuit environment, targets and partial observable agents are distributed in the space. First, we assume two alternate basic roles of an agent: searcher or pursuer, based on the existence of free targets that are not captured in the agent's neighborhood. Then, agents are distributed clustered that each searcher forms a separate cluster and pursuers are clustered based on their neighborhood relationships. This clustering process is conducted by the first module–fuzzy clustering algorithm in Section 3.1, and Figure 4 gives an illustrative clustering result. After clustering, an agent alternates between the second module: search policy in Section 3.2 and the third module: pursuit algorithm–CCR in Section 3.3, based on its real-time neighborhood. The whole algorithm of FSC2 is given in Algorithm 1, the details of which are introduced in the following sections.

3.1. Distributed fuzzy clustering for task allocation

We define that a pursuer is free if it has not captured a target, while a target is free if it has not been captured. So, an agent is either a searcher, which explores the space to find a free target, or a pursuer, which cooperates with other free pursuers to capture a free target. In the multi-target SOP, since four pursuers

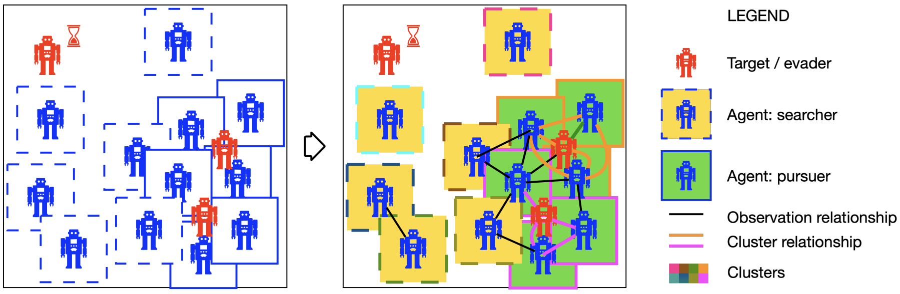


Figure 4: Illustration of the distributed clusters of agents determined by their neighborhood.

Algorithm 1: FSC2 for each agent in the SOP

```
while the termination conditions are not satisfied do
    role, cluster_center, cluster_members, Memory ←
      Fuzzy clustering (Algorithm 2 in Section 3.1).
    if role is a searcher then
        As an SOS agent (Section 3.2), find free targets.
    else if role is a pursuer then
        As a CCR agent (Section 3.3), cooperate with
          cluster_members in pursuing cluster_center.
```

are required to capture each target, distributed task allocation or clustering is needed to determine which group of pursuers capture which free target.

The main challenge in the multi-agent distributed clustering is the consensus issue in two folds due to the partial observation uncertainty and the interaction uncertainty. First, since agents cannot fully observe the world or share the same knowledge through communications, they cannot independently make exactly the same decision. To address this issue, we adopt the fuzzy clustering and utilize its fuzziness in identifying the cluster memberships to reach a consensus with a higher probability. Second, an agent may frequently switch between the roles of searcher and pursuer over a short period of time steps due to its partial observability, which causes instability in the distributed clustering. We, therefore, introduce an incremental agent memory in the fuzzy clustering.

Fuzzy membership. Since the task of the pursuers is to capture targets, for agent k, the cluster centers are all its m^k local free targets $T = \{T_1, ..., T_{m^k}\}$, while the n^k local free pursuers $A = \{A_1, ..., A_{n^k}\}$ need to be clustered, and both T_j and A_i are 2-D positions. The fuzzy membership value of the free pursuer A_i with respect to the cluster center T_j in agent k's view is calculated by

$$\mu_{ij}^k = \frac{(\|A_i - T_j\|_1^2)^{\frac{1}{1-\alpha}}}{\sum_{j=1}^{m^k} (\|A_i - T_j\|_1^2)^{\frac{1}{1-\alpha}}} \in [0, 1], \tag{1}$$

where $\alpha > 1$ is the fuzzifier [32], the value of which is 1.5 in our experiments. Thus, agent k can obtain its fuzzy membership matrix

$$M^k = [\mu_{ij}^k] \in R^{n^k \times m^k}, \tag{2}$$

the i-th row M_{i*}^k of which is the fuzzy membership value of agent i with respect to all local cluster centers in agent k's point of view. Based on M^k, agent k can obtain its membership matrix

$$\hat{M}^k \sim M^k, \tag{3}$$

which is a binary matrix. Its only one element with the value 1 in the i-th row $\hat{M}_{i*}^k$ is sampled from the random distribution determined by M_{i*}^k, since an agent can only belong to one cluster. Based on $\hat{M}^k$, the cluster center of agent k is the target

$$T_c|_{\hat{M}_{kc}^k \neq 0, c=1,...,m^k}, \tag{4}$$

while agent k's cluster members are the pursuers

$$\{A_i | \hat{M}_{ic}^k \neq 0, i = 1, ..., n^k\}. \tag{5}$$

The distributed fuzzy clustering based task allocation process in Equation (1) to (5) is summarized in Algorithm 2.

Agent memory. Note that, each agent's *Memory* of the environment (line 1 of Algorithm 2) is updated through its experiences, which includes the captured status of targets and locked status of pursuers. So, the maximum size of *Memory* is the same for all pursuers, which is determined by the possible number of targets and pursuers in the environment. Without a *Memory*, an agent may oscillate between the roles of a searcher and a pursuer. For instance, an agent may walk one step closer to a target, see the target captured by 4 pursuers, and know that itself is a searcher; if it then walks one step away from the target, the agent can only see 3 pursuers surrounding the target and cannot identify for certain whether it is captured, although it previously observed its captured status. In other cases, a target may be falsely captured such as when it is only blocked by another free target. When that free target walks out of its way, the previous "captured" target becomes free again. In such scenarios, the agent should also update its *Memory* when it is pretty sure based on its newest observation.

In addition, note that, although the number of local clusters is determined by the number of local free targets in Equation (1), the number of members in each cluster is not specified in Equation (5). So, it is possible that more pursuers are clustered into one same nearer target while less pursuers to a farther one. It may be a bit greedy and redundant sometimes that pursuers first cooperate to capture one nearer target as soon as possible and then pursue others. However, this redundancy in the self-organizing clustering may improve the system's robustness to individual robot's software or hardware failures.

Algorithm 2: Distributed fuzzy clustering of agent k

Input : local observation o_t^k of agent k at time t.
Output: $role, cluster\_center, cluster\_members, Memory$.

Update captured targets and locked pursuers in $Memory$.
if *there are no local free or neighboring targets* **then**
  $role \leftarrow$ searcher.
  $cluster\_center \leftarrow$ the agent itself A_k.
  $cluster\_members \leftarrow$ the agent itself A_k.
else
  $role \leftarrow$ pursuer.
  $T = \{T_1, ..., T_{m^k}\} \leftarrow$ local free targets.
  $A = \{A_1, ..., A_{n^k}\} \leftarrow$ local free pursuers.
  $cluster\_center \leftarrow$ Equation (4).
  $cluster\_members \leftarrow$ Equation (5).

