UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Multi-agent Coordination Algorithms for Pursuit-Evasion

by Lijun Sun

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of
Professor Chin-Teng Lin,
Professor Yuhui Shi

University of Technology Sydney
Faculty of Engineering and Information Technology

July 2023

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Lijun Sun, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Computer Science, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior to publication.

Date: July, 2023

ACKNOWLEDGMENTS

I would like to sincerely thank my two principle supervisors: Professor Chin-Teng Lin and Professor Yuhui Shi. Thank you so much for granting me this opportunity and financial support. I greatly appreciate your continuous patience, encouragements, and insightful ideas for my research work. In these four years, I received numerous help and learned a great many from two professors, which are essential to be a good independent researcher.

I would like to thank all CIBCI (Computational Intelligence and Brain Computer Interface Lab) members. In particular, thank Dr Yu-Kai Wang, who spent time and gave me valuable suggestions. Thank Dr Yu-Cheng Chang for discussing research problems and working out solutions with me. I really appreciate all your help. Thank Dr Jie Yang for being nice and help me a lot on many issues.

I would like to specially thank my boyfriend and work partner, Dr Chao Lyu. Thank you so much for accompanying me and supporting me. We experienced all happiness and difficulties together. We dreamed, worked, and helped each other. It is you that allows me to be more confident, more powerful, and happier, regardless of the circumstance.

Last, I would like to give my gratitude to my family members, my parents, my sister, my sister's husband, and my nephew. Thank you so much for being so considerate, which allowed me to study without worries.

Dedication

To my loved ones

Abstract

The multi-agent coordination or swarm intelligence is a paramount concern in multi-agent systems (MAS) that determines the exclusive advantage over single-agent systems. Although diverse swarm robots tasks are achieved and complex multi-agent strategies emerge, the real-world application of MAS is still challenging and limited, such as in the large-scale warehouse robots, autonomy traffic, and swarm drones. Among diverse MAS benchmarks, the pursuit-evasion game is a popular, general, and representative one that models practical coordination demands and has attracted sustained research efforts. Therefore, based on the pursuit-evasion variants, this research investigates the following five coordination aspects and proposes corresponding solutions.

First, the safe multi-agent coordination problem is investigated. Popular multi-agent benchmarks provide limited safety support for the safe multi-agent reinforcement learning (MARL) research, where negative reward for collisions cannot guarantee the safety. Therefore, this research proposes a new safety-constrained multi-agent environment: MatrixWorld, based on the general pursuit-evasion game. In particular, the multi-agent safety constraints are implemented by three classification ways of pursuit-evasion games: the multi-agent-environment interaction model, the collision resolution mechanism in multi-agent action execution model, and the game termination condition. Besides, MatrixWorld is a lightweight co-evolution framework for the learning of pursuit tasks, evasion tasks, or both, where more pursuit-evasion variants can be designed based on different practical meanings of safety.

Second, the NP-hard distributed coordination problem is investigated throughout our research. For example, in the fully observable pursuit of a single evader, this research proposes the cooperative co-evolutionary particle swarm optimization algorithm for robots (CCPSO-R). It introduces the concept of virtual agents and utilizes the cooperative co-evolutionary evaluation mechanism for the decentralized cooperation of on-line planning pursuers. Experiments are conducted on a scalable swarm of pursuers with 4 types of evaders, the results of which show the reliability, generality, and scalability of the proposed CCPSO-R. Comparison with a representative dynamic path planning based algorithm Multi-Agent Real-Time Pursuit (MAPS) further shows the effectiveness of CCPSO-R.

Third, the NP-complete multi-agent task allocation problem is investigated in the pursuit-evasion variants with more than one evaders. For example, in the fully observable pursuit of multiple evaders, this research proposes the two-stage approach: BiPCCR, which solves in a dynamic optimization way. In particular, a multi-evader pursuit (MEP)

fitness function is proposed for the involved bi-quadratic assignment problem (BiQAP), which significantly reduces the search cost. Besides, based on the domain knowledge, one BiQAP solver is improved to work better statistically. In this work, the safety of CCPSO-R algorithm is enhanced in the proposed PCCPSO-R algorithm for the simultaneous multi-agent decision-making and action execution.

Fourth, the multi-agent observation uncertainty and interaction uncertainty are investigated in the partial observable pursuit-evasion variants. Further, to avoid the coordination performance degradation due to communication failures and be immune from the communication cost, a more restricted self-organizing setup with only implicit coordination is considered. To address the above challenges, this research proposes a distributed hierarchical framework called the fuzzy self-organizing cooperative co-evolution (FSC2) algorithm. The experimental results demonstrate that by decomposing the task by FSC2, superior performance are achieved compared with other implicit coordination policies fully trained by general MARL algorithms. The scalability of FSC2 is proved that up to 2048 FSC2 agents perform efficiently with almost 100% capture rates. Empirical analyses and ablation studies verify the interpretability, rationality, and effectiveness of component algorithms in FSC2.

Last, open problems and magics in the autocurriculum learning are explored in the co-evolutionary pursuit-evasion variants. To better understand related research works and more accurately use similar terminologies, this research reviews and analyzes the co-evolution mechanism in the multi-agent setting, which clearly reveals its relationships with autocurricula, self-play, arms races, and adversarial learning. Then, through adversarial learning, this research achieves various arms race outcomes of different co-evolution mechanisms. Based on experiments, arms races with steady and converging improvement are more practical for increasingly complex behaviors, while policy cycles between two rival sides are useful for producing diverse policies. In particular, this research finds that the passive (evasive) policy learning benefits more from co-evolution than active (pursuing) policy learning in an asymmetric adversarial game.

List of Publications

Related to the Thesis :

1. Lijun Sun, Chao Lyu, and Yuhui Shi. Cooperative coevolution of real predator robots and virtual robots in the pursuit domain. Applied Soft Computing 89 (2020): 106098, doi: 10.1016/j.asoc.2020.106098. (This paper was generated from Chapter 3.)

2. Lijun Sun, Chao Lyu, Yuhui Shi, and Chin-Teng Lin. Multiple-preys pursuit based on biquadratic assignment problem. In 2021 IEEE Congress on Evolutionary Computation (CEC), pp. 1585-1592. IEEE, 2021, doi: 10.1109/CEC45853.2021.9504823. (This paper was generated from Chapter 4.)

3. Lijun Sun, Yu-Cheng Chang, Chao Lyu, Ye Shi, Yuhui Shi, and Chin-Teng Lin. Toward multi-target self-organizing pursuit in a partially observable Markov game. Information Sciences 648 (2023): 119475. doi: 10.1016/j.ins.2023.119475. (This paper was generated from Chapter 5.)

4. Lijun Sun, Yu-Cheng Chang, Chao Lyu, Yuhui Shi, and Chin-Teng Lin. MatrixWorld: a pursuit-evasion platform for safe multi-agent coordination and autocurriculum. arXiv preprint arXiv:2307.14854 (2023). (This paper was generated from Chapter 2 and 6.)

Others :

5. Lyu, Chao, Yuhui Shi, and Lijun Sun. A novel local community detection method using evolutionary computation. IEEE Transactions on Cybernetics 51, no. 6 (2019): 3348-3360.

6. Lyu, Chao, Yuhui Shi, and Lijun Sun. A novel multi-task optimization algorithm based on the brainstorming process. IEEE Access 8 (2020): 217134-217149.

7. Lyu, Chao, Yuhui Shi, and Lijun Sun. Community detection based on surrogate network. In International Conference on Bio-Inspired Computing: Theories and Applications, pp. 45-53. Singapore: Springer Singapore, 2021.

8. Lyu, Chao, Yuhui Shi, Lijun Sun, and Chin-Teng Lin. Community detection in multiplex networks based on evolutionary multi-task optimization and evolutionary clustering ensemble. IEEE Transactions on Evolutionary Computation (2022).

9. Lyu, Chao, Yuhui Shi, and Lijun Sun. "Data-driven evolutionary multi-task optimization for problems with complex solution spaces." Information Sciences 626 (2023): 805-820.

TABLE OF CONTENTS

List of Figures

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Multi-agent pursuit-evasion

Inspiration from biological pursuit-evasion. Pursuit-evasion or predator-prey is a common natural phenomenon where predators pursue their preys for dinner, while preys evade predators for life, and their co-evolutionary arms race and co-adaptations never stop. For example, the coordination of predators increases their hunting efficiency and success, such as encircle the prey and feed on it one-by-one [19]. Coordinated defensive and evasive behaviors of preys better detect, confuse, reduce exposure to, and evade from predators, such as the aggregation behaviors of birds and fish.

Reasons to investigate pursuit-evasion in AI. Similar to nature, in the artificial intelligence (AI), the pursuit-evasion (PE) is a general multi-agent coordination problem that achieves extensive attention for several reasons. First, the PE is an adversarial game between two swarms of agents, i.e., pursuers and evaders. It involves both cooperative and competitive interactions that are general in multi-agent systems (MAS). Second, the pursuit and evasive behaviors are general for many multi-agent tasks. For example, as indicated by Miller et al. [53, 54], by interpreting the pursuer and evader differently, the evasive pursuer avoidance can be applied in collision avoidance and dispersion, the pursuit behavior can be used in the goal-directed navigation, robot arm grasping, foraging, following, and aggregation, while the adversarial co-evolution between pursuers and evaders is a testbed for the investigation of adaptive behaviors and robust policies. Third, PE has wide real-world applications, such as in the warfare [39], aerospace

[33, 87, 94, 97], football game, search-and-rescue [65], cops-and-robbers [13, 29, 59], and pollutants and cleaning [21].

Challenges in the pursuit-evasion research. Compared with the single-agent system, the multi-agent system is more attractive and challenging that it is beyond the simple summation of multiple independent agents. The coordination and thus the swarm intelligence is the key for its success. Typically, a single-agent system can be modeled as a Markov decision process (MDP) with the Markov assumption that the current state of the environment and action of the agent can determine the next state without conditioning on historical states or actions. When multiple agents are involved in the multi-agent system (MAS), a general extension of the MDP is the Markov game (MG) or stochastic game (SG). Further, when the full observation of an agent is constrained due to the state uncertainty, the MAS is formulated as a partially observable Markov game (POMG). The MG generalizes the multi-agent MDP (MMDP) in terms that all agents in the MG have their own utility or reward functions, while the same utility function is shared by all agents in the MMDP. Similarly, the POMG generalizes the decentralized partially observable MDP (Dec-POMDP). While the complexity of MDP and MMDP are P-complete and EXPTIME [3, 63, 76], both the decentralized MDP (Dec-MDP) and Dec-POMDP are NEXP-complete [11], and MG and POMG are PPAD-complete without known polynomial time algorithms [56].

Challenge 1: safe multi-agent reinforcement learning (MARL). MARL has achieved encouraging performance in solving complex multi-agent tasks, some of which are human-level. However, the safety of MARL policies is one critical concern that impedes their real-world applications. Furthermore, popular multi-agent benchmarks provide limited safety support for safe MARL research, where negative rewards for collisions are insufficient. Therefore, this thesis proposes a new safety-constrained multi-agent environment: MatrixWorld, based on the general pursuit-evasion game. Besides, research on co-evolution is still open and it is supposed to be promising for automatically generating more complex agent behaviors and intelligence from the arms races in multi-agent interactions. The proposed MatrixWorld is a lightweight co-evolution framework for the learning of pursuit tasks, evasion tasks, or both, where more pursuit-evasion variants can be designed based on different practical meanings of safety.

Challenge 2: distributed coordination. Decentralized decision-making algorithms are preferred than centralized solutions, due to the scalability issue of the multi-agent coordination problems. However, as proved by Bernstein et al. [11], the complexity is fundamentally different between the centralized and decentralized control of Markov de-

cision process (MDP). Even the decentralized decision-making problem with two agents is NP-complete. This problem is investigated throughout this work. For example, in the fully observable pursuit of a single evader where more than one pursuers are needed for a successful capture of the evader, most previous work cannot simultaneously assure the 100% reliable capture rate, good scalability to problem size, and generality to evader types. Therefore, aiming at the scalable and general swarm intelligence, this research introduces the concept of virtual agents and utilizes the cooperative co-evolutionary evaluation mechanism for the cooperation in the decentralized on-line planning of pursuers.

Challenge 3: multi-agent task allocation. In the fully observable pursuit of multiple evaders, two task allocation problems exist. One is to assign each evader to a specific group of pursuers for pursuing. The other is to assign each capturing position of an evader to a specific pursuer for capturing. The first problem is a nonlinear assignment problem that is proved to be NP-complete [18, 64], while the concurrent mutual collision avoidance in the second problem is another challenge for the safe multi-agent coordination. Therefore, this research formulates this game as a dynamic optimization problem and proposes a two-stage approach to solve these two problems, wherein the inherent properties of the game and coordination algorithm with safety concerns are explored.

Challenge 4: partial observability. In the partially observable pursuit of multiple evaders, general decentralized coordination algorithms rely on inter-agent communications for issuing commands, assigning roles, eliminating conflicts, sharing information, negotiating, etc. Therefore, such algorithms are typically fragile to communication failures. However, reliable communication may be unavailable due to communication attacks, incompatible protocols like human-robot interactions, limited channels, physical distance, damage, or energy conservation. With the constraints of partial observation and noncommunication, the interaction uncertainty considering the independent agents' behaviors will be severer. As a result, the distributed task allocation of assigning evaders to groups of pursuers becomes more challenging. Furthermore, due to the same reason, an subtask of multi-agent search is introduced that evaders in the environment need to be first found before they can be pursued and captured by pursuers. It is also challenging since pursuers cannot share their search experience with each other and no global consistent belief map can be built among them. Besides, since the evaders are moving, pursuers need to repeatedly search the space while keeping coordination so that they are like a multi-agent system and better than multiple independent and non-coordinated agents. Therefore, this research proposes to alleviate these problems by developing more effective implicit coordination algorithm that does not rely on communication.

Challenge 5: useful arms race and autocurriculum learning. The adversarial pursuit-evasion task is more like a co-evolution framework where both pursuers and evaders co-evolve, learn, and get improved together. In 2019, the concept of autocurriculum was proposed by Leibo et al. [46] to specially refer the automatic curriculum leanring (CL) in multi-agent settings, where the sequence of tasks or challenges is self-generated from the mutual counter-adaptations in multi-agent interactions, i.e., an arms race. However, based on both biological observations [23] and artificial intelligence (AI) investigations [46], it is noted that challenges do not definitely mean more difficult or complex tasks, and adaptations to these challenges do not necessarily involve increasingly strong abilities. Even the commonly adopted difficulty ranking of tasks, i.e., from easy to hard, in curriculum learning is not guaranteed to be optimal in all cases [93]. The conditions for obtaining different arms race outcomes have attracted researchers' attention. Particularly, people are more interested in generating stronger models, such as agents with more complex behaviors. Therefore, this research revisits the co-evolution mechanism in the multi-agent setting, discusses its relationships with several similar concepts: self-play, autocurricula, arms races, and adversarial learning, and explores the magics therein and practical way to produce useful or preferable arms race results.

1.2 Literature review

Timeline of pioneer works in pursuit-evasion. The pursuit-evasion domain is a classic and popular benchmark problem in the multi-agent system (MAS). In the past decades, there are several pioneer works in this filed. In 1953, according to Littlewood [48], the lion-and-man game was invented by R. Rado where a lion wants to capture the evading man in a bounded circular arena. In 1965, Isaacs [39] first used the differential game to formulate the two-agent zero-sum pursuit-evasion problem. In 1978, Parsons [65] studied the pursuit-evasion on graphs from the application that searchers need to find a lost spelunker in a cave whose behaviors are unpredictable and even adversarial in the worst case, and claimed that the problem was raised by Richard Breisch. In 1978 and 1983, according to Bonato [13], the cops-and-robbers game was first investigated by Quilliot [68] and independently investigated by Nowakowski and Winkler [59]. As the cops-and-robbers game is also called the vertex-pursuit game, it is played on a reflexive graph where cops and robbers take turn to observe, make decision, and move until the cops occupy the same vertexes of the robbers, i.e., the capture, or the time is run out. In 1985, Benda et al. [9] used the pursuit problem in grid worlds to investigate the optimal

organization structure and communication mode of agents for their optimal cooperation. This pursuit game in grid worlds is most related to our study here.

Categories of works in pursuit-evasion. The literature work in the pursuit-evasion domain can be broadly classified into four categories. (1) In the first category, the works, such as Benda et al. [9], Stephens [78], and Osawa [61], investigated the pursuit problem by correlating the organization structure of agents with the intelligence of the system. They showed that the organization structure of agents is crucial but not by itself. The next two categories focus more on the multi-agent coordination strategies design. (2) The works in the second category implemented the implicit coordination with a group of independent agents, such as Stephens et al. [79] and Haynes et al. [35, 37, 38]. Without considering partnerships and multi-agent interactions in the strategy, i.e., no perception of other partners, nor communications, agents may fail to cope with conflict scenarios and thus the final cooperation performance are not satisfied. (3) In contrast, works in the third category designed coordination strategies for cooperative agents. Tan [86] specially compared the performance of cooperative agents with that of independent agents. The results showed that for both normal tasks that can be accomplished by a single agent and joint tasks that need more than one agents, perception cooperation help improve the efficiency in accomplishing tasks compared with a group of independent agents. Coordinated learning, such as sharing policy model parameters or experiences, can speed up learning. More successive work tried to answer the questions like what cooperative methods to use or how to be cooperative, such as the game theoretic techniques used by Levy et al. [47], the case learning [36], the path planning methods [91], the reinforcement learning (RL) approaches [4, 26, 34, 40], and the evolutionary computation (EC) [57]. In addition, representation learning was also investigated in this category, such as the ad-hoc teaming works [5–8]. Their goal was to enable the cooperation with different, even previously unseen or unknown teammates. Therefore, their focus is different that models or representations of other agents are the core learning task. (4) Finally, in the fourth category, adaptations and arms race mechanisms were explored in the adversarial pursuit-evasion games. Its study has been started since 1994 [51, 53, 54, 71], which is slightly later than the research on pursuit or evasion alone since 1953 [48]. As a general co-evolution framework, its related works include co-evolution, self-play, autocurricula, arms races, adversarial learning, etc. Our study here belongs to the last two categories, designs the multi-agent coordination strategies, and explores arms races in the pursuit-evasion.

Single evader pursuit with full observation. In the pursuit of a single evader,

Korf [44] manually designed greedy coordination strategies. Haynes et al. [35–38] used the genetic programming (GP) [45], strongly typed genetic programming (STGP) [55], and cases learning methods to improve the pursuit performance. However, the capture rate cannot always reach 100%. Undeger and Polat [91] proposed the multi-agent real-time pursuit (MAPS) algorithm by formulating the multi-agent dynamic pursuing problem as a dynamic path planning and task allocation problem. Ishiwaka et al. [40] investigated the emerging mechanism of predators' cooperative behaviors in the continuous world through Q-learning. Barrett et al. [4, 7] improved the ad hoc teaming performance in the pursuit by learning the models of teammates. Stone et al.[80] gave a detailed survey on the pursuit domain.

Multiple evaders pursuit with full observation. Compared with the single evader pursuit game, a new task allocation problem to assign each evader to each four pursuers is introduced in the multiple evaders pursuit domain. This is a nonlinear assignment problem called the bi-quadratic assignment problem (BiQAP). First, Burkard et al. [18] introduced the BiQAP arising from the very large scale integration (VLSI) synthesis, which is also the only one published application of BiQAP except this research. It is proved that BiQAP is NP-complete [18, 64]. Therefore, heuristic methods are designed in the literature. Burkard et al. [17] proposed several approaches to BiQAP, which includes three deterministic improvement methods, i.e., the first improvement method (FIRST), the best improvement method (BEST), and the Heider's improvement method (HEIDER); three simulated annealing algorithms (SIMANNs); one taboo search method; and one hybrid of taboo search and SIMANN. Afterward, Mavridou et al. [52] proposed the greedy randomized adaptive search procedure (GRASP) algorithm that integrates the FIRST algorithm as the local search method to the greedily constructed initial solution in each of its iteration. The comparison of these algorithms on the BiQAP benchmark problems [18] showed that HEIDER achieves the best balance between the solution quality and the computational cost among the deterministic methods, while only GRASP finds all optimal solutions on all test instances but consuming more computations [17, 20, 52]. However, for the multi-agent task allocation problem, the best algorithm like GRASP may be not the most preferable choice due to the real-time requirement in the multi-agent game, while the simpler solver with good performance is more favorable.

Multiple evaders pursuit with partial observation. 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 [86]. In particular, the centralized training and decentralized execu-

tion (CTDE) is a general framework of coordinated learning for decentralized multi-agent systems. One effective way of implementing CTDE is to apply the concept of parameter sharing [34], which enables the extension of single-agent reinforcement learning (RL) algorithms to the multi-agent setting, such as the actor-critic algorithm [85]. It is extremely useful for the learning of large-scale homogeneous agents by training shared (value or policy) models from collective experiences. Besides, it also benefits the coordinated learning efficiency as shown in the experiments of [90, 98].

To enhance the cooperation of agents, many CTDE MARL algorithms use the centralized (action) value functions, such as MADDPG [49], COMA [28], QMIX [69], and MAPPO [98]. 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. For example, in the MADDPG, each agent separately maintains a centralized critic function that takes the joint observation and joint action as inputs, while agents' actor functions use only local information. In contrast, although MAPPO also learns a centralized value function that accesses the global information out of the partial observations of the agents, it uses the parameter sharing for homogeneous agents and thus potential for large-scale applications. Mean field reinforcement learning [96] also uses the CTDE framework and 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. Another way to enhance the cooperation is to allow communications. For example, DGN (graph convolutional reinforcement learning) [42] achieves the cooperation of agents through local communications to interchange the intermediate outputs of agent models. As for other MARL works tested regarding pursuit, they are mostly subject to several or all of the constraints: small-scale, fully observable pursuers, single-target pursuit, occupation-based capture, with communications, and permitted collisions (see also [24]).

Adversarial pursuit-evasion. Reynolds [71] co-evolved a population of agents in the one-versus-one role-reversal tag game, a symmetric pursuit and evasion. Sub-optimal strategies emerge automatically through the competitive co-evolution. Miller et al. [54]

and its two extended work [51, 53] discussed the generality of pursuit-evasion to the research of adaptive behaviors in biology, machine learning, and robot tasks. They co-evolved the pursuit and evasion behavioral tactics and morphologies in both the asymmetric and symmetric one-versus-one pursuit-evasion games. Nolfi et al. [58] generally discussed the relationship between co-evolution and the auto-curriculum learning. In particular, the evolutionary arms race between pursuit and evasion strategies was investigated in the one-versus-one scenarios based on Khepera robots. They demonstrated that the cycling problem of strategies caused by sudden behavior pattern changes can be reduced and more stable co-evolutionary process can be produced by evaluating individuals in the current generation against sampled competitors from all previous generations. They also investigated the influence of sensory abilities to the emergence of arms races in progressively producing more general solutions. Zheng et al. [100] proposed a co-evolutionary pursuit benchmark problem for MARL. Zhou et al. [101] transformed the large-scale pursuit-evasion game to a two mass players differential game and find their Nash equilibrium. Besides, in terms of adversarial game benchmarks, more complex adversarial games were investigated in the literature, such as capture the flag (CTF) [41], and StarCraft II learning environment (SC2LE) [97].

Multi-agent environment for safe policy and safe multi-agent reinforcement learning. In thriving multi-agent reinforcement learning (MARL) research, increasingly complex multi-agent behaviors have emerged in increasingly complicated environments. However, in current practice, MARL is still weak and challenging in resolving safety guarantees in multi-agent coordination scenarios [60, 99], such as autonomous driving [95]. To date, the safe MARL has not gone beyond toy domains such as grid worlds. Therefore, this thesis revisits the significance of the discrete pursuit-evasion game and introduces MatrixWorld, a new safety-constrained multi-agent pursuit-evasion platform.

Here, the safety concept is defined in terms of collision avoidance. For instance, collisions may stem from conflicts of interest during the multi-agent cooperation, the resolution of which is so important for determining the essential contribution of a coordination algorithm [14, 15]. When collisions occur, the multi-agent software environment should deal with these conflicts correctly by presenting reasonable and practical collision results and blaming the responsible party through e.g., rewards, based on which algorithms can properly learn. However, as indicated by Terry et al. [88], the real collision resolution mechanism implemented by MARL environments may be biased and lead to unexpected outcomes for the stochastic game (SG), where the multi-agent actions should

be executed simultaneously. This may result from the bugs and complexities of codes when resolving the race conditions.

Popular multi-agent grid-world benchmarks provide limited safety support, based on which general MARL studies rarely report their safety performances even when they truly matter in associated applications. For example, in the Pursuit and Battle tasks of MAgent [100], rewards are given in terms of the attacking action. However, what if two agents (of the same team) intend to go to the same position? The collision resolution mechanism is transparent without any reward feedback. Similar situations are also applicable to the predator-prey task of multi-agent particle environments (MPE) [49], the Pursuit task of PettingZoo [88], etc. In these cases, any biased or impractical collision resolution mechanisms with transparent information to the coordination algorithm will impede its safety learning. Nevertheless, in practice and as shown in our experiments, negative rewards for collisions are insufficient for guaranteeing the safety of MARL policies, where significant efforts are expected in open safe MARL research.

Another safety limitation of popular MARL grid-world environments is that their collision outcomes are not diverse enough to allow algorithms to learn various safety requirements in applications. This is because they focus more on providing diverse multi-agent behavior scenarios than diverse safety circumstances. For example, in the same pursuit-evasion framework, if agents are drones, any collisions will lead both to crashes (death and disappearance). However, if pursuers are predators and evaders are prey, predators will survive and prey will die after their collisions since they are unequal. Furthermore, if pursuers are mobile agents while evaders are stationary targets, pursuers and evaders can collide while collisions between pursuers are not allowed, as in the multi-agent path finding (MAPF) problem. These various safety definitions may bring different inequalities to the learning processes of agents and thus their resultant behaviors.

1.3 Research overview

This research investigates the multi-agent coordination algorithms for problems and challenges in the pursuit-evasion domain, the research map of which is shown in Figure 1.1. Some coordination sub-tasks are common and fundamental to all pursuit-evasion games and are thus investigated in all coordination algorithms, such as the distributed pursuit problem. Algorithms proposed for more constrained pursuit-evasion games can solve less constrained ones better, for example, the FSC2 algorithm can better solve all

previous games. The main research work and contributions of this thesis can be stated by answering the following five questions.

Problem 1: how to design the multi-agent environment for safe multi-agent coordination? In Chapter 2, this research considers the design and software implementation of multi-agent environments for safe multi-agent coordination research and autocurriculum learning. Accordingly, a new safety-constrained multi-agent environment: MatrixWorld is proposed based on the general pursuit-evasion game. Compared with the limited safety support by popular multi-agent benchmarks, the multi-agent safety constraints are implemented by three classification ways of the pursuit-evasion game: the multi-agent-environment interaction model, the collision resolution mechanism in multi-agent action execution model, and the game termination condition. Particularly, this research shows how to cover diverse safety definitions in various applications through our design.

Problem 2: how to design a distributed coordination algorithm? The distributed pursuit coordination problem in the fully observable single dynamic evader pursuit game is investigated in Chapter 3. Different from previous works, the pursuit domain is formulated as a dynamic optimization problem and a cooperative co-evolutionary particle swarm optimization (CCPSO-R) algorithm is proposed, which is distributed and scalable. Unlike previous EA methods and RL algorithms that incrementally construct the solution or learn the policy with a training stage, CCPSO-R is an on-line algorithm that plans one step ahead under the immediate guidance of the fitness function. Moreover, different from the robotic PSO (RPSO) [22], the PSO variant specially designed for robots, CCPSO-R separates the multi-agent collision avoidance mechanism from the EA itself by integrating it into the modular fitness function design. In particular, real and virtual agents coexist in CCPSO-R. Virtual agents provide the action space for the real agents by sampling and selectively covering each real agent's vicinity under the guidance of PSO [77]. So, CCPSO-R is more efficient and effective compared with an exhausted exploration of the vicinity.