Global distributed consistency metric. To evaluate the consistency in the distributed clustering process between the global n agents and m targets, a consistency matrix $C = [c_{ij}] \in R^{n\times n}$ can be calculated from $\{\hat{M}^k | k = 1, ..., n\}$. $c_{ij} \in \{-1, 1, ..., m\}$ is the global target index of the non-zero item of $\hat{M}^i_{j*}$, which represents the cluster (or target) index for agent j from agent i's point of view, and $c_{ij} = -1$ means that agent i has no idea of the cluster of agent j because agent j is located out of the local view of agent i.

The global DC (distributed consistency) can be defined as

$$DC \doteq \frac{2}{n\cdot(n-1)} \sum_{i=1}^{n-1} \sum_{j=i}^{n} \frac{|\{k | k \in \hat{C}_i \cap \hat{C}_j, \text{and } c_{ik} == c_{jk}\}|}{|\hat{C}_i \cap \hat{C}_j|} \in [0,1], \tag{6}$$

where $|\cdot|$ is the the cardinality of a set; $\hat{C}_i = \{k | k = 1, ..., n, \text{and } c_{ik} \neq -1\}$ is the set of visible local pursuers for agent i. The process of computating DC in Equation (6) is to compare every two rows C_{i*} and C_{j*} of C and calculate the ratio of consistent decisions between agent i and agent j in their common knowledge about the other pursuers. Due to this special meaning in our application, we define $0/0 = 1$ for Equation (6), which means that two agents without local physical interactions have fully consistent decisions.

3.2. Search policy in self-organized search (SOS)

In the self-organized search (SOS), a searcher does not have any prior knowledge about the environment or the number of searchers and targets. As in natural self-organization systems, such as a school of fish or a flock of birds, the objective is to equip searchers with the abilities that

(1) a single searcher can perform an effective search by itself when there are no targets or searchers in its local view;

(2) a searcher has a tendency to follow other visible searchers so that a flock of searchers can be formed since the natural flocking behavior can increase the harvesting efficiency, which is especially true with a bigger group [33];

(3) a flock of searchers can perform effective "migration"–like actions rather than tangling with each other so that the flock as a whole loses searching ability.

To achieve these goals, we use the actor-critic algorithm [34] to enable self-organizing searchers to learn from experiences in the centralized training and decentralized execution way.

The parameter θ of policy π_θ is updated with the learning rate α_1 (3×10^{-4} and 10^{-4} in the search and pursuit experiments) according to

$$\theta = \theta + \alpha_1 \nabla_\theta J(\pi_\theta), \tag{7}$$

where

$$\nabla_\theta J(\pi_\theta) = E_{\tau\sim\pi_\theta}[\sum_{t=0}^{tmax} \nabla_\theta \log \pi_\theta(a_t|s_t) A_t], \tag{8}$$

and $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...)$ is the trajectory; A_t is the generalized advantage estimation (GAE) [35] in the form of

$$A_t = \sum_{l=0}^{tmax-t} (\gamma\lambda)^l \delta^V_{t+l}, \tag{9}$$

with γ, λ being two constants (0.99 and 0.97 in our experiments) and

$$\delta^V_t = R(s_t, \vec{a}_t) + \gamma V_\phi(s_{t+1}) - V_\phi(s_t). \tag{10}$$

being the temporal difference (TD) residual of the approximate value function $V_\phi(\cdot)$ with discount γ.

The parameter ϕ of the value function $V_\phi(s_t)$ is optimized by minimizing the following loss function with stochastic gradient descent and learning rate α_2 (10^{-3} and 10^{-4} in the search and pursuit experiments, respectively):

$$\phi = \arg\min_\phi E_{s_t, \hat{R}_t\sim\pi_\theta}[(V_\phi(s_t) - \hat{R}_t)^2]. \tag{11}$$

where $\hat{R}_t = \sum_{t'=t}^{tmax} \gamma^{t'-t} R(s_{t'}, \vec{a}_{t'})$ is the discounted return from point t with reward function $R(s_{t'}, \vec{a}_{t'})$ and discount factor γ.

Reward function. For the SOS task, individual agent's reward function $R^i(s_t, \vec{a}_t)$ in the POMG is given in Table 1. Though simple, experiments show that it achieves satisfied cooperation, and no additional efforts in the multi-agent credit assignment are needed as in the Dec-POMDP formulation.

Table 1: Reward function $R^i(s_t, \vec{a}_t)$ for self-organized search (SOS)

Action	Reward
Search for a target	10 to the contributing agent
Collide with another agent	-12 × # of agents collided with
Collide with an obstacle	Die in its location
Move before termination	-0.05

We once try to give the search reward to the contributing flock, which is a connected component of the graph whose vertexes are agents and edges represent local observations among agents. We assume that if one member agent searches for a target, the whole flock of agents obtain the reward equally to encourage flocking behavior. However, with such a reward mechanism, agents tangle with each other in local regions, although they indeed prefer gathering. Instead, when we simply give a reward only to the contributing agent that finds the target, as in Table 1, the training performance improves significantly.

Note that, the episode reward is defined as the mean of all agents' discounted accumulated rewards in the same environment. In this way, the episode reward score will not increase with the number of agents involved, and thus, the scores are comparable between trials with different numbers of agents.

Parameter sharing based centralized training. In training, agents in the same environment instance maintain a central experience pool and train shared critic and actor models with their newest collective episode experiences. The shared models in different environment instances are coordinated by communicating and averaging their gradients to stabilize the training.

3.3. Cooperative coevolution algorithm for robots (CCR)

According to FSC2 (Algorithm 1), after distributed task allocation, the mission of a free pursuer is to cooperate with other cluster members pursuing the targeting cluster center. For the single-target pursuit, we propose the CCR (cooperative coevolution for robots) algorithm based on CCPSO-R [36, 37], which further improves the cooperation of pursuers in their simultaneously decision making and execution process.

Cooperative coevolutionary evaluation scheme. Similar to CCPSO-R, the real agents in the CCR are the pursuers that execute physical actions in the environment, which can be represented by 2-D positions $\{A_i, i = 1, ..., n\}$. For each real agent A_i, all the neighboring positions one step away from it, including its current position, form a group of virtual agents $\{A_i^1 = A_i, ..., A_i^5\}$ that can act as the candidate next positions for the real agent. The decision-making process of a real pursuer is to evaluate its virtual agents in the cooperative coevolutionary scheme and greedily select the best one as its next position. The pursuit performance is ensured by the evaluation quality of the virtual agents, i.e., how well the fitness function is designed to guarantee conflict-free efficient cooperation in the pursuit.