Problem 3: how to solve the multi-agent task allocation problem? The coordination problem of multi-agent task allocation is explored in the pursuit game for multiple dynamic evaders in Chapter 4. Similar to the above work, the multiple evaders pursuit problem is formulated as a dynamic optimization problem and a two-stage approach: BiPCCR is proposed to solve the new static optimization problem in each time step. In the first stage, to decompose the multiple evaders pursuit (MEP) problem to multiple single evader pursuit (SEP) problems, the biquadratic assignment problem (BiQAP) is used to

model the task allocation between each evader and each four pursuers, which extends the application of BiQAP to the multi-agent system (MAS). To evaluate the multi-tasks assignment, a multiple evaders pursing fitness function is developed based on the single evader pursuing function proposed in [83]. Particular, by exploring the game properties, this new fitness function significantly reduce the search cost to $O(n)$ elements in the 8-D space. Besides, based on the domain knowledge, one BiQAP solver is improved to work better statistically. In the second stage, to more efficiently solve each allocated single evader pursuit, the proposed PCCPSO-R algorithm improves the CCPSO-R to enable the safe simultaneous observation, decision-making, and action execution of agents.

Problem 4: how to handle the partial observability? The partial observation constraint and the new challenges it introduced are studied in the partially observable pursuit game for multiple dynamic evaders in Chapter 5. Specially, to promote the research of implicit multi-agent coordination without communications, this work fills the current research gap and investigates the self-organizing pursuit (SOP) problem by imitating the natural self-organization system, which is featured by large-scale, decentralization, partial observation, no communication, no interagent collision, multiple distributed evaders, and surrounding-based capture. Methodologically, to cope with the more challenging observation uncertainty and interaction uncertainty [56] resulted from no communication, the distributed hierarchical framework called the fuzzy self-organizing cooperative coevolution (FSC2) is proposed with three component algorithms: (1) a fuzzy task allocation algorithm that deals with the consensus issue in the distributed task allocation without communications; (2) a reinforcement learning (RL) policy that is trained to learn the unknown distributed search strategy for large-scale homogeneous agents; and (3) the cooperative co-evolution algorithm for robots (CCR) that further improves the CCPSO-R and PCCPO-R algorithms in the safe close coordination.

Problem 5: how to get increasing improvements from autocurricula and effectively train the passive policy? Rather than only learn the active pursuing policy, the autocurricula in co-evolutionary pursuit-evasion games is investigated in Chapter 6. First, this research reviews and analyzes the co-evolution mechanism in the multi-agent setting, which clearly reveals its relationships with autocurricula, self-play, arms races, and adversarial learning. This allows us to better understand related research works, more accurately use similar terminologies, and discuss some common misleading expectations and magics, especially for researchers who are new to this field. Then, toward the strategy with increasing complexities in the autocurricula, various arms race outcomes of different co-evolution mechanisms are achieved through adversarial

learning. Based on experiments, arms races with steady and converging improvement are more practical for increasingly complex behaviors, while policy cycles between two rival sides are useful for producing diverse policies. In particular, this research finds that the passive (evasive) policy learning benefits more from co-evolution than active (pursuing) policy learning in an asymmetric adversarial game. An arms race can drive the passive policy to a higher level than that in normal RL.

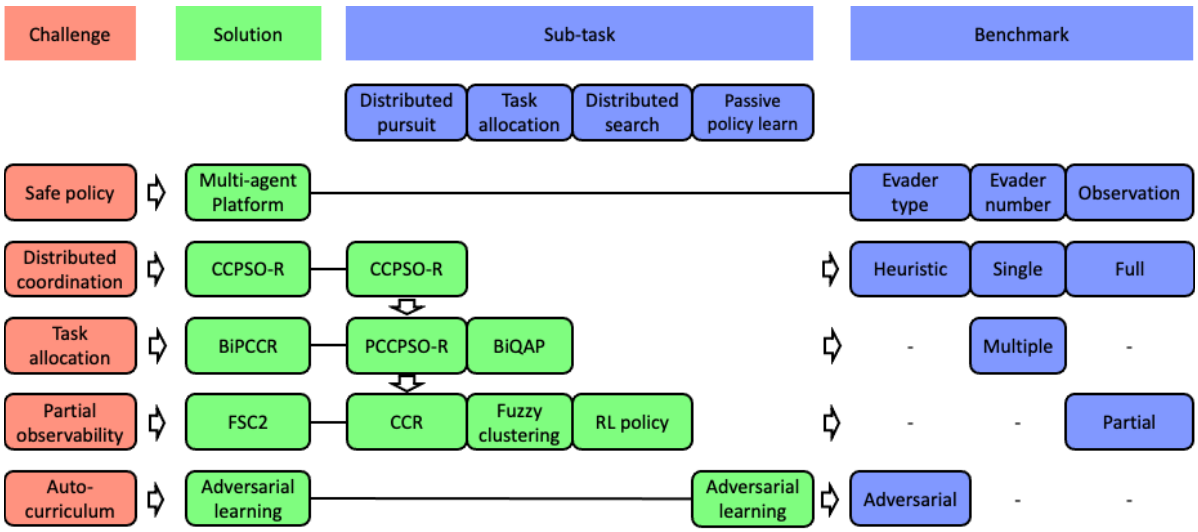


Figure 1.1: Research map of multi-agent coordination algorithms for pursuit-evasion. “-" indicates the same with its above configuration.

1.4 Summary

The rest of this thesis is organized in a way of solving more and more constrained pursuit-evasion tasks. Chapter 2 presents a safety-constrained multi-agent pursuit-evasion platform, based on which the works of Chapter 3 to 6 are conducted. Chapter 3 addresses the fully observable pursuit task of a single evader. Chapter 4 deals with the fully observable pursuit task of multiple evaders. Chapter 5 copes with the partially observable pursuit task of multiple evaders. Chapter 6 approaches the adversarial co-evolutionary pursuit and evasion tasks. Chapter 7 gives the conclusion, limitation, and future work of this research.

CHAPTER 2

MATRIXWORLD: SAFETY-CONSTRAINED MULTI-AGENT PURSUIT-EVASION PLATFORM

2.1 Background

In the current practice, multi-agent reinforcement learning (MARL) is still weak and challenging in resolving the safety guarantees in multi-agent coordination [60, 99], such as autonomous driving [95]. Although complex multi-agent behaviors have already been successfully trained, the safe MARL has not gone beyond toy domains like grid worlds. In the current multi-agent benchmarks, such as MAgent [100], MPE [49], and PettingZoo [88], there are two main safety related limitations. First, they focus more on the diverse multi-agent behaviors design rather than diverse safety requirements, which is however crucial for real-world applications. Second, their collisions resolution mechanisms are biased, transparent, or can lead to unexpected outcomes in the simultaneous multi-agent action execution [88], whereas the stochastic game (SG) is the most popular model for multi-agent sequential problems.

To address the above two limitations and serve as one start work in safe MARL, this chapter proposes a safety-constrained adversarial multi-agent environment: MatrixWorld, based on co-evolutionary pursuit-evasion games. In particular, a safety-constrained multi-agent action execution model is proposed for the general software implementation of safe multi-agent environments. Its rationality, feasibility, and diversity are guaranteed by considering practical collision types and resultant collision

outcomes, which will guide a learning algorithm learn correctly from right reward feedback. Based on MatrixWorld, nine pursuit-evasion variants are designed from real-world applications and conventional literature works, which provide the benchmark problems for the following chapters in this thesis.

2.2 Safety-constrained multi-agent action execution model

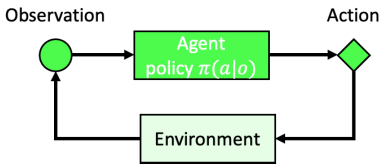
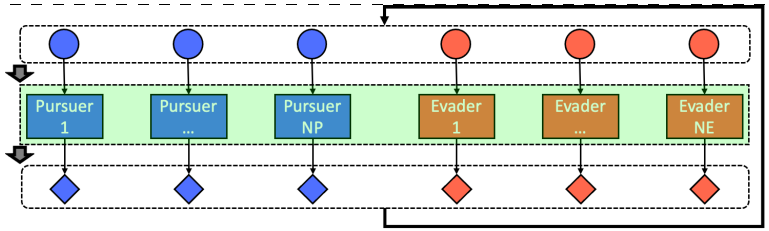
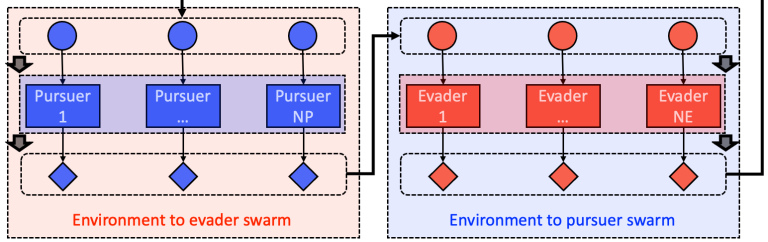
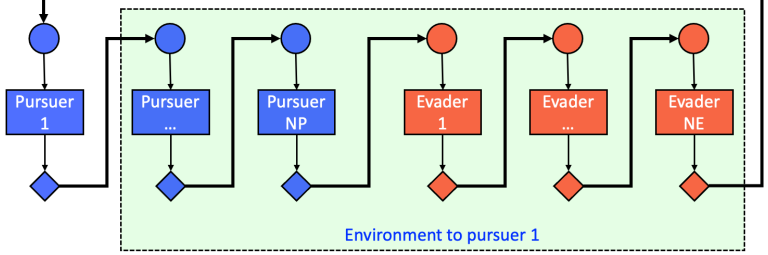
The safety-constrained multi-agent action execution model includes two parts: (1) the multi-agent-environment interaction model, which determines the execution of agents' actions; and (2) the safety-constrained multi-agent collision resolution mechanism, which determines the collision results and who should be responsible, i.e., the evidence to give right rewards, etc.

2.2.1 Multi-agent-environment interaction model

For a single-agent system, the typical agent-environment interaction (AEI) loop is that: at each time step, (1) an agent receives an observation from the environment, (2) based on the observation the agent (policy) makes its decision and outputs an action, (3) the action is executed in the environment, and (4) the environment changes by responding to the action and other factors [85].

For adversarial multi-agent settings, e.g., pursuit-evasion games, their agent environment interaction models can be categorized based on two dimensions: whether agents take turns in a swarm-by-swarm manner, and whether agents take turns in an agent-by-agent manner, as shown in Table 2.1. Therefore, the second row in the table is a strict stochastic game where all agents concurrently observe, make decisions, and execute actions. The third row in the table is a two-swarm turn-taking game and a one-swarm stochastic game for each swarm. Similar to the two-player turn-taking game, a two-swarm turn-taking game can be defined as a game in which two swarms of agents take turns, i.e., swarm-by-swarm, observing, making decisions, and executing actions until the end of the game. Finally, the fourth row in Table 2.1 is a multi-agent turn-taking game, rather than a single-agent system, because the other agents also optimize some objectives in terms of the current agent in an AEI loop [72].

Table 2.1: Multi-agent-environment interaction models for distributed adversarial multi-agent systems. Task environment properties [72]: **Single-agent vs. multi-agent**: whether the other agent(s) optimizes some objectives that depend on the current agent. **Cooperative vs. non-cooperative**: whether all agents share a common goal. **Static vs. dynamic**: whether the environmental state or the agent's observation changes if the agent does nothing regardless of the flying time. **Deterministic vs. nondeterministic (stochastic)**: whether "the next state s' is completed determined by the current state s and the action a executed by the agent(s)", i.e., whether the transition function $P(s'|s,a) = 1$ or not.

Game of	Two-swarm turn taking	Agent-agent turn taking	Agent-environment interaction model	Environment model & properties
Single-agent system				Markov decision process
Adversarial multi-agent system (e.g., pursuit-evasion)	✗	✗		Stochastic game - multi-agent - non-cooperative - dynamic - stochastic
	✓	✗		Two-swarm turn-taking game - multi-agent - non-cooperative - dynamic - stochastic
	✓	✓		Agent environment cycle game - multi-agent - non-cooperative - dynamic - stochastic

2.2.2 Safety-constrained multi-agent collision resolution mechanism

This research implements safety constraints by designing a collision resolution mechanism based on meaningful and practical safety definitions. As shown in Figure 2.1, collisions in the multi-agent setting are classified into three types: (1) agent-obstacle collisions, (2) cooperative agent-agent collisions, and (3) competitive agent-adversary collisions. Agent-obstacle collisions are common in single and multiple agent systems, where only one of the two colliding objects is the decision-maker that should be held responsible. Collisions between cooperative agents are not hostile and arise from conflicts of interest during cooperation, where both sides are accountable. In contrast, competitive agent-adversary collisions may be intentional from one side and unexpected from the other, where the reward strategy depends on the give case. Furthermore, this research considers three outcomes for any collision: (1) reach (target) and disappear, (2) reach (target) and alive, and (3) bounce back (stay put) and alive. These collision types, outcomes, and resultant reward strategies form the collision resolution mechanism that satisfies the safety requirements of different scenarios in Table 2.2.

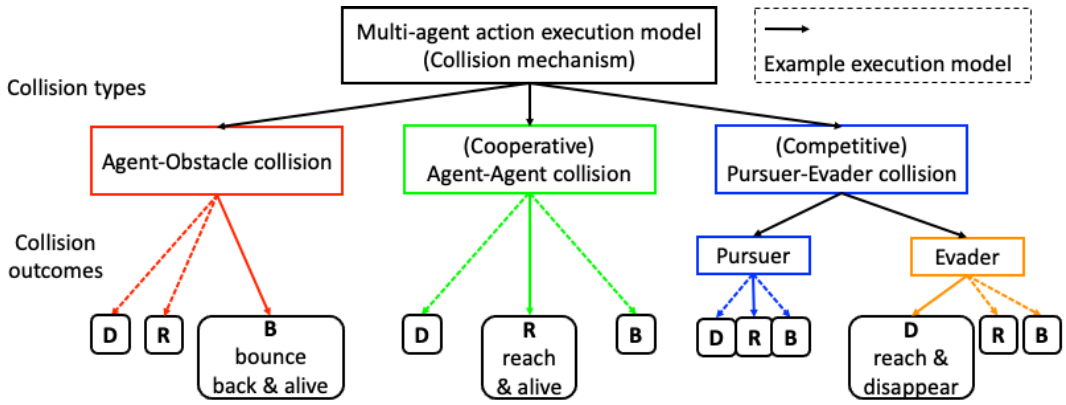


Figure 2.1: Safety-constrained multi-agent collision resolution mechanism for the multi-agent environment modeled by stochastic game.

2.3 Pursuit-evasion game variants

Pursuit-evasion game variants can be defined based on three dimensions: (1) the agent-environment interaction model (Section 2.2.1), (2) the collision resolution mechanism (Section 2.2.2), and (3) the capture behavior. In view of the literature conventions and

Table 2.2: Rationality and example applications of various collision outcomes in the multi-agent setting. **D:** reach (target) and disappear. **R:** reach (target) and alive. **B:** bounce back (stay put) and alive.

Collision type	Agent type	Outcome			Rationality (example application)
		D	R	B	
Agent-obstacle	Agent	✓	-	-	Vehicle crash.
		-	✓	-	Allowed in some MARL environments.
		-	-	✓	Practical in many real-world applications.
Agent-agent	Cooperative	✓	-	-	Vehicle crash.
		-	✓	-	Allowed in some MARL environments.
		-	-	✓	Bumper cars.
Agent-adversary	Evader	✓	-	-	Vehicle crash.
	Pursuer	✓	-	-	
	Evader	✓	-	-	Predator eats prey.
	Pursuer	-	✓	-	
	Evader	-	-	✓	Pursuer is stronger.
	Pursuer	-	✓	-	
	Evader	-	-	✓	Bumper cars.
	Pursuer	-	-	✓	

real-world applications, this research proposes nine pursuit-evasion tasks, i.e., the -D, -R, -B, -O, -S, -SB, -SD, -SDB, and -TO variants of the pursuit-evasion task in Table 2.3.

This thesis mainly considers the first multi-agent-environment interaction model in Table 2.1 for the stochastic game due to its generality in modeling multi-agent games, based on which most of the pursuit-evasion variants are designed. In addition, the Pursuit-Evasion-TO task is designed based on the two-swarm turn-taking and agent-agent turn-taking model in Table 2.1 due to its advantages in theory analysis and the past sustained research [13].

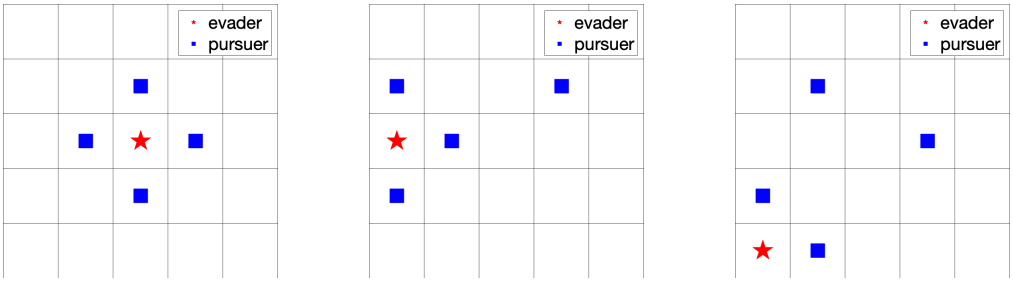
For cooperative agent-agent collisions, typical MARL tasks allow agents to collide with each other without fatal consequences, i.e., “reach and alive". Therefore, this research includes this concern in the -R, -O, -S, and -SD variants of the Pursuit-Evasion task, which may ease the learning of complex strategies. However, in some real-world applications, such collisions may be illegal, where the “bounce back and alive" setting is actually applied. Hence, the -B, -SB, and -SDB variants are designed accordingly to upgrade their safety.

For capture behaviors in the grid world, two main categories are considered. One is the occupation-based capture in which an evader is captured if one or more pursuers

Table 2.3: **MatrixWorld**: Task definition. Collision type: **A-O**: agent-obstacle; **A-A**: cooperative agent-agent; **P-E**: pursuer-evader. Collision outcome: **D**: reach (target) and disappear; **R**: reach (target) and alive; **B**: bounce back (stay put) and alive.

Task environment (Pursuit-Evasion)		Agent-environment interaction		Collision resolution mechanism			Capture behavior	Related work and application (examples)
		Two-swarm turn-taking	Agent-agent turn-taking	A-O	A-A	P-E		
-D	Evader / Pursuer	✗	✗	D	D	D	Occupation-based	Real-world drone and vehicle swarm
-R	Evader / Pursuer	✗	✗	B	R	R	Occupation-based	Multi-agent path finding (MAPF) MPE predator-prey [49]
-B	Evader / Pursuer	✗	✗	B	B	R	Occupation-based	Multi-agent path finding (MAPF)
-O	Evader / Pursuer	✗	✗	B	R	D / R	Occupation-based	Predator-prey; Space search
-S	Evader / Pursuer	✗	✗	B	R	B / R	Surrounding-based	Benda et al. [9]; Sun et al. [82]
-SB	Evader / Pursuer	✗	✗	B	B	B / R	Surrounding-based	Predator-prey
-SD	Evader / Pursuer	✗	✗	B	R	B / R	Surrounding-based disappear if captured	Predator-prey
-SDB	Evader / Pursuer	✗	✗	B	B	B / R	Surrounding-based disappear if captured	Predator-prey
-TO	Evader / Pursuer	✓	✓	B	R	D / R	Occupation-based	Cops and Robbers or vertex-pursuit [13, 59, 68]; PettingZoo Pursuit [88]

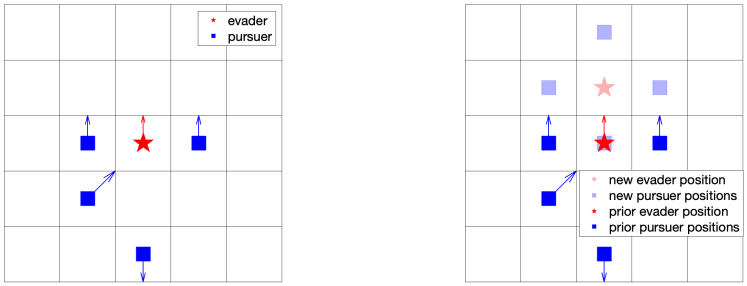
occupy its position. The other is the surrounding-based capture in which an evader is captured if enough pursuers and environment boundaries or obstacles encircle it such that it cannot move.



(a) (b) (c)

Figure 2.2: Illustration of the surrounding based capture.

Bounded world vs. toroidal world. In many researches, a toroidal world is selected to simulate an infinite world, where an agent comes out of one edge will comes in immediately from the opposite edge. However, this kind of world is not practical. As



(a) (b)

Figure 2.3: One trick of the coordination strategy in a toroidal world.

depicted in Figure 2.3a, if the red pentagram is a linear evader which moves in a straight line towards north and just escapes the nearly encirclement of the pursuers (blue squares), in the real infinite world, the pursuers will never catch the evader if they have the same speed. But in the toroidal world, if the pursuers move as shown in Figure 2.3b, they will capture the evader in the next step. Therefore, in this research, rather than toroidal worlds, bounded grid worlds are selected, which can at least represent partially, although not all, the real world scenarios, such as an indoor room or an outdoor park with boundaries, etc.

2.4 API

The application programming interface (API) of MatrixWorld is designed based on the convention of the RL community, as shown in Figure 2.4. The interface is the same while keeping the background multi-agent-environment interaction model in Table 2.1 transparent for users' convenience. In the dictionary information returned from the environmental step function, this research provides (1) the game status data to monitor and compare the game progress regardless of the specific values in a reward structure; (2) the collision status data to give safety related feedback for performance evaluation, reward shaping, safety constraint construction, etc.; and (3) the "alive" status data of the agents to track their remains and population decreases due to death.

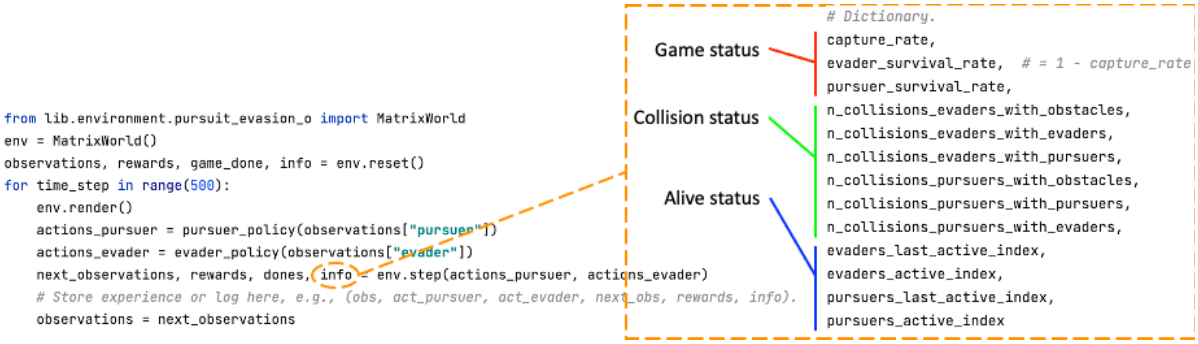


Figure 2.4: Illustration of basic usage of MatrixWorld.

2.5 Summary

This chapter investigates the safe multi-agent coordination problem by studying safe and rational multi-agent collision resolution mechanisms in the general software implementation of multi-agent environments. A safety-constrained multi-agent pursuit-evasion platform: MatrixWorld is presented, where nine representative pursuit-evasion games are provided, i.e., the -D, -R, -B, -O, -S, -SB, -SD, -SDB, and -TO variants of the pursuit-evasion task. The following chapters consider three games in the Pursuit-Evasion-S: the full observable pursuit of single heuristic evader (Chapter 3), the full observable pursuit of multiple heuristic evaders (Chapter 4), and the partial observable pursuit of multiple heuristic evaders (Chapter 5), and one game in the Pursuit-Evasion-O: the partial observable pursuit of multiple adversarial evaders (Chapter 6).

CHAPTER 3

SINGLE EVADER PURSUIT WITH FULL OBSERVATION

3.1 Background

The pursuit domain, or pursuit-evasion problem is a classical and interesting research domain which acts as one of the widely used fundamental testbeds for coordination techniques. On one hand, its apparently simple problem setup and flexibility in approaches or concept evaluations lead to both its popularity and the toy domain impression. On the other hand, it is challenging and thus a good domain for the research of swarm intelligence emerged from the cooperation among agents or agents, which has drawn much attention of researchers on variants of the pursuit domain.

This chapter investigates the dynamic pursuit problem between a single evader and a swarm of fully observable pursuers in the bounded grid world. The cooperative co-evolutionary particle swarm optimization (PSO) (CCPSO-R) algorithm is proposed for the distributed coordination of pursuers, which has the following advantages. First, CCPSO-R is a distributed coordination algorithm. Based on the cooperative co-evolutionary framework, CCPSO-R introduces the co-evolution of real and virtual agents to achieve the balance of individual and swarm benefits in the multi-agent distributed decision-making. Second, CCPSO-R is an online planning algorithm without fixed behavioral rules. With limited domain knowledge, the fitness function is designed in a modular manner, which considers the safe multi-agent coordination problem in the fitness evaluation rather than the algorithm itself. Based on the immediate guidance of the proven efficient swarm intelligence algorithm PSO, CCPSO-R plans one step ahead per time step in a

distributed way. So, no global path planner is necessary as a centralized method, no task allocation is required as a path planning algorithm, and no model pre-training is needed as a reinforcement learning policy. Third, the experimental results show that CCPSO-R is reliable that it can achieve 100% capture rate over different types of evaders in a limited time. CCPSO-R is also scalable that it can provide uniformly capture with any number of pursuers.

3.2 Cooperative co-evolutionary of real and virtual agents

In this research, coevolved pursuers cooperate to encircle an evader, and the performance evaluation function is called the fitness function in the evolutionary computing (EC) [27], which is (functionally) identical to an objective function in the optimization filed. So, the pursuit domain problem can be treated as an optimization problem in the sense that the goal is to improve the fitness of the pursuit process. In particular, it is a dynamic optimization problem where the movement of the evader is a kind of environment change to the pursuers. The optimization is conducted by a particle swarm optimization (PSO) based cooperative co-evolutionary (CC) algorithm called CCPSO-R, which solves the distributed on-line planning problem of pursuers in each time step.

In CCPSO-R, there are N_s independently evolved subpopulations with subpopulation size N_p, and the first individual of each subpopulation represents a unique real agent while the others represent virtual agents. All the real agents consist of the pursuers swarm which actually pursue the evader in the grid world, while the virtual agents are to explore the vicinity of the corresponding real pursuer agent in its subpopulation and guide the pursuer to a better position. So, in this sense, virtual agents can be seen as the action space of the corresponding real pursuer. The real pursuer chooses its locally optimal action, but in terms of the global benefit of the whole pursuers swarm. That is, the evaluation of an agent position is conducted by considering the rest real pursuers' positions in the other subpopulations. Since the proposed algorithm works in the mode of cooperative co-evolutionary algorithms, it is called the cooperative co-evolutionary PSO for robots (CCPSO-R), as illustrated in Algorithm 3.1, which will be explained in detail from three aspects: the fitness function design and evaluation, the update rules of agents, and the diversity maintenance mechanism in the following.

Algorithm 3.1: CCPSO-R

```
Initialization
while the evader is not captured and time limit is not reached do
    for each subpopulation do
        Re-evaluate the subpopulation due to environmental changes.
        for each virtual agent do
            Update its velocity and position using (3.8) and (3.9).
            Evaluate the fitness together with the rest real pursuers.
        if unique virtual agents < T_v then
            Re-initiate and re-evaluate the virtual agents.
        Update the velocity and position of the real pursuer using (3.13) and (3.14).
        Evaluate the fitness of the real pursuer with the rest real pursuers.
        if the real pursuer becomes the global best then
            Re-initiate and re-evaluate the virtual agents.
        else if the real pursuer gets trapped in a deadlock then
            Add a random noise to the real pursuer's position.
            Re-evaluate the whole population.
    if the real pursuer swarm get trapped in a local optimum then
        Add random noises to all real pursuers' positions.
        Re-evaluate the whole population.
```

3.2.1 Fitness function and evaluation

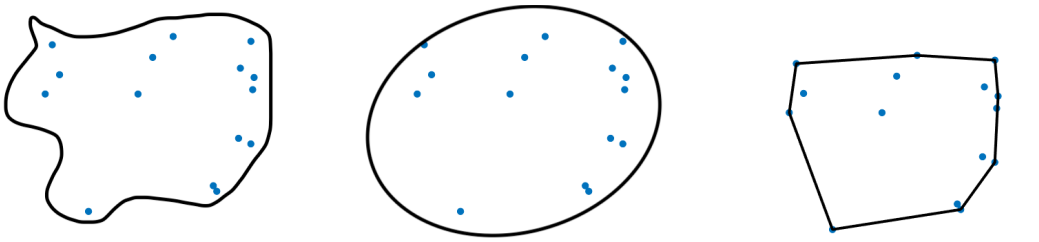
Fitness function. According to the capture definition in Chapter 2 and the task that a swarm of pursuers need to encircle an evader, the fitness function should subject to the following metrics.