In particular, the cooperative coevolutionary evaluation scheme means that the fitness evaluation of an individual agent is not only determined by itself, but also on the other real agents. For the target cluster center T_c and pursuer cluster $\{A_1, ..., A_i^j, ..., A_{n^i}\}$, where the i-th member A_i^j is the j-th virtual agent of the i-th real pursuer and n_i is the total number of cluster members, the fitness function f_{stp}^{ij} was proposed in CCPSO-R [36] as follows:

$$f_{stp}^{ij} = f_{closure}^{ij} + f_{expanse}^{ij} + f_{uniformity}^{ij}, \tag{12}$$

where

$$f_{closure}^{ij} = inconv(T_c, A_1, ..., A_i^j, ..., A_{n^i}) \tag{13}$$

evaluates whether the target T_c is located in the convex hull formed by the pursuer cluster: 0 indicates that it is inside, 0.5 indicates that it is on the edge, and 1 indicates that it is outside;

$$f_{expanse}^{ij} = \frac{1}{n^i}\left(\sum_{k=1,k\neq i}^{n^i} \|A_k - T_c\|_1 + \|A_i^j - T_c\|_1\right) \tag{14}$$

gives the spatial extent of the pursuer cluster in terms of T_c; and

$$f_{uniformity}^{ij} = std\left(\begin{bmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{bmatrix}\right) \tag{15}$$

or

$$\begin{aligned} f_{uniformity}^{ij} = \ & std([N_{12}, N_{21}, N_{23}, N_{32}]) \\ & + std([N_{11}, N_{13}, N_{31}, N_{33}]). \end{aligned} \tag{16}$$

evaluates how evenly the pursuer cluster is distributed around T_c based on the standard deviation $std(\cdot)$ where N_{ij} is the number of pursuers in the (i, j)-th space bin (for details, see [36]).

However, f_{stp}^{ij} only solves the cooperative single-target pursuit problem by letting agents make decisions sequentially, while its parallel decision-making version PCCPSO-R [37] can only resolve partial conflicts by introducing two secure distances in the fitness function. Hence, we propose a new fitness function based on f_{stp}^{ij} to enable conflict-free cooperation in single-target pursuit. In detail, the fitness function for the j-th virtual agent of the i-the real pursuer A_i^j can be defined as

$$f^{ij} = \begin{cases} \infty, & \text{if } nnd_{entity}^{ij} == 0 \text{ or} \\ & (nnd_{target}^{ij} \neq 1 \ \& \ nnd_{pursuer}^{ij} == 1) \\ f_{convention}^{ij}, & \text{else if } nnd_{target}^{ij} == 1 \ \& \\ & nnd_{pursuer}^{ij} == 1 \\ f_{stp}^{ij}, & \text{else} \end{cases} \tag{17}$$

where nnd_{entity}^{ij} is the distance to the nearest neighbor with the set *entity*, which could be pursuers, targets or obstacles. In the simultaneous decision-making and execution process, the secure distance between a pursuer and a target is 1 and that between pursuers is 2 to ensure that there are no collisions, and pursuers are not allowed to approach closer than this limit unless they are capturing a target. However, when the condition $(nnd_{target}^{ij} == 1 \ \& \ nnd_{pursuer}^{ij} == 1)$ is satisfied, it means that more than one pursuers may choose to occupy the same capturing position in the next step, where a conflict may occur but can be resolved by the lexicographic convention fitness function $f_{convention}^{ij}$ as follows.

Lexicographic convention. In the proposed lexicographic ordering, 2-D positions are sorted first in the ascending order of their first-dimension values and then based on their second-dimension values, and this is known by all agents. This is used in the lexicographic convention that pursuers coordinate their choices of one-step-away open capturing positions by the following steps.

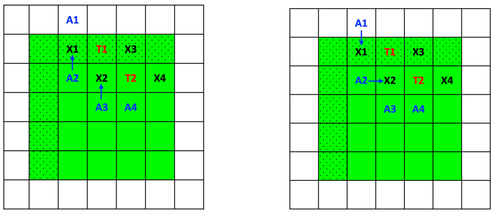
(1) All local open capturing positions are sorted.

(2) All local free pursuers are sorted.

(3) The neighboring open capturing positions and pursuers are paired in the priority order.

If the next candidate position or virtual agent A_i^j of the current real pursuer A_i is its assigned capturing position under a certain partial observation, $f_{convention} = -1$; otherwise, $f_{convention} = \infty$, which means that the choice not satisfying the lexicographic convention is not allowed.

Concept of certain partial observation. The concept of certain partial observation is introduced to ensure multi-agent collision free in the pursuit. It is in contrast to the uncertain partial observation, which is defined as the partial observation that satisfies the following two conditions, as illustrated in Figure 5. First, there exist risky capturing positions, which are the open capture positions on specific boundaries of the local view that will be assigned to a local free pursuer based on the lexicographic convention. Second, there are other free pursuers neighboring the assigned captured position. Under such uncertain observations, an agent may make risky decisions that may lead to collisions. For simplicity, we prevent the current agent from taking the assigned capturing position by setting $f_{convention} = \infty$. Although this may influence the efficiency, it can ensure that there are no collisions in the single-target pursuit due to the observation uncertainty in the POMG.



(a) Incorrect decisions by A3 due to its observation uncertainty.

(b) Actual decisions of pursuers.

Figure 5: Illustration of uncertain partial observation under the lexicographic convention of Section 3.3; collisions may result if such scenarios are not detected. A1, A2, A3, and A4 are the pursuers, T1 and T2 are the targets, X1, X2, X3, and X4 are the open capturing positions, and these entities are numbered in the lexicographic order given in Section 3.3. The green background is the perception range of A3, and the dashed regions are the specific boundaries where risky capturing positions may appear. For A3, X1 is a risky capturing position that is located on the specific boundary of its local view and is assigned to a local free pursuer based on the local lexicographic convention without the detection of such scenarios. Meanwhile, the assigned capturing position X2 of A3 has another neighboring free pursuer A2. The decision of A3, which is made based on uncertain observation satisfying the above two conditions as in (a), may deviate from the actual decisions of pursuers as in (b) and risk collisions.

4. Experiments

4.1. Environment

First, for convenience in comparing the self-organizing search (SOS) agents trained by different MADRL algorithms with their official public code, we made several changes to the PettingZoo Pursuit-V3 environment [38], including the initialization, reward function, some utility functions, and bugs. Second, for the multi-target self-organized pursuit (SOP), we implemented the environment ourselves for more compact code. The local observation $o^i(s)$ of agent i is always represented as an $11 \times 11 \times 3$ binary matrix, where the 3 channels are for targets, agents, and obstacles. All code is available at https://github.com/LijunSun90/pursuitFSC2.

4.2. Experimental setup

Since the proposed FSC2 framework decomposes the multi-target SOP into three subtasks that are solved by different techniques, we have different baselines for different (sub-)tasks and the same algorithm, such as actor-critic, is used and performs differently in different experiments. For clarity, we summarize all experiments and analyses in Table 2.

Note that for the multi-target SOP, we integrate the best performing actor-critic trained self-organizing searcher as the SOS agent in FSC2 (Algorithm 1). All MADRL algorithms use the same policy and value model, which is a two-layer ReLU multi-layer perceptions (MLP) with hidden layers of size 400 and 300. The baselines for the SOS are as follows and use the reward function given in Table 1.

(1) A swarm of independent random-walk searchers: Each searcher randomly walks in the space, taking no account of its surroundings and past history.