- *CLOSURE* $f_{closure}$: the evader should locate inside the convex hull of the pursuers' positions;
- *SWARM EXPANSE* $f_{expanse}$: the swarm of pursuers should concentrate around the evader, i.e., a smaller swarm expanse of the pursuers is preferred;
- *UNIFORMITY* $f_{uniformity}$: the pursuers should distribute uniformly around the evader;
- *COLLISION AVOIDANCE* f_{repel}: collisions among real (pursuer or evader) agents are not allowed in the practical sense.

It is obvious that a single pursuer itself cannot form a solution. In CCPSO-R, a complete solution to the pursuit problem is composed by the positions of all the pursuers. However, before formulating the fitness function, a definition needs to be introduced first.

Definition of Convex Hull [25]: The *convex hull* of the point set P, denoted by conv(P), is the intersection of all convex regions that contain P.

An intuitive illustration of this definition can be found in [25] as in Figure 3.1.



(a) A nonconvex region enclosing P.

(b) A convex region enclosing P.

(c) The convex hull of P.

Figure 3.1: Illustration of the convex hull of the point set P [25].

Further, this research defines the function:

$$inconv(p, conv(P)) = \begin{cases} 0, & \text{if point } p \text{ is in } conv(P) \\ 0.5, & \text{if point } p \text{ is on the edge of } conv(P) \\ 1, & \text{otherwise} \end{cases} \tag{3.1}$$

Hence, the fitness function for the jth ($j = 1,...,N_p$) individual (agent) in the ith ($i = 1,...,N_s$) subpopulation $p^{ij}_{pursuer}$ is defined as

$$f^{ij} = f^{ij}_{repel} \cdot (f^{ij}_{closure} + f^{ij}_{expanse} + f^{ij}_{uniformity}) \tag{3.2}$$

where

$$f^{ij}_{repel} = \begin{cases} e^{-2\cdot(nnd^{ij} - D_{min})}, & \text{if } nnd^{ij} < D_{min} \\ 1, & \text{else} \end{cases} \tag{3.3}$$

corresponds to the above *COLLISION AVOIDANCE* metric,

$$f^{ij}_{closure} = inconv(p_{evader}, conv(p^{11}_{pursuer}, ..., p^{ij}_{pursuer}, ..., p^{N_s 1}_{pursuer})) \tag{3.4}$$

corresponds to the above *CLOSURE* metric,

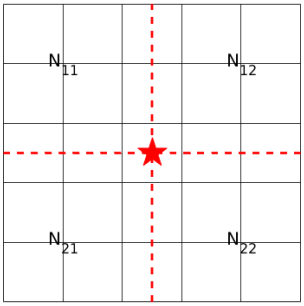
$$f^{ij}_{expanse} = \frac{1}{N_s}\Big(\sum_{k=1, k\neq i}^{N_s} |p^{k1}_{pursuer} - p_{evader}| + |p^{ij}_{pursuer} - p_{evader}|\Big) \tag{3.5}$$

corresponds to the above *SWARM EXPANSE* metric, and

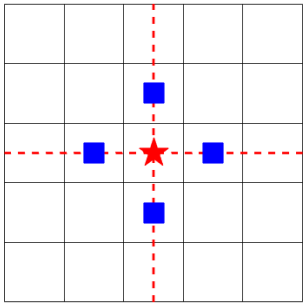
$$f^{ij}_{uniformity} = std\left(\begin{bmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{bmatrix}\right) \tag{3.6}$$

corresponds to the above *UNIFORMITY* metric.

In the above formulas, nnd^{ij} is the nearest neighbor distance, i.e., the minimum of the pairwise Euclidean distances between the jth individual in the ith subpopulation and all the real pursuers in the other subpopulations; D_{min} is a specified secure distance for collision avoidance; p_{evader} is the position of the evader; $p^{ij}_{pursuer}$ is the position of the jth pursuer in the ith subpopulation; $std(\cdot)$ stands for the standard deviation function; and $N_{kh}(k = 1,2; h = 1,2)$ is the counts of the real pursuers in the (k,h)-th bin out of the overall four bins split by the horizontal and vertical lines which intersect at the position of the evader, as shown in Figure 3.2. Note that, the number of the real pursuers



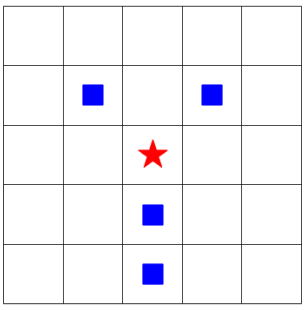
(a) 4 bins split around the evader.



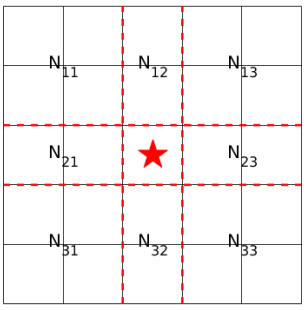
(b) Example for the uniformity assessment.

Figure 3.2: Illustration of the uniformity assessment.

on the split lines is divided by 2 and equally assigned to the two adjacent bins. Hence, $N_{11} = N_{12} = N_{21} = N_{22} = 1$ for the example shown in Figure 3.2b. However, the formula



(a) The uniformity assessment is 0 by equation (3.6) based on the split of Figure 3.2a.



(b) An alternative split method for the uniformity assessment of equation (3.7).

Figure 3.3: Illustration of the alternative uniformity assessment.

(3.6) cannot always give the objective uniformity assessment that is consistent with a human's subjective judgment, as the deadlock phenomenon shown in Figure 3.3a. In this scenario, the evader always keeps still in the center of the map, while the pursuers start to encircle the evader from randomly generated initial positions and stop forever since the game state shown in Figure 3.3a, which is obviously not the expected capture state. So, the deadlock phenomenon is a game state where the pursuit task has not been accomplished but all the pursuers stop forever as if they are locked. One reason of the deadlock phenomenon is the fitness function wrongly evaluates an intermediate game state as fittest, i.e., the task is finished. Therefore, to design a better fitness function, in the uniformity assessment formula, an alternative space split strategy is performed as shown in Figure 3.3b, and the following uniformity assessment will replace equation (3.6) in such situations:

$$f^{ij}_{uniformity} = std([N_{12}, N_{21}, N_{23}, N_{32}]) + std([N_{11}, N_{13}, N_{31}, N_{33}]), \tag{3.7}$$

the first and second part of which are the axial and diagonal uniformity assessments, respectively.

Fitness evaluation. To evaluate the fitness of the jth ($j = 1, ..., N_p$) individual (agent) in the ith ($i = 1, ..., N_s$) subpopulation, a complete solution should be first composed by replacing the ith real pursuer with $p^{ij}_{pursuer}$ from the real pursuers swarm:

$$\left[p^{11}_{pursuer}, ..., p^{(i-1)1}_{pursuer}, p^{ij}_{pursuer}, p^{(i+1)1}_{pursuer}, ..., p^{N_s 1}_{pursuer}\right].$$

Then, the fitness of an agent can be evaluated by equation (3.2), as shown in Figure 3.4.

From the practical point of view, no collisions of any two real agents are allowed. Since there are totally N_s subpopulations in the cooperative co-evolution population, a priority scheduler is used to coordinate among them. In particular, a priority scheduler will decide the order of movements among real agents, which can be the evolving order of subpopulations in asynchronous situations, and can also be used to coordinate, for example, two real agents when they both want to go to the same position in synchronous situations. To be as simple as possible, here the priorities are in consistent with the indexes of subpopulations. In other words, after the evader moves, the subpopulations evolve one-by-one and the newly updated real pursuer is counted into the dynamics of the environment for the fitness evaluations of the subsequent subpopulations. So, if $k > h$, the pursuer $p^{k1}_{pursuer}$ always moves ahead of the pursuer $p^{h1}_{pursuer}$.

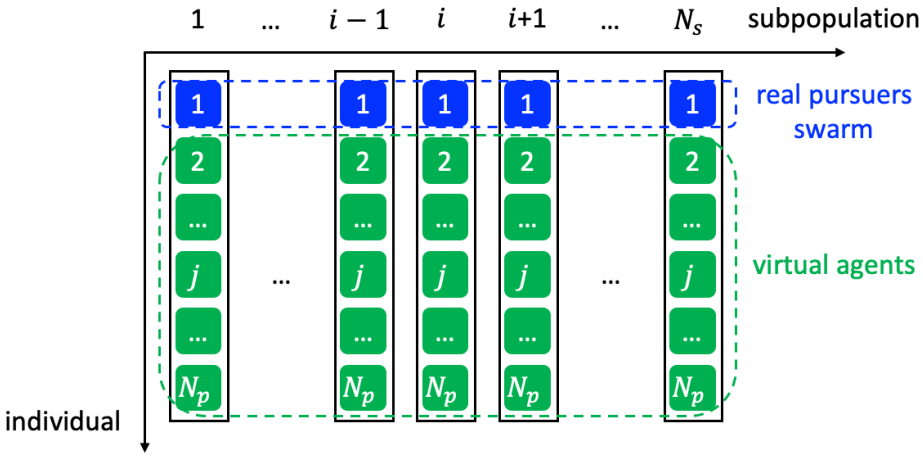


Figure 3.4: Illustration of the fitness evaluation for $p^{ij}_{pursuer}$.

3.2.2 Behavioral update rule

Two update rules are designed separately for virtual and real agents:

1. For a virtual agent j ($j \in \{2,...,N_p\}$), the PSO update rules are as follows:

$$v^{ij}_{pursuer} = nnd(w \cdot v^{ij}_{pursuer} + c_1 \cdot r_1 \cdot (pi^{ij}_{pursuer} - p^{ij}_{pursuer}) + c_2 \cdot r_2 \cdot (pg^{i}_{pursuer} - p^{ij}_{pursuer})) \tag{3.8}$$

$$p^{ij}_{pursuer} = nbn((p^{ij}_{pursuer} + v^{ij}_{pursuer}), p^{i1}_{pursuer}) \tag{3.9}$$

where

$$nnd(v) = \arg\min_{p_n \in SN} |\angle p_n - \angle v| \tag{3.10}$$

and

$$SN = \{(1,0),(1,1),(0,1),(-1,1),(-1,0),(-1,-1),(0,-1),(1,-1)\}. \tag{3.11}$$

$nnd(v)$ outputs one of the eight unit vectors in SN which has the minimum angle distance with the input velocity v. By using the function $nnd(\cdot)$, every agent can only move one step by one step. In this way, unlike the multi-steps case in a general PSO, the path planning and the worry about collisions in the half way to a destination are not ever necessary . $v^{ij}_{pursuer}$ is the velocity for the jth individual (agent) in the ith subpopulation which has the position $p^{ij}_{pursuer}$. In addition, $pi^{ij}_{pursuer}$ is the individual historical best position for the jth individual (agent) in the ith subpopulation, while $pg^{i}_{pursuer}$ is the global best position of the ith subpopulation. The coefficient $w \in R$ is called the inertia

weight, $c_1, c_2 \in R^+$, and r_1, r_2 are uniformly distributed random numbers in the range of $(0,1)$. Besides,

(3.12)

$$nbn(p^{ij}_{pursuer}, p^{i1}_{pursuer}) = \begin{cases} \arg\min_{p^{i1}_b} |\angle(p^{i1}_b - p^{i1}_{pursuer}) - \angle(p^{ij}_{pursuer} - p^{i1}_{pursuer})|, \\ \qquad\qquad \text{if } p^{ij}_{pursuer} \text{ is out of the vicinity of } p^{i1}_{pursuer} \\ p^{ij}_{pursuer}, \qquad \text{otherwise} \end{cases}$$

is designed to output the nearest boundary neighbor p^{i1}_b in the constrained vicinity of the real pursuer $p^{i1}_{pursuer}$. This function is illustrated in Figure 3.5, where the constrained vicinity of $p^{i1}_{pursuer}$ is shown in a dashed square, which is determined as the minimum one that can accommodate the specified number of virtual agents. Note that, the $nbn(p^{ij}_{pursuer}, p^{i1}_{pursuer})$ function in Equation (3.9) is very important because it can assure all the virtual agents $p^{ij}_{pursuer} (j \geq 2)$ are in the constrained vicinity of the real pursuer $p^{i1}_{pursuer}$, without which the subpopulation may lose the vicinity exploring capability for the real pursuer.

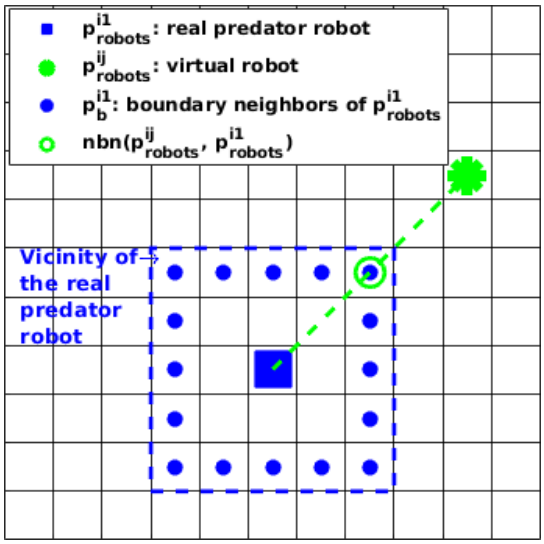


Figure 3.5: Illustration of the function $nbn(p^{ij}_{pursuer}, p^{i1}_{pursuer})$ in Equation (3.12).

2. For the real pursuer ($j = 1$), the PSO update rules are as follows:

(3.13) $$v^{i1}_{pursuer} = nnd(pg^i_{pursuer} - p^{i1}_{pursuer})$$

(3.14) $$p^{i1}_{pursuer} = p^{i1}_{pursuer} + v^{i1}_{pursuer}$$

So, the real pursuer does not need to perform any exploring task, but just quickly becomes the global best in its subpopulation.

To summarize, by utilizing different optimization mechanisms for different kinds of agents, virtual agents are responsible for exploring and finding potential better positions in the vicinity of the real pursuer, while the real pursuer in each subpopulation just makes use of the achievements of the virtual agents and becomes the global best.

3.2.3 Diversity maintenance mechanism

When a swarm intelligence algorithm converges, all individuals may be attracted to the same position, no matter it is the global or local optimum. However, for the pursuit case here, the convergence of virtual agents in a subpopulation brings the disadvantage that the capability of exploring potentially better positions is getting worse. Therefore, if the number of unique virtual agents in a subpopulation is defined as the subpopulation diversity, the diversity of each subpopulation must be maintained to keep its exploring capability. Besides, due to the existence of unexpected deadlocks, suitable strategies should be integrated in the coordination algorithm to deal with such problems.

Based on the above ideas, this research proposes the diversity maintenance mechanisms which are performed as follows:

- Update the population in each generation based on the scheme that the fitness of the newly generated individual is not worse than its parental agent, which will guide the agent to explore more positions without harm to the fitness.

- Redistribute the virtual agents once the number of unique virtual agents positions in a subpopulation decreases below a threshold T_v, i.e., the subpopulation converges. That is, the subpopulation has found better solutions and all agents are attracted to the global best. In this situation, virtual agents should be redistributed to the space for better exploration. This strategy corresponds to the line 8-9 in Algorithm 3.1.

- Redistribute virtual agents once the real pursuer becomes the global best in the subpopulation. Because the role of virtual agents is to help the corresponding real pursuer to find better positions, once this real pursuer becomes the global best in its subpopulation, the object of virtual agents is reached and they should be redistributed to the space to find potential better positions for the real pursuer. This strategy corresponds to the line 12-13 in Algorithm 3.1.

- Add a random noise to the position of the real pursuer if it is not the global best in its subpopulation but abnormally keeps stills for a long time, in which it must

have gotten stuck in a deadlock. This strategy corresponds to the line 14-16 in Algorithm 3.1.

- Add random noise to the positions of all the real pursuers if they converge when the evader has not been captured, the situation of which can be seen as that the swarm of pursuers gets trapped in a local optimum. This strategy corresponds to the line 17-19 in Algorithm 3.1.

3.3 Experiments

In this section, two different experiments are presented. Experiment 1 is conducted in a 30×30 grid world to verify the performance of the proposed CCPSO-R. Experiment 2 is to compare CCPSO-R with a representative dynamic path planning based pursuit algorithm MAPS [91]. From the experimental results, pros and cons of two different strategies can be seen in spite that CCPSO-R and MAPS are originally designed towards different capture definitions.

In particular, to verify the generality of algorithms, four types of evaders are implemented. The evader initially locates in the center of the world, but behaves differently according to its type defined as follows:

- STILL EVADER: the still evader keeps still in its initial position forever.
- RANDOM EVADER: the random evader randomly moves to a next position according to the uniform distribution.
- LINEAR EVADER: the linear evader initially chooses one of the 8 directions in which the number of pursuers is minimum, and moves in that direction in a straight line since then. Only when the evader locates on edges of the map, it will re-calculate a new direction according to the same criterion. However, when the way of the linear evader is blocked by a pursuer, it cannot move any more but only wait for the other pursuers coming to encircle it.
- SMARTER LINEAR EVADER: the smarter linear evader, represented as linear_smart, is very similar to the linear evader. The only difference is that when its way is blocked by a pursuer, it moves to an unoccupied neighbor which has the minimum angle distance with its current direction and then it continues its movement in the same direction if there are no obstacles.

From the above descriptions, the capabilities of the evaders and the difficulties of encircling evaders can be intuitively ranked as “still evader < random evader < linear evader < linear_smart evader", which will be further verified by the following experiments.

3.3.1 Experiment 1 (Surrounding-based pursuit)

3.3.1.1 Experimental setup

To verify the scalability of CCPSO-R, various sizes of the swarm of pursuers, i.e., 4, 8, 12, 16 and 24, are used, from which the advantages originated from the swarm intelligence of swarm of pursuers can be expected.

The other implementation details are as follows: the initial real pursuers are deployed randomly in the whole grid world without overlapping; the population size of each subpopulation is 20; the evader moves in 90% of the time ensuring that pursuers move faster or a longer distance than the evader; in equation (3.3) $D_{min} = 1$ which is the minimum secure distance between two agents; in equation (3.9) the parameters $w = 1, c_1 = c_2 = 2$ which are set as recommended in [77]; T_v is 9 in the line 8 of Algorithm 3.1 which is the number of grids for a 3×3 vicinity; when the real agent is not the global best in its subpopulation but keeps still over 5 iterations, it is seen as getting trapped in a deadlock which corresponds to the line 14 of Algorithm 3.1; when the swarm of pursuers keeps still over 10 iterations, the swarm is seen as converged, and if the swarm has converged but the evader has not been captured, the swarm is seen as getting trapped in a local optimum which corresponds to the line 17 of Algorithm 3.1.

In addition, for environmental changes such as real pursuer agent position change in other subpopulations, the current subpopulation needs to be re-evaluated as shown in the line 4 of Algorithm 3.1, where the individual historical best position pi^{ij}_{agents} will not be inherited, and the global best pg^{i}_{agents} will be re-calculated. This is because, although the experimental results of inheriting and not inheriting the individual historical memory pi^{ij}_{agents} differ, it is hard to select either one due to their competitive performances.

As for the performance metrics, this research uses the number of successful captures, the average number of moves to capture the evader, and their standard deviations over 100 randomly generated test cases given the maximum 1000 time steps, the random seeds of which are set from 1 to 100.

Table 3.1: Number of captures, average number of moves, and their standard deviations to capture different evaders with various number of pursuers out of 100 test cases.

No. of pursuers	Metrics	Evader			
		Still	Random	Linear	Linear_Smart
4	No. of captures	100	100	100	100
	Avg. of moves	30.450	49.840	46.900	204.060
	Std. of moves	19.943	36.867	31.886	198.927
8	No. of captures	100	100	100	100
	Avg. of moves	22.220	33.780	42.240	121.820
	Std. of moves	13.384	23.039	46.583	111.922
12	No. of captures	100	100	100	100
	Avg. of moves	20.470	24.520	30.190	76.780
	Std. of moves	11.364	13.414	24.943	72.839
16	No. of captures	100	100	100	100
	Avg. of moves	17.360	18.360	25.620	49.850
	Std. of moves	9.648	11.277	23.560	50.048
24	No. of captures	100	100	100	100
	Avg. of moves	15.060	14.060	19.670	35.400
	Std. of moves	10.688	6.151	21.879	32.588

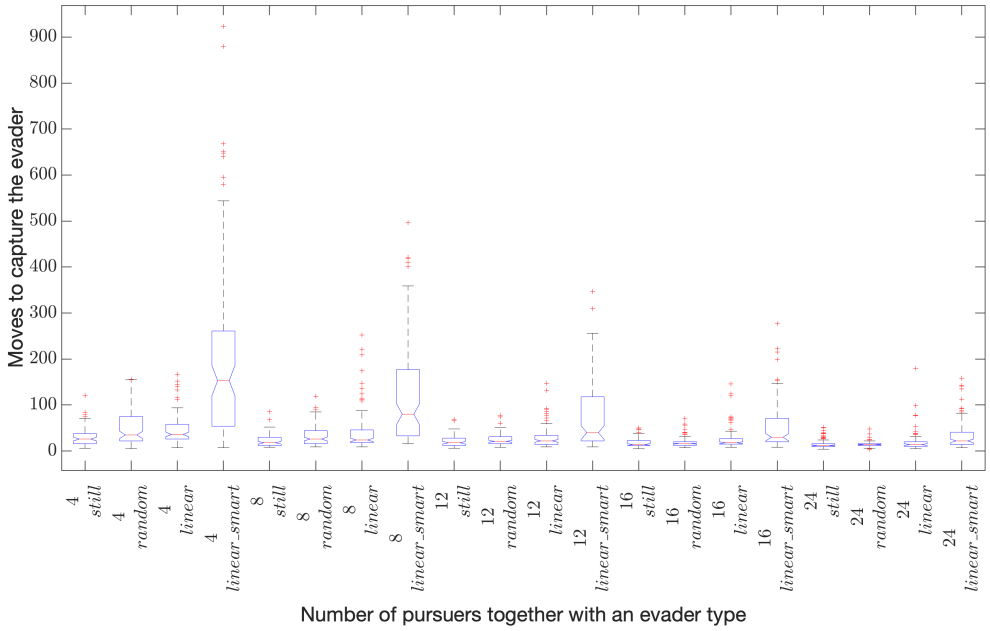


Figure 3.6: Box plot of the moves to capture a specific evader with a specific number of pursuers.

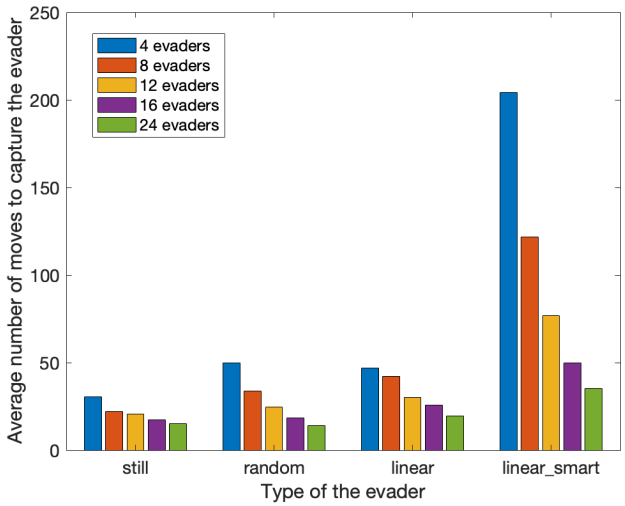


Figure 3.7: Bar graph of the average moves to capture a specific evader with a specific number of pursuers.

3.3.1.2 Experimental results

The simulation results are summarized in Table 3.1, from which it can be seen that CCPSO-R is reliable with the capture rate being 100% in a limited time, no matter what type of the evader it is. As expected before, to a swarm of pursuers, the difficulties, in terms of the average number of moves, to capture each type of evader can be generally ranked as “still evader < random evader < linear evader < linear_smart evader", which can be seen more clearly from Figure 3.6. This conclusion is in consistent with the common opinion in literature (such as [35] and [80]) that compared with the random evader, the straight line moving evader is more effective because it breaks the movement locality. Hence, the straight line moving evader is more difficult to be captured, which leads to the low capture rates in previous work, such as the manually designed methods [35, 44], EA based method [35] and the case learning method [36].

In addition, the data of Table 3.1 is shown in the manner of Figure 3.7, from which an evident fact can be found that the more pursuers the more efficient the pursuit is. Besides, from the decreasing standard deviations as more real pursuers are involved, as shown in Figure 3.6, it can be concluded that with the swarm size of the pursuers gets larger, the pursuit performance is getting more and more stable and robust.

To give a more intuitive impression of the pursuit process, several representative episodes taken from an experiment against the linear_smart evader are displayed in Figure 3.8.

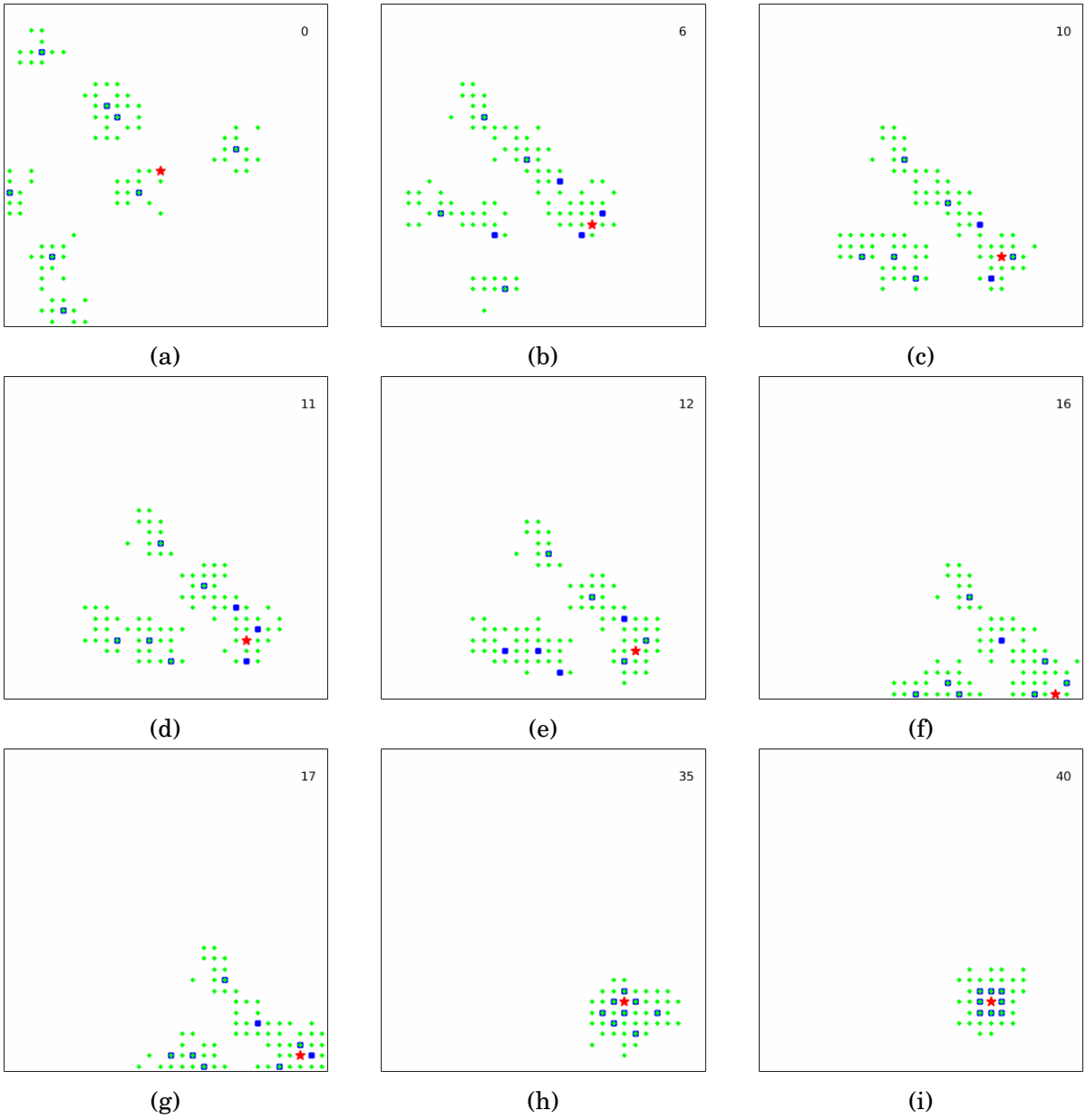


Figure 3.8: Illustration of the pursuit process, taking the pursuit of a linear_smart evader as an example. (a) is the initialization. From (a)-(b), the evader moves in the southeast direction in a straight line. In (c), the evader encounters an orthogonal real pursuer. So, in (d), the evader moves to a nearest unoccupied neighbor in the south. After that, from (d) to (e), the evader continues to move in its previous straight line direction. Until (f), the evader reaches an edge. So, in (g), the evader re-selects the north as its new escape direction. In (h), the evader is captured. And in (i), the pursuers swarm converge.