(2) A swarm of independent complete searchers: A complete searcher searches the space in a systematic way to ensure that every position on the map is visited at least once; this search is complete so that all targets are guaranteed to be found without a time limit. The optimal systematic search strategy is a solution to the Hamiltonian path problem where every position is visited exactly once; this problem is NP-complete [39]. For simplicity, we employ an intuitive systematic strategy in which the searcher first moves to its nearest map corner and then, starting from that corner, performs zigzag or snakelike walking assuming that the searcher knows the scope of the grid world but does not know the targets' positions. In addition, since the search success is defined as the agent occupying the target's position, the simple systematic searcher is actually equivalent to a searcher with a perception range of 1.

(3) A swarm of coordinated searchers trained by the ApeX-DQN, the current documented best performing model in pursuit [40]: We tested the learning rates $\{10^{-6}$, $\mathbf{10^{-5}}$, 10^{-4}, and $10^{-3}\}$; the batch sizes {128, 256, 512, and **1024**}; the rollout fragment lengths {**32** and 128}; and Adam epsilons {**0.00015**, and 10^{-8}}, where the best values are shown in bold, and the other parameter values are the same as in [40].

(4) A swarm of coordinated MADDPG searchers with communication during training: The OpenAI MADDPG implementation [2] is used in which an agent has access to all other agents' observations and actions through interagent communication; these are used in training the critic function $Q(\vec{o}, \vec{a})$. We tested the learning rates $\{10^{-4}$, $\mathbf{10^{-3}}$, and $10^{-2}\}$; the batch sizes {256, **512**, and 1024}; and the model update rates {4, **100**, and 500}, where the best values are shown in bold.

4.3. Self-organizing search (SOS) experiments

The training performances of the actor-critic, ApeX-DQN, and MADDPG models for 8 agents searching 50 targets in 40×40 grid worlds are shown in Figure 6. The average episode

[2] https://github.com/openai/maddpg

Table 2: Summary of the experiments and analyses in Section 4

Section	Task	Experiment comparison & analysis
4.3	Self-organizing search (SOS)	- Independent random-walk searchers - Independent complete searchers - Swarm of MADDPG searchers - Swarm of ApeX-DQN searchers - Swarm of actor-critic searchers (our approach) - Behavior analysis
4.4	Fuzzy-based distributed task allocation	Distributed task allocation consistency analysis
4.5	Single-target pursuit	- Actor-critic trained pursuer - CCR pursuer (our approach)
4.6	Multi-target self-organizing pursuit (SOP)	Scalability experiments and complexity analysis

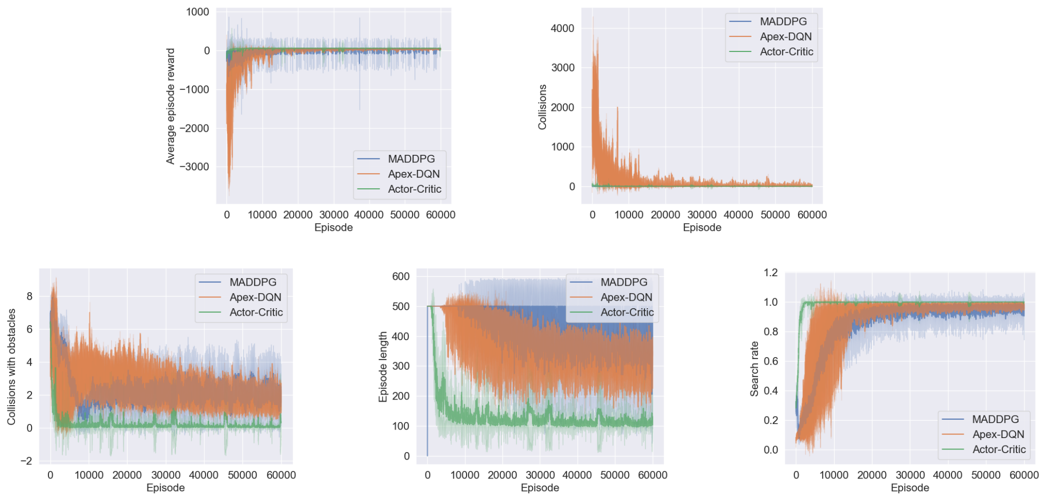


Figure 6: Training performance comparison on self-organizing search (SOS) over 10 random seeds, where the solid lines and shaded areas represent the mean and standard deviation of the corresponding performance, respectively.

reward, episode length, number of collisions between agents, and number of collisions with obstacles all contribute to the reward received by agents as given in Table 1 and thus the agents' training, while the episode search rate is not part of the reward function and is presented to illustrate the effectiveness of the training.

The actor-critic model has the best training performance in terms of convergence speed, the final converged values, and the stability of the training performance. In contrast, both MADDPG and ApeX-DQN are influenced more by the random seeds in the training. MADDPG oscillates severely during the training process. Regarding ApeX-DQN, we observed that the convergence speed is not the most important metric since its performance may degrade and diverge badly with a faster convergence speed. Therefore, we chose the parameters that enable ApeX-DQN's performance to improve steadily, the final performance of which is proven to be better than the best training performance of the parameters with faster convergence that later degrade.

Second, we compare a single agent's searching performance in the 20×20, 40×40, 60×60, and 80×80 grid worlds with 5 targets in Figure 7. With the increase in the environment size and

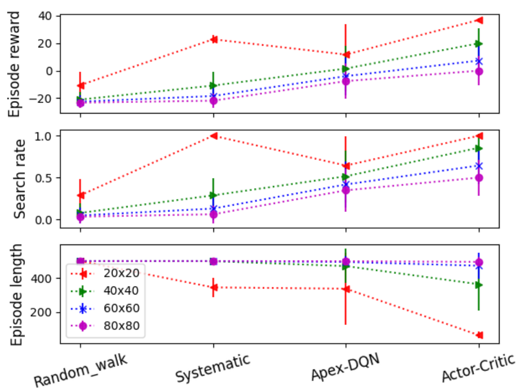


Figure 7: Single SOS agent performance comparison in grid worlds of different sizes, where the mean and standard deviation of the experimental results in 100 independent runs are plotted

the sparsity of targets, the performances of all policies change accordingly, and the actor-critic searcher is always the best. For the random-walk searcher, the environment size has little influence on its performance due to its local random movements, which take longer to explore farther regions. For the systematic searcher, when the environment size is too large to allow it to perform a complete systematic search in a limited time, its performance is slightly better than that of the random-walk searcher. Therefore, compared with a complete searcher, the actor-critic searcher has better performance in searching targets in a limited time in most scenarios.