3.3.1.3 Discussion

The results achieved in Section 3.3.1.2 can be explained from the algorithm's point of view. First, as the pursuit domain is treated as an optimization problem, as long as the designed fitness function (or objective function) properly models the investigated problem, minimizing the fitness function will lead pursuers to a successful capture. Second, on one hand, every step of a real pursuer is greedy since it moves to the best virtual agent position in its subpopulation; on the other hand, each step of a real pursuer is not totally greedy since the virtual agents exploration in its vicinity is not exhausted. So, pursuers may eventually capture the evader but the process may be slow.

Besides, in some of the past work, such as RL and path planning approaches, the capture is defined differently as that the evader position is occupied by a pursuer, the further idealization and simplification of which result in an unpractical problem setup for agents applications. This chapter adopts another conventional definition that the evader is encircled by pursuers such that it cannot move any more, which considers both the collision avoidance and the safety of agents from the practical point of view. However, to further validate the effectiveness of the proposed CCPSO-R, the comparison will be conducted in the next sub-section.

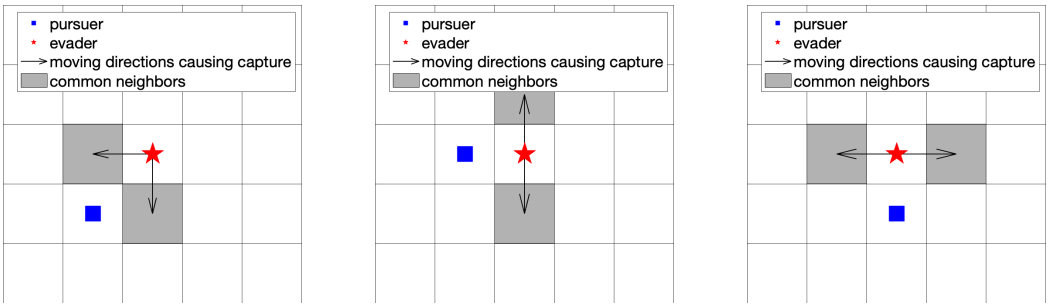
3.3.2 Experiment 2 (Occupation-based pursuit)

For pursuit domain problems, path planning and task allocation based strategies are intuitive and may be the first solution that comes to one's mind. Therefore, in this section, the proposed CCPSO-R is compared with a representative dynamic path planning based algorithm named MAPS [91].

3.3.2.1 Experimental setup

All the experimental setups are the same as those in the Experiment 1 except the number of pursuers, the grid world sizes, and the way in which the capture status is calculated. For fair comparisons, the capture definition is modified to the one adopted by MAPS that the position of the evader is occupied by any pursuer. In particular, in CCPSO-R, when pursuers cooperate to encircle the evader, there must be at least one such moment that the evader is adjacent to a pursuer and they get common neighbors as illustrated in Figure 3.9. Because diagonal obstacles are considered in our collision avoidance design, the number of the common neighbors is at most 2 except the evader's own position. If the evader keeps still or moves to anyone of the common neighbors the next moment,

no matter it is compelled by other coordinated pursuers or just due to the simplicity of itself, the adjacent pursuer can definitely occupy the evader's new location since pursuers always move after the evader. The evader is seen as captured at this moment.



(a) Example 1. (b) Example 2. (c) Example 3.

Figure 3.9: Illustration of the modified capture status of CCPSO-R used in the comparison with MAPS.

3.3.2.2 Experimental results

Two algorithms MAPS and CCPSO-R are run in the same randomly generated scenarios where the initial pursuers and evader positions are the same. To see whether the map size influences the comparison results much, a bigger grid world size 150×150 together with a smaller grid map size 30×30 are applied with two pursuer swarm sizes 4 and 8. Experimental results are summarized in Table 3.2. Since both algorithms capture the evader 100%, the metric of "No. of captures" is not listed. It is not hard to understand that generally with the increase of the pursuers number, less moves with smaller standard deviations are needed for a capture; and with the increase of the grid map size, values in these two metrics increase accordingly.

To further validate the significance of the comparison results in Table 3.2, t-tests are conducted at the 5% significance level, as shown in Table 3.3. It can be seen that CCPSO-R significantly outperforms MAPS in 11 out of the total 16 experimental scenarios where more pursuers are involved with smarter evaders in bigger worlds.

3.3.2.3 Discussion

The comparison results show that although CCPSO-R is designed for the coordinated encirclement of the evader until every available orthogonal neighbor of the evader has been occupied by a pursuer, CCPSO-R also performs well in the position occupying based

Table 3.2: Comparison results: Average number of moves and their standard deviations to capture different evaders with various number of pursuers in different sizes of grid worlds out of 100 test cases.

Metrics			Avg. of moves		Std. of moves	
Number of pursuers			4	8	4	8
Size of the grid map: 30 × 30						
Evader	Still	MAPS	7.09	4.87	3.777	2.751
		CCPSO-R	**6.5**	**4.66**	**2.819**	**2.271**
	Random	MAPS	9.01	6.78	6.973	5.868
		CCPSO-R	**8.3**	**6.04**	**3.335**	**2.828**
	Linear	MAPS	17.89	10.55	**6.977**	8.184
		CCPSO-R	**16.9**	**7.71**	8.487	**5.695**
	Linear Smart	MAPS	17.84	10.55	**7.102**	8.211
		CCPSO-R	**16.92**	**7.75**	8.474	**5.723**
Size of the grid map: 150 × 150						
Evader	Still	MAPS	36.6	26.16	18.229	14.633
		CCPSO-R	**33.37**	**23.07**	**13.9**	**10.992**
	Random	MAPS	38.62	28.55	18.968	16.516
		CCPSO-R	**35.84**	**26.07**	**13.654**	**11.401**
	Linear	MAPS	99.33	64.29	44.576	37.918
		CCPSO-R	**82.75**	**41.32**	**30.274**	**29.684**
	Linear Smart	MAPS	99.19	64.37	44.849	37.814
		CCPSO-R	**82.84**	**41.44**	**30.192**	**29.798**

capture definition, which proves the effectiveness of the cooperative coevolutionary based coordination strategy.

In addition, compared with just occupying the same position of the evader with only one pursuer, occupying all the orthogonal neighbors of the evader simultaneously as uniformly as possible is more complicated. Therefore, the proposed CCPSO-R can accomplish more complicated coordination tasks compared to MAPS.

On the other hand, as presented in [91] that MAPS is a real-time pursuit algorithm, its MATLAB version, which is rewritten by us from its original C++ codes, still runs faster than CCPSO-R. Because the calculation of the average number of moves per second over all the test cases has a high requirement on the runtime environment for fair comparison, this metric is not listed here. But the following conclusions can still be drawn. MAPS is faster. However, one problem of it is that its performance is constrained by the number of pursuers. Because one important step in MAPS is to assign the possible escape directions of the evader to every pursuer optimally by iterating every possible

Table 3.3: Statistical test for CCPSO-R vs. MAPS using the t-test at the 5% significance level.

World size		30×30		150×150	
No. of pursuers	Evader	p-value	Significant	p-value	Significant
4	Still	0.013848	Yes	0.0062804	Yes
	Random	0.13224	No	0.02533	Yes
	Linear	0.10273	No	7.3641e-05	Yes
	Linear Smart	0.11762	No	9.8947e-05	Yes
8	Still	0.13084	No	0.0017754	Yes
	Random	0.11016	No	0.027238	Yes
	Linear	0.00018443	Yes	1.3396e-11	Yes
	Linear Smart	0.00020334	Yes	1.1589e-11	Yes

assignments, the combinatorial number of which is $(n-1)!$ where n is the number of pursuers. With the increase of n, this combinatorial number will increase very fast and it is becoming less practical to get all the permutations at once, which also brings more burden to the memory. This is also the reason that only the simulation results with up to 8 pursuers are compared. But, for CCPSO-R, although it is not as efficient as MAPS, its scalability on the pursuer swarm size is much better.

Finally, despite the fact that both MAPS and CCPSO-R adopt the same sequential movement strategy that the evader always moves first and then the pursuers move one by one, another loop is embedded in the current CCPSO-R implementation that each real pursuer position can only be updated when all its corresponding virtual agents have been updated sequentially, as shown in line 5-10 of Algorithm 3.1. It is known that embedded loops are generally not expected for an efficient algorithm. So, a parallel update and evaluation for the virtual agents should alleviate the efficiency constrain of CCPSO-R which will be investigated in future work.

3.4 Summary

This chapter treated the pursuit domain as an optimization problem and presented the cooperative coevolutionary algorithm—CCPSO-R, which, for the first time, introduces the combination of the real agents and virtual agents into the correspondences between

the individual representation of an EA and the agents in an application. Before the work in this thesis, an individual in an EA will be assigned to a real agent. However, in the proposed CCPSO-R algorithm, only the first individual in each subpopulation corresponds to a real agent, while the rest individuals are all virtual agents, who act as a kind of action space for real agents by sampling and exploring their vicinities.

Besides, it should be noted that there are no fixed behavior rules for the swarm of pursuers. Instead, the swarm of agents is guided directly by the fitness function, which is designed in a modular manner by incorporating very limited domain knowledge. As one module, the collision avoidance consideration is integrated in the fitness function, which itself is another fitness function for repelling and can be versatile by tuning its parameter D_{min}. If the $D_{min} = 1$, as it is in this chapter, the agent swarm can capture the evader while moving without collisions.

Finally, the performances of generality, stability, and scalability of the proposed CCPSO-R with four types of evaders; the still evader, the random evader, the linear evader, and the linear_smart evader are tested. Experimental results have been summarized based on 100 randomly generated test cases whose random seeds are set as 1-100 for their reproducibility. Based on these experiments, it can be concluded that the proposed CCPSO-R can always capture the evader stably and no additional modifications are needed under different scenarios. In addition, a comparison with a representative dynamic path planning and task allocation based algorithm MAPS has also been conducted. Experimental results further prove the outstanding performance of the proposed CCPSO-R.

However, to be simple, the coordination priority scheduler was designed based on the subpopulation indexes, which indicates that the real pursuers move in a fixed sequential order. This may be unreasonable when it is better to firstly move one specific pursuer which blocks others' ways. In addition, pursuers move sequentially, rather than synchronously, will deteriorate the pursuit efficiency when the swarm of pursuers gets larger. Therefore, three works need to be done in future: one is to study the memory inheritance strategy in dynamic optimization problems as mentioned in Section 3.3.1.1; one is to implement the parallel update and evaluation for the virtual agents; one is to improve the coordination scheduler towards the synchronous cooperation based on parallel computing by learning from experiences.

CHAPTER 4

MULTIPLE EVADERS PURSUIT WITH FULL OBSERVATION

4.1 Background

This chapter investigates the dynamic pursuit problem between multiple evaders and a swarm of fully observable pursuers in the bounded grid world, i.e., the multiple-evaders pursuit (MEP) problem. If the capture of an evader is defined as that it cannot move anymore due to the surrounding of pursuers, there are two kinds of task allocations. One is about assigning which evader to which group of pursuers so that all evaders can be captured. The other is about assigning which capturing position to which pursuer to encircle the evader simultaneously. In this chapter, the MEP is modeled as a dynamic optimization problem and each its time step is solved by a two-stage approach: BiPCCR (BiQAP-PCCPSOR). Firstly, the first kind of task allocation problem is modeled as the biquadratic assignment problem (BiQAP) and a MEP fitness function is proposed for the evaluation of such BiQAP task allocations. In this way, the MEP is transformed to several single-evader pursuit (SEP) problems. Secondly, for each SEP, this chapter extends the coordinated SEP strategy CCPSO-R (cooperative coevolutionary particle swarm optimization for robots) to its parallel version as PCCPSO-R to enable the parallel implicit capturing position allocating by parallel observation, decision making, and moving of pursuers. Through experiments of the current BiQAP solvers on the task allocation, this chapter improves the best one of them in statistic based on the domain knowledge. Moreover, the advantages of PCCPSO-R in the capturing efficiency over

CCPSO-R is testified in the MEP experiments.

4.2 Biquadratic assignment problem

The biquadratic assignment problem (BiQAP) [18, 20, 64] of size n is a combinatorial optimization problem, whose solution is typically represented as the permutation φ, which is the bijective mapping between two sets $A = \{1, ..., n\}$ and $B = \{1, ..., n\}$, with the objective being

$$\min_{\varphi \in S} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{o=1}^{n} c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)}, \tag{4.1}$$

where S is the set of all permutations of $\{1, 2, ..., n\}$, $\varphi(i) \in \{1, 2, ..., n\}$ is the i-th dimension of the permutation, and $c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)} \in \Re^{n^8}$ is the cost of assigning the numbers i, j, k, o to $\varphi(i), \varphi(j), \varphi(k), \varphi(o)$, respectively.

In particular, BiQAP is a nonlinear assignment problem, a special case of M-adic assignment problem when $M = 4$, and a generalization of the quadratic assignment problem (QAP). First, Burkard et al. [18] investigated BiQAP motivated by the problem in the VLSI synthesis, which, to the best knowledge of authors, is the only one published application of BiQAP so far. Meanwhile, they proposed the first method for generating BiQAP benchmarks with known optimal solutions. Afterwards, since BiQAP is NP-complete [18, 64], heuristic methods are pursued and reported in literature. In [17], three deterministic improvement methods, i.e., the first improvement method (FIRST), the best improvement method (BEST), and the Heider's improvement method (HEIDER); three simulated annealing algorithms (SIMANNs); one taboo search method; and one hybrid of taboo search and SIMANN were proposed. Then, in [52], a greedy randomized adaptive search procedure (GRASP) was proposed, one iteration of which first greedily constructs an initial solution and then FIRST is applied as the local search method.