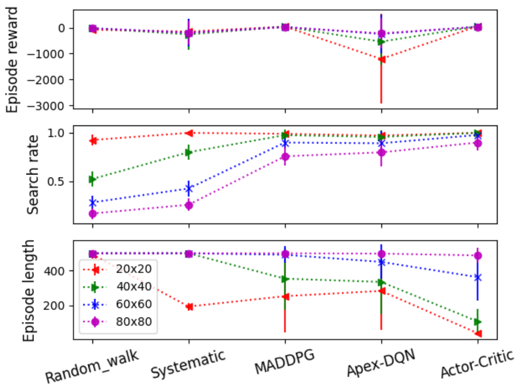


Figure 8: Swarm performance comparison of 8 SOS agents searching 50 targets in grid worlds of different sizes, where the mean and standard deviation of the experimental results in 100 independent runs are plotted

Third, we compare the swarm performance of different policies by searching 50 targets with 8 searchers in $20 \times 20, 40 \times 40, 60 \times 60$, and 80×80 grid worlds, as shown in Figure 8. The smaller the environment is, the larger the swarm density is, and the more challenging the mulit-agent coordination is; and the actor-critic swarm always performs best. Although MADDPG is the algorithm that considers the multi-agent interactions the most in its critic function learning, its performance is not as good as that of actor-critic. In addition, since MADDPG learns a unique critic function for each agent, when the number of agents changes, it needs to relearn.

Finally, the comparison of Figures 7 and 8 proves two facts. First, the superiority of a swarm of independent agents over a single-agent system stems from the benefits of introducing more agents, such as random-walk agents and systematic agents. Second, coordinated inferior agents may sometimes outperform single superior agents in some aspects, such as the swarm of ApeX-DQN agents that outperform the single actor-critic agent.

Behavior interpretation of the SOS policy and sparse targets exploration. One basic problem to be solved in self-organized search is how a searcher behaves when there is no information (no targets and no other searchers) in its current perception, i.e., in the case of an empty observation. To simulate natural flocking, Reynolds [41] proposed three behavioral rules for individual agents: (1) avoid collisions with neighbors; (2) match velocity with neighbors, and (3) stay close to neighbors, which also appear in the three behavior patterns of individual fish models in the movement of a school [33]. However, as indicated in [41], these three behaviors can only support aimless flocking; it is also observed in our experiments that if we only apply these three rules, agents can group together yet become tangled with each other in local regions so that the whole group loses the search ability.

Similar to the case of adding a global direction or global target as the flock's migratory urge in [41], we observe that the successfully trained self-organizing searchers learn similar behaviors by themselves. As shown in Figure 9, we test the actor-critic searcher's behavior by always feeding it with the empty observation, and then estimate the searcher's action distribution over its 5 legal actions by running these tests in 100 independent runs with 1000 steps per run.

It can be seen that although different policies trained with different random seeds have different preferences, the common result is that they prefer a particular action most of the time and stochastically choose other actions. In contrast to the random walk with a uniform action distribution, shown as the red dashed line in Figure 9, this trained action distribution ensures that a searcher will move in one direction most of the time and occasionally switch to another direction, which benefits the target search since the searchers are moving farther away, exploring nonrepeatable areas most of the time, and covering a wide expanse of the map in a limited time. This searching behavior also provides a way to the space exploration problem with sparse targets, as the example shown in Figure 7.

In addition, since the self-organizing searchers are homogeneous, when all searchers perform similar behaviors, as a whole, the self-organizing search swarm behaves as an emergent self-organized pattern. In other words, the self-organized pattern in the self-organizing search emerges here because the agents are homogeneous and behave according to the same meaningful actions.

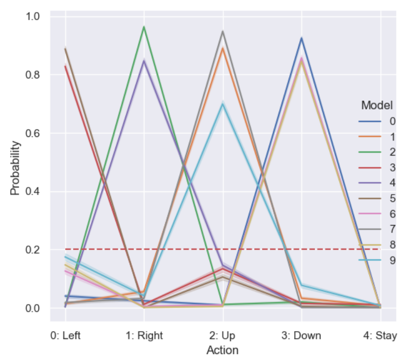


Figure 9: Behavior probability or action distribution of actor-critic trained self-organizing search (SOS) policy with the empty observation, which is estimated from 100 independent runs with 1000 steps per run. The different models are actor-critic policies trained with different random seeds.

4.4. Empirical analysis of the consistency in distributed task allocation

In the distributed task allocation, pursuers and targets are grouped into clusters such that the multi-target SOP is locally decomposed into several single-target pursuit problems. However, in this distributed decision-making process, there may be inconsistency to some extent. As illustrated in Figure 10a, due to the partial observability of pursuers, it is common that an agent can only observe part of another agent's local perception so that they have different knowledge of the world, which is the source of inconsistency in distributed clustering.

For hard clustering, such as k-means, an agent randomly selects one of its nearest targets as its cluster center, while for fuzzy clustering, the choice of targets is determined stochastically by the fuzzy membership matrices. The random choices between the nearest targets in hard clustering and fuzzy membership values in the fuzzy clustering may all stochastically result in different consistency matrices C. We multiply the DC value of each matrix C with its corresponding probability and obtain the stochastic DC value. Figure 10a gives an example scenario in which fuzzy clustering is stochastically superior to hard clustering. Such scenarios occur when the uncertainty outside of the common observation area brings better options for the agents, such as T_2 to A_2 in Figure 10a. In contrast, as illustrated in Figure 10b, fuzzy clustering is stochastically inferior to hard clustering when the uncertainty outside of the common observation area fails to provide better options for the agents, such as T_3 to A_2, and when there is no any uncertainty.

However, since uncertainty is inherent in the partially observable game, an agent can never determine the level of uncertainty from only its own local view without other related information communicated between neighboring agents. In addition, what is important here is that with fuzzy clustering, in scenarios where fuzzy clustering is stochastically inferior to hard clustering, its stochastic process enables it to be as good as or even better than hard clustering. In contrast, with hard clustering, in scenarios where hard clustering is not stochastically superior to fuzzy clustering, its clustering result will never beat the fuzzy clustering result. Therefore, fuzzy clustering reduces the influence of uncertainty in distributed task allocation in partially observable environments, especially in cases without interagent communication.

4.5. Single-target pursuit (STP) experiments

In the literature, it is not uncommon to train fully observable agents in pursuing a single target with the surrounding-based capture definition. Therefore, in this section, we train single-target pursuers with the actor-critic (AC) algorithm and the reward function in Table 3. All training is conducted in 6×6 grid worlds so that the agents' 11×11 perception range can cover the whole space and the partially observable agents can fully perceive the environment. Then we compare the STP performance between the trained AC pursuers and the proposed CCR pursuers in Table 4.

Table 3: Reward function $R^i(s_t, \vec{a}_t)$ for single-target pursuit (STP)

Action	Reward
Capture a target	5 to the contributing agent
Neighbor a target	0.1 to the contributing agent
Collide with an agent	-0.2 $\times$ # of agents collided with
Collide with an obstacle	Stay in place for one time step
Move before termination	-0.05

Table 4: Single-target pursuit performance comparison between the actor-critic (AC) trained pursuer and the CCR pursuer in 6*x*6 grid worlds. The values are the mean and standard deviation of the results from 100 independent runs.

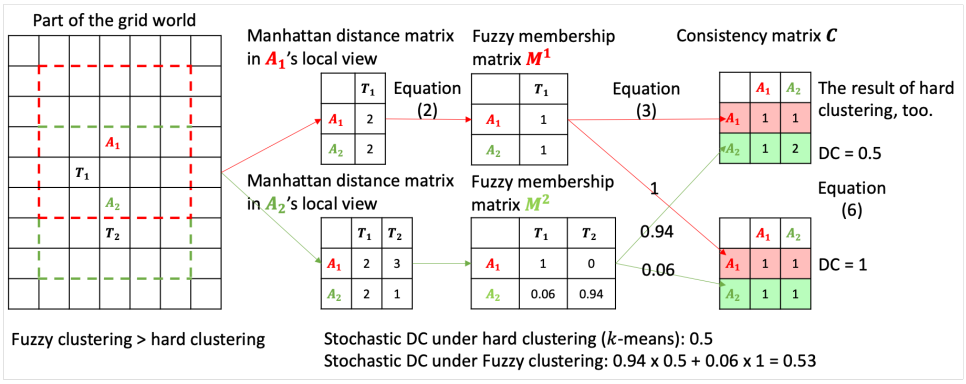
Algorithm	Capture rate	Episode length	Collisions
AC	0.96 (0.196)	30.21 (100.339)	20.07 (97.457)
CCR	**1.0** (**0.0**)	**5.21** (**1.971**)	**0.0** (**0.0**)

It can be seen that although AC pursuers can learn to capture the target very quickly with a high capture rate and low collisions, collisions are difficult to be absolutely avoided in the simultaneous decision-making and simultaneous action execution of the POMG. There is no chance of canceling, retracting, or coordinating a wrong decision, which leads to collisions since all agents' actions are performed simultaneously after their simultaneously made decisions. This is different from the scheme of simultaneous decision-making and sequential action execution in some multi-agent games where collisions can be resolved by, for example, the priorities of agents before the decisions are actually executed. In contrast, CCR pursuers are more efficient and reliable, as they simultaneously utilize the cooperation coevolution mechanism to evaluate individual decisions with respect to the swarm benefits and utilize the lexicographic convention to resolve possible collisions in execution.

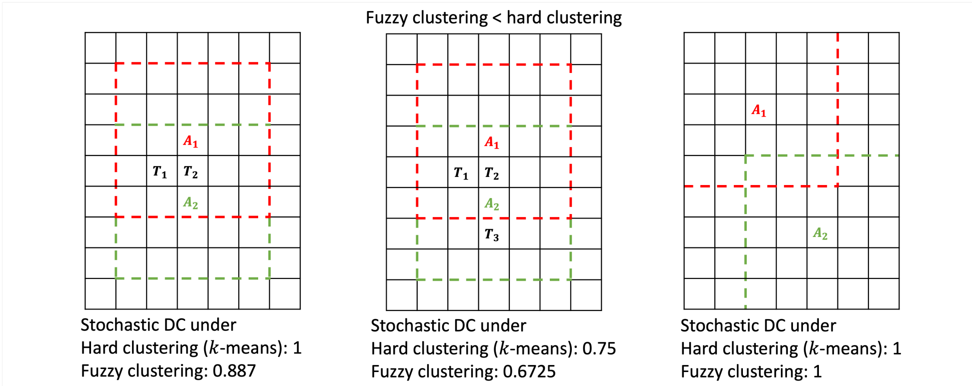
4.6. Self-organizing pursuit (SOP) experiments

We test the swarm performance and scalability of up to 2048 FSC2 agents in multi-target SOP in 40×40 and 80×80 grid worlds, as shown in Figures 11 and 12, respectively. Almost all experiments achieve a nearly 100% average capture rate except that the number of pursuers is too small to cover the space in the maximum of 500 time steps, such as in the cases of 4 and 8 pursuers in 80×80 grid worlds in Figure 12. However, the more than 68% average capture rate proves the efficient search ability of FSC2 agents in such trials.

Note that, the collisions in 0.22% and 2.1% of the trials in Figures 11 and 12 all occur when the FSC2 agent is a searcher, i.e., the SOS agent in Algorithm 1. This does not mean a performance degradation of SOS agents in SOP tasks. Rather, it reveals the weak safety guarantee of RL algorithms. Figure 13 gives two consecutive frames showing an inter-agent collision when 128 targets and 512 pursuers are deployed in the 40×40 grid world. Although SOS agents learn to interact with each other in the multi-agent environment and the collisions are reduced significantly, it cannot be avoided absolutely. In the search subtask, SOS agents are only trained in very simple environments where boundary walls are the only obstacles. By deploying SOS agents in the multi-target SOP, however, they are often surrounded by increasingly complex distribution of captured targets and locked pursuers that are equivalent to obstacles, and the environment is more like a complicated maze. Besides, compared with the collision avoidance with static obstacles, the multi-agent collision avoidance is a more complicated coordination problem that is harder to be fully guaranteed



(a) Illustrative scenarios: fuzzy clustering is stochastically superior to hard clustering. Note that, in M^2, the membership value of A_1 to T_2 is 0 because A_2 can infer that T_2 is outside the perception scope of A_1 as all agents are homogeneous and have the same perception radius.



(b) Illustrative scenarios: fuzzy clustering is stochastically inferior to hard clustering.

Figure 10: The computational process of DC in Equation (6) and stochastic comparisons between fuzzy clustering and hard clustering in distributed task allocation, where the symbols ">" and "<" represent stochastically superior and inferior, respectively, and a dashed rectangle around an agent of the same color indicates its local perception scope with an inf-norm radius of 2 for the purpose of illustration.

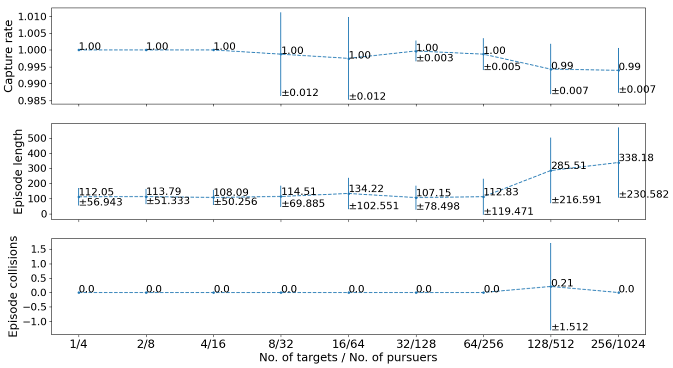


Figure 11: Swarm performance in the multi-target self-organizing pursuit (SOP) in 40×40 grid wolds with different numbers of targets and pursuers, where the mean and standard deviation of the experimental results in 100 independent runs are plotted.

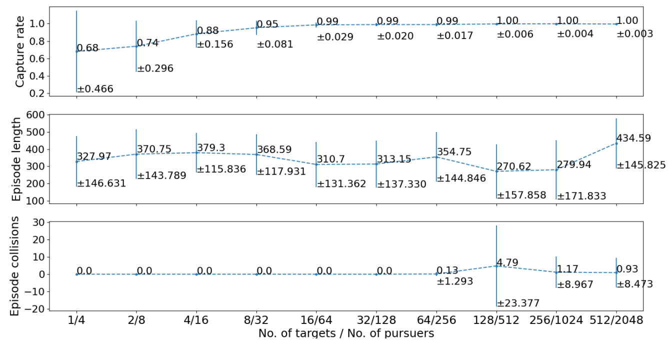


Figure 12: Swarm performance in the multi-target self-organizing pursuit (SOP) in 80×80 grid wolds with different numbers of targets and pursuers, where the mean and standard deviation of the experimental results in 100 independent runs are plotted.

by RL. In such scenarios, FSC2 agents can still capture nearly 100% of the targets within the limit of 500 time steps without collisions most of the time, which can also be seen from the large standard deviation of the nonzero mean collisions in Figures 11 and 12.

In addition, the relatively stable swarm performance of FSC2 agents indicates that the three proposed subsolutions in FSC2, i.e., the MADRL-trained self-organizing search (SOS) agents, fuzzy-based distributed task allocation, and the CCR-based single-target pursuit, all fulfill their responsibilities effectively and efficiently, which also indicates the good scalability of FSC2 agents. Due to the fully distributed nature of the proposed self-organizing algorithm FSC2, its application and per-

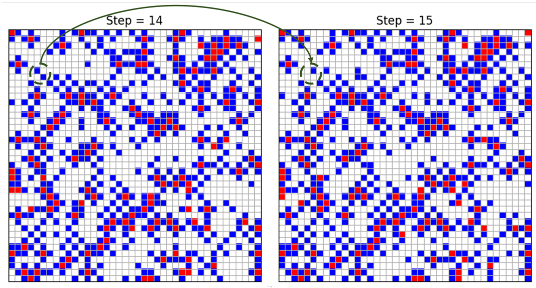


Figure 13: Multi-agent collision scenario illustration in the multi-target SOP with 128 targets (red squares) and 512 agents (blue squares) in 40×40 grid world: the two circled agents in step 14 are two searchers that collide with each other in step 15.

formance are not limited by the swarm size.

4.7. Discussion

Computational complexity analysis. For a distributed partially observable agent without communication, the computational complexity is not related to the swarm size but only related to the observation range. Assume that there are n pursuers and m targets in the local observation defined by the range r, where $n + m \leq r^2$, and let $c_i, i = 1, 2, ...$ be some constants. First, for the distributed task allocation in Section 3.1, the time complexity in terms of Equations (2) to (5) is $(c_1 \cdot n \cdot m + c_3 \cdot m) + c_3 \cdot n \cdot m + c_4 \cdot m + c_5 \cdot n = O(n \cdot m)$. Second, for the SOS in Section 3.2, the time complexity of the policy model with input size $3r^2$ is $O(r^2)$. Third, for the single-target pursuit in Section 3.3, the time complexity [3] of Equation (12) is $O(nlogn) + c_1 \cdot n + c_2 \cdot (m + n) = O(nlogn)$, while the time complexity of calculating the lexicographic convention in Equation (17) is $O(n^2) + O(m^2) + O(n \cdot m) = O(max(n, m)^2)$ in the worst case. Therefore, based on Algorithm 1, FSC2's time complexity is $O(max(n, m, r)^2)$ in the worst case.

Generalization of FSC2 and comparison with existing work. As introduced in Section 1, there are many capture definitions in the pursuit domain. The proposed FSC2 algorithm can be extended to other multi-agent pursuit games, although it is originally proposed for the 4-pursuer-surrounding-based capture. For example, FSC2 satisfies the mass capture based pursuit in [19]. In FSC2, when pursuers surround the target or even before the target cannot move, the mass center of pursuers will match that of the target. But instead of the mass center of the group including all pursuers matching that of the evader group and thus one mass capture in [19], four pursuers take charge of each target and thus there are many distributed mass captures in the FSC2. Therefore, compared with Zhou et al. [19], FSC2 is more suitable for the pursuit where pursuers and targets are spatially distributed. In particular, FSC2 can additionally deal with the interagent collision avoidance. On the other hand, FSC2 can directly solve the pursuit problems with one more time step if the capture is occupying-based and the number of pursuers needed for a target is not greater than 4, as in MPE [17]. FSC2 agents only need to walk towards the target one more step after they surround the target and the target cannot move. Actually, not only the occupying-based pursuit, pursuers can do many things as long as the target is surrounded, such as tagging the target as in MAgent [18]. In the proposed fuzzy-based distributed task allocation, we do not limit the number of agents in a cluster to greedily capture one visible target with as many pursuers as possible. This is beneficial when applying the FSC2 in other pursuit problems under the occupying-based capture yet with more than 4 pursuers for each target. In addition, the fitness function, i.e., Equation (17), of the CCR algorithm is originally designed to suit the capture with more than 4 pursuers, as shown in its sequential decision-making version: CCPSO-R [36]. The only necessary modifications are the capture definition and the order of agents in which they walk toward the target to ensure that there are no collisions.

[3] http://www.qhull.org/html/qh-code.htm#performance

5. Conclusion

This work investigated the large-scale multi-target SOP problem by formulating it as a POMG and proposed the distributed algorithm FSC2 based on the fuzzy logic, MARL, and evolutionary computation. It does not rely on interagent communication and is thus naturally robust to unavoidable communication failures in general multi-agent game setups. In particular, FSC2 decomposes the SOP into three well formulated sub-problems. First, for the distributed task allocation, the influence of interaction uncertainty and partial observation uncertainty on the consistent distributed clustering is reduced. Second, in the search, a single SOS agent has superior performance in searching for targets within a limited time, and a swarm of SOS agents performs best in terms of multi-agent interactions. Third, for the target pursuit, CCR can ensure effective target pursuing with the safety guarantee in the inter-agent collision-avoidance. Finally, we prove the effectiveness of FSC2 in the SOP experiments with more than 2048 agents.

However, the safety of interagent collision avoidance is difficult to be guaranteed by MADRL without explicit communications. This is also the reason that we only apply MARL in the search sub-task, not the target pursuit task, the latter of which needs more close coordination and challenges the RL methods more. In future work, more complex self-organizing patterns are expected to emerge that are not simply due to homogeneous agents, and the distributed implicit multi-agent coordination problem needs to be further investigated.

References

[1] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraula, E. Bonabeau, Self-organization in biological systems, Princeton university press, 2001.

[2] M. Rubenstein, A. Cornejo, R. Nagpal, Programmable self-assembly in a thousand-robot swarm, Science 345 (6198) (2014) 795–799. doi:10.1126/science.1254295.

[3] F. Berlinger, M. Gauci, R. Nagpal, Implicit coordination for 3d underwater collective behaviors in a fish-inspired robot swarm, Science Robotics 6 (50). doi:10.1126/scirobotics.abd8668.

[4] S. Warnat-Herresthal, H. Schultze, K. L. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz, et al., Swarm learning for decentralized and confidential clinical machine learning, Nature 594 (7862) (2021) 265–270. doi:https://doi.org/10.1038/s41586-021-03583-3.

[5] D. Ye, M. Zhang, A. V. Vasilakos, A survey of self-organization mechanisms in multiagent systems, IEEE Transactions on Systems, Man, and Cybernetics: Systems 47 (3) (2017) 441–461. doi:10.1109/TSMC.2015.2504350.

[6] R. Isaacs, Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization, New York: John Wiley and Sons, 1965.

[7] D. Ye, M. Shi, Z. Sun, Satellite proximate pursuit-evasion game with different thrust configurations, Aerospace Science and Technology 99 (2020) 105715. doi:https://doi.org/10.1016/j.ast.2020.105715.

[8] Y. Guan, D. Maity, C. M. Kroninger, P. Tsiotras, Bounded-rational pursuit-evasion games, in: 2021 American Control Conference (ACC), 2021, pp. 3216–3221. doi:10.23919/ACC50511.2021.9483152.

[9] Z. Wang, B. Gong, Y. Yuan, X. Ding, Incomplete information pursuit-evasion game control for a space non-cooperative target, Aerospace 8 (8). doi:10.3390/aerospace8080211.

[10] X. Tang, D. Ye, L. Huang, Z. Sun, J. Sun, Pursuit-evasion game switching strategies for spacecraft with incomplete-information, Aerospace Science and Technology 119 (2021) 107112. doi:https://doi.org/10.1016/j.ast.2021.107112.

[11] J. E. Littlewood, A mathematician's miscellany, Methuen & Co. Ltd., London, 1953.

[12] T. D. Parsons, Pursuit-evasion in a graph, in: Theory and Applications of Graphs, Springer Berlin Heidelberg, Berlin, Heidelberg, 1978, pp. 426–441.

[13] R. Nowakowski, P. Winkler, Vertex-to-vertex pursuit in a graph, Discrete Mathematics 43 (2) (1983) 235 – 239. doi:https://doi.org/10.1016/0012-365X(83)90160-7.

[14] F. V. Fomin, P. A. Golovach, J. Kratochvíl, On tractability of cops and robbers game, in: G. Ausiello, J. Karhumäki, G. Mauri, L. Ong (Eds.), Fifth Ifip International Conference On Theoretical Computer Science – Tcs 2008, Springer US, Boston, MA, 2008, pp. 171–185.

[15] A. Bonato, The game of cops and robbers on graphs, American Mathematical Soc., 2011.

[16] T. H. Chung, G. A. Hollinger, V. Isler, Search and pursuit-evasion in mobile robotics, Autonomous robots 31 (4) (2011) 299. doi:10.1007/s10514-011-9241-4.

[17] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, Neural Information Processing Systems (NIPS).

[18] L. Zheng, J. Yang, H. Cai, M. Zhou, W. Zhang, J. Wang, Y. Yu, Magent: A many-agent reinforcement learning platform for artificial collective intelligence, Proceedings of the AAAI Conference on Artificial Intelligence 32 (1).

[19] Z. Zhou, H. Xu, Decentralized optimal large scale multi-player pursuit-evasion strategies: A mean field game approach with reinforcement learning, Neurocomputingdoi:https://doi.org/10.1016/j.neucom.2021.01.141.

[20] I. E. Weintraub, M. Pachter, E. Garcia, An introduction to pursuit-evasion differential games, in: 2020 American Control Conference (ACC), 2020, pp. 1049–1066. doi:10.23919/ACC45564.2020.9147205.

[21] M. Benda, V. Jagannathan, R. Dodhiawala, On optimal cooperation of knowledge sources-an empirical investigation, Tech. rep., BCS-G2010-28, Boeing Advanced Technology Center, Boeing Computing Services, Seattle, Washington (1986).

[22] Y. Wang, L. Dong, C. Sun, Cooperative control for multi-player pursuit-evasion games with reinforcement learning, Neurocomputing 412 (2020) 101–114. doi:https://doi.org/10.1016/j.neucom.2020.06.031.

[23] M. Lv, B. D. Schutter, S. Baldi, Non-recursive control for formation-containment of hfv swarms with dynamic event-triggered communication, IEEE Transactions on Industrial Informatics (2022) 1–1doi:10.1109/TII.2022.3163573.

[24] K. H. W. Mykel J. Kochenderfer, Tim A. Wheeler, Algorithms for Decision Making, MIT Press, 2022.
URL https://algorithmsbook.com/

[25] M. Tan, Multi-agent reinforcement learning: Independent vs. cooperative agents, in: Proceedings of the tenth international conference on machine learning, 1993, pp. 330–337.

[26] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, S. Whiteson, Counterfactual multi-agent policy gradients, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 32, 2018.

[27] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, S. Whiteson, Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 4295–4304.

[28] J. K. Gupta, M. Egorov, M. Kochenderfer, Cooperative multi-agent control using deep reinforcement learning, in: G. Sukthankar, J. A. Rodriguez-Aguilar (Eds.), Autonomous Agents and Multiagent Systems, Springer International Publishing, Cham, 2017, pp. 66–83. doi:https://doi.org/10.1007/978-3-319-71682-4_5.

[29] J. Jiang, C. Dun, T. Huang, Z. Lu, Graph convolutional reinforcement learning, in: International Conference on Learning Representations, 2019.

[30] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, J. Wang, Mean field multi-agent reinforcement learning, in: International conference on machine learning, PMLR, 2018, pp. 5571–5580.

[31] C. de Souza, R. Newbury, A. Cosgun, P. Castillo, B. Vidolov, D. Kulić, Decentralized multi-agent pursuit using deep reinforcement learning, IEEE Robotics and Automation Letters 6 (3) (2021) 4552–4559. doi:10.1109/LRA.2021.3068952.

[32] J. C. Bezdek, Pattern recognition with fuzzy objective function algorithms, Springer, Boston, MA, 2013. doi:https://doi.org/10.1007/978-1-4757-0450-1.

[33] T. Pitcher, A. Magurran, I. Winfield, Fish in larger shoals find food faster, Behavioral Ecology and Sociobiology 10 (2) (1982) 149–151. doi:https://doi.org/10.1007/BF00300175.

[34] V. Konda, J. Tsitsiklis, Actor-critic algorithms, Advances in neural information processing systems 12.

[35] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, arXiv preprint arXiv:1506.02438.

[36] L. Sun, C. Lyu, Y. Shi, Cooperative coevolution of real predator robots and virtual robots in the pursuit domain, Applied Soft Computing 89 (2020) 106098. doi:https://doi.org/10.1016/j.asoc.2020.106098.

[37] L. Sun, C. Lyu, Y. Shi, C.-T. Lin, Multiple-preys pursuit based on bi-quadratic assignment problem, in: 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 1585–1592. doi:10.1109/CEC45853.2021.9504823.

[38] J. K. Terry, B. Black, M. Jayakumar, A. Hari, R. Sullivan, L. Santos, C. Dieffendahl, N. L. Williams, Y. Lokesh, C. Horsch, et al., Pettingzoo: Gym for multi-agent reinforcement learning, arXiv preprint arXiv:2009.14471.

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

[40] J. K. Terry, N. Grammel, A. Hari, L. Santos, B. Black, Revisiting parameter sharing in multi-agent deep reinforcement learning, arXiv preprint arXiv:2005.13625.

[41] C. W. Reynolds, Flocks, herds and schools: A distributed behavioral model, in: Proceedings of the 14th annual conference on Computer graphics and interactive techniques, 1987, pp. 25–34. doi:10.1145/37401.37406.