According to the experimental results of [17, 20, 52] on benchmarks generated by the method in [18], HEIDER is the best among the deterministic methods in terms of the tradeoff between the solution quality and the computational cost, and GRASP is the only one algorithm that finds optimal solutions for all test instances 100% with the cost of more computations. However, for real-world applications like the MEP BiQAP task allocation, GRASP may be not the first-class choice, and the simpler the solver is the better, as will be shown by the experimental results of Section 4.4.

4.3 The proposed two-stage approach: BiPCCR

The multiple-evaders pursuit (MEP) occurs in a finite grid world where 5 possible actions of agents: moving north, west, south, and east one step away, and staying still. The capture of an evader is defined as that the evader cannot move any more due to the surrounding of pursuers, as illustrated in Fig. 2.2. So, in general, 4 pursuers are needed for the capture of one evader. Once an evader is captured, it will not disappear. The MEP is said to be a success if all the evaders are captured.

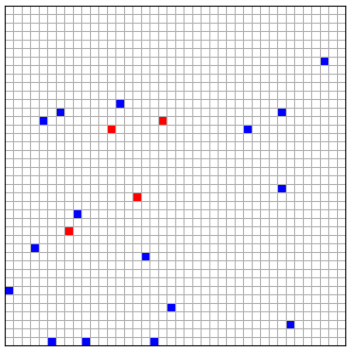


Figure 4.1: An example multiple-evaders pursuit scenario.

Algorithm 4.1: Proposed two-stage approach: BiPCCR to MEP

```
Initialize the environment.
for each time step do
    for each evader do
        Observe, make decision, and take one action.
    The central virtual pursuer observes and makes the BiQAP task allocation
      decision by Algorithm 4.3.
    All the real pursuers concurrently observe, make decision, and take one action
      by PCCPSO-R.
    if termination conditions are satisfied then
        The game is terminated.
```

For one time step in MEP, as shown in Fig. 4.1, firstly, tasks need to be allocated that each evader is assigned to each 4 pursuers and the MEP problem is transformed to several single-evader pursuit (SEP) problems. Due to the evaluation of such a task allocation involves the assignments between an evader and 4 pursuers, it can be naturally modeled as the biquadratic assignment problem (BiQAP). Secondly, all the pursuers will concurrently cooperate with their group members to capture the assigned evader where

a distributed coordinated single-evader pursuit strategy CCPSO-R [83] will be modified to its parallel version PCCPSO-R to improve the scalability. As the time goes, both evaders and pursuers move, the environment changes, and thus both each single-evader pursuit fitness and the task allocation fitness change. So, for the MEP game, it is in fact a dynamic optimization problem that the centralized BiQAP task allocation and decentralized PCCPSO-R are conducted every time step due to the past environmental changes, until all evaders are captured or time limit is reached. As a whole, the proposed two-stage approach to MEP is described in Algorithm 4.1, which will be introduced in detail in Section 4.3.1 and 4.3.2.

4.3.1 BiQAP task allocation in the dynamic optimization

For the MEP, a central virtual pursuer is designed to allocate each 4 pursuers to the pursuit of a evader. Specially, if the set of pursuers is represented by $PURSUERS = \{1,2,..,n\}$, and the set of evaders is represented by $EVADERS = \{1,2,...,m\}$ with $n = 4m$, then the mapping from $PURSUERS$ to $EVADERS$ is a biquadratic semi-assignment problem (semi-BiQAP) [16, 64] since 4 pursuers will be assigned to the same prey.

Equivalently, the semi-BiQAP between $PURSUERS$ and $EVADERS$ can be transformed to the BiQAP between $PURSUERS$ and $\overline{EVADERS}$ by repeating each prey 4 times, i.e., $\overline{EVADERS} = \{1,...,n\} = \{1,1,1,1,...,m,m,m,m\}$. Futher, since the mapping between $PURSUERS$ and $\overline{EVADERS}$ is bijective, the problem is equivalent to the BiQAP between $\overline{EVADERS}$ and $PURSUERS$ with the objective (4.1) which is rewritten here for convenience.

$$\min_{\varphi \in S} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{o=1}^{n} c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)}. \tag{4.2}$$

Moreover, for the practical application of MEP, the matrix $[c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)}]$ is sparse that

$$c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)} \begin{cases} \neq 0, & \text{if } (i,j,k,o) \in K_1, \\ = 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where $K_1 = \{C(w,w+1,w+2,w+3) | w \in \{1,5,...,n-3\}\}$ and $C(\cdot,\cdot,\cdot,\cdot)$ is all the combinations of the 4 numbers. For example, when $w = 1$,

$$K_1 = \{C(0,1,2,3)\} = \{(0,1,2,3),(0,1,3,2),(0,2,3,1),...,(3,2,1,0)\}$$

and $|K_1| = 24$.

Therefore, the objective (4.2) can be simplified as

$$\min_{\varphi \in S} \sum_{(i,j,k,o) \in K_1} c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)}. \tag{4.4}$$

Further, since each evader is repeated 4 times in $\overline{EVADERS}$, the cost is the same for all $(i,j,k,o) \in C(w,w+1,w+2,w+3)$ for any fixed w. So, the matrix $[c]$ is symmetric and the optimization problem of (4.4) is equivalent to

$$\min_{\varphi \in S} \sum_{(i,j,k,o) \in K_2} c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)}, \tag{4.5}$$

where $K_2 = \{(w,w+1,w+2,w+3) | w \in \{1,5,...,n-3\}\}$.

So, when the permutation φ is given, (i) due to the sparsity of the matrix $[c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)}]$, only $|K_1| = 4! \cdot n/4 = O(n)$ non-zero elements need to be considered , among which only $|K_2| = n/4$ elements are distinguished due to the matrix symmetry; (ii) to evaluate a solution φ, $n/4$ additions of the cost coefficients $c_{i,j,k,m,\varphi(i),\varphi(j),\varphi(k),\varphi(o)}$ will be needed.

As for the cost coefficient, $c = f^i$, where f^i is the fitness function for the i-th single-evader pursuit. Therefore, this research proposes the fitness function for the MEP as

$$f = \sum_{i \in \{1,2,...,n/4\}} f^i, \tag{4.6}$$

and the MEP BiQAP task allocation (4.5) is equivalent to

$$\min f. \tag{4.7}$$

In a summary, the one time step task allocation problem of assigning each evader to 4 pursuers is modeled as the BiQAP, which is an optimization problem with the objective (4.7). According to the convention in solving BiQAP, a solution is represented as a permutation of $\{1,2,...,n\}$. As for the MEP, it is a dynamic optimization problem where the fitness function Eq (4.6) changes every time step since the agents (pursuers and evaders) are moving. To solve such dynamic optimization problem, based on the domain knowledge, this research proposes a scheme to greedily construct an initial good solution based on the space distributions of agents in Algorithm 4.2. However, note that, this procedure is conducted only once at the beginning of the game, while for successive time steps, since the MEP is a slow-changing dynamic optimization problem, the best solution found in the last time step is used as the initial solution for the task allocation of the current time step.

Algorithm 4.2: Greedy inidividual construction scheme

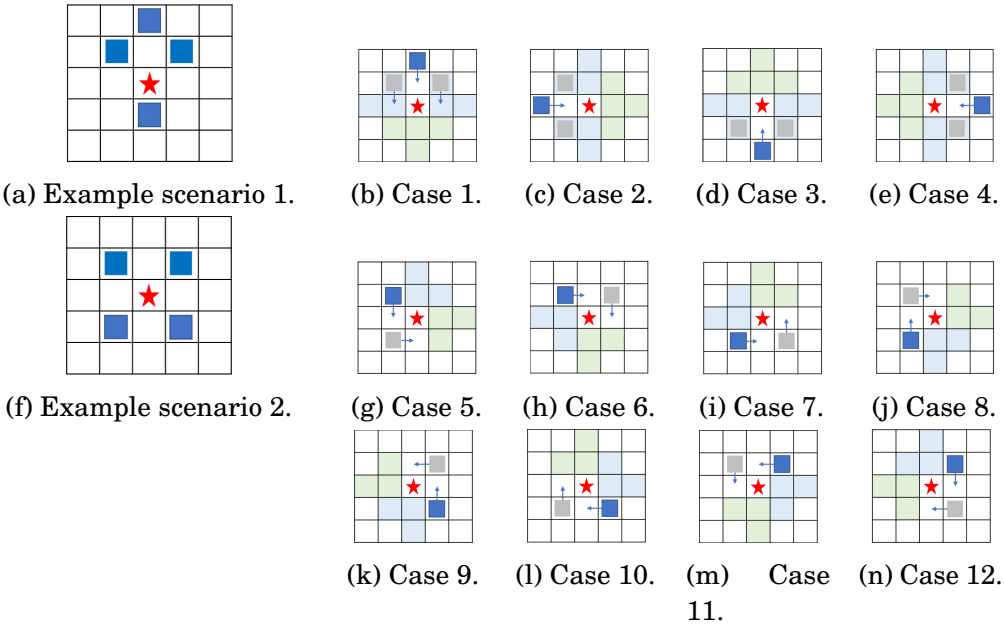
```
1  M ← pairwise distances between evaders and pursuers.
2  for each evader do
3      Assign the nearest 4 pursuers to the evader.
4      Delete these 4 pursuers from M.
```

Algorithm 4.3: HEIDER-Random [17]

```
1  Construct an initial solution by Algorithm 4.2 as the current solution and set the
     flag converged as False.
2  Generate the two-positions interchanging set
     I = {(i,j)|i,j ∈ {1,...,n}, i < j, mod(i,4) ≠ mod(j,4)} and randomize its elements
     order for the semi-neighborhood searching.
3  while not converged do
4      Set the flag neighborhood_exhausted as False.
5      while not neighborhood_exhausted do
6          Find the next neighbor by exchanging the i-th and j-th positions of the
             current solution where (i,j) is the next element of I in the cyclic way.
7          if the neighbor's fitness is better than the current solution then
8              neighborhood_exhausted ← True.
9              The current solution ← the neighbor.
10         else if I is traversed then
11             neighborhood_exhausted ← True.
12             converged ← True.
13 Output the current solution and its fitness.
```

For solving the BiQAP task allocation, this research modifies HEIDER as HEIDER-Random to stochastically improve the solution's quality and the algorithm's efficiency by two tricks in the line 2 of Algorithm 4.3: (i) randomize the neighborhood searching order in I; (ii) remove the ineffective neighborhood searching and name it as the semi-neighborhood searching due to the inherent semi-assignment nature of the MEP. That is, this research does not search the neighbor derived from interchanging the i-th and j-th positions of the current solution when the conditions $i,j \in \{1,...,n\}, i < j, mod(i,4) \neq mod(j,4)$ are not satisfied. This is because interchanging the relative orders of two pursuers assigned to the same evader in the permutation solution does not change the single-evader pursuit fitness f^i and thus the task allocation fitness f.

4.3.2 Parallel CCPSO-R algorithm for each single evader pursuit



(a) Example scenario 1. (b) Case 1. (c) Case 2. (d) Case 3. (e) Case 4.

(f) Example scenario 2. (g) Case 5. (h) Case 6. (i) Case 7. (j) Case 8.

(k) Case 9. (l) Case 10. (m) Case 11. (n) Case 12.

Figure 4.2: The local capture patterns in the repair stage of PCCPSO-R.

As shown in Algorithm 3.1 and mentioned in [83], the sequential scheme in the observation, decision making, and moving of real pursuers limits the scalability of CCPSO-R. Therefore, this research proposes a parallel version of CCPSO-R, which is named as PCCPSO-R, by parallelizing the subpopulation-by-subpopulation procedure of the *for* loop in Algorithm 3.1 in the following ways.

First, the single-evader pursuit function Eq (3.2) is modified as

$$f^{ij} = f^{ij}_{repel-evader} \cdot f^{ij}_{repel-pursuer} \cdot (f^{ij}_{closure} + f^{ij}_{expanse} + f^{ij}_{uniformity}) \tag{4.8}$$

where $f^{ij}_{repel-evader}$ and $f^{ij}_{repel-pursuer}$ have the same form of Eq (3.3) yet with different meanings of NND^{ij} and different values of the Manhattan distance D_{min}. For $f^{ij}_{repel-evader}$, NND^{ij} is the nearest distance to evader agents and the secure distance D_{min} is set as 1, while for $f^{ij}_{repel-pursuer}$, NND^{ij} is the nearest distance to pursuer agents and D_{min} is 2. However, note that, the function Eq (4.8) is for a pursuer's fitness in capturing an evader, while to evaluate the capturing fitness f^i of the whole pursuer swarm, $f^{ij}_{repel-evader} = f^{ij}_{repel-pursuer} = 1$.

Second, a repair stage is added when the pursuer swarm and their assigned evader is close enough in a limited area such that strong and obvious capturing patterns can

be matched and more effective simple rules can be applied. In particular, for scenarios like that shown in Fig. 4.2a, 4 local capture patterns Fig. 4.2b to Fig. 4.2e are extracted, while for scenarios like that shown in Fig. 4.2f, 8 patterns Fig. 4.2g to Fig. 4.2n are extracted. In these pattern cases, the red pentagram is the evader, the blue square with an moving direction arrow is the current pursuer, the grey square is also a pursuer that will prevent the current pursuer from moving to the next capturing position according to the configurations in Eq (4.8), and the shallow blue and shallow green filled cells represent the potential positions that a one-step away pursuer may locate for each distinct capturing position. Therefore, once a pursuer detects a matched local capturing pattern, it will enter the repair stage and behave according to the pointed moving direction such that all pursuers will move according to the rules to capture the evader without collisions.

4.4 Experiments

4.4.1 Experiment 1 (BiQAP solver for the task allocation)

For the comparison of BiQAP solvers, test instances are randomly generated in a 40×40 grid world with different problem sizes, which is the length of a permutation solution, i.e., the number of pursuers, or 4 times of the number of evaders. For each test instance, an initial good solution is constructed using the method in Algorithm 4.2 whose fitness value is evaluated by the MEP fitness function Eq (4.6) and listed in the "Initialization" column of Table 4.1. For GRASP, $\Lambda = 1000, \alpha = 0.25$ and $\beta = 0.3$ are used as in [20, 52]. For SIMANN3, its initial temperature is 2000 for the problems of size 12 and 5000 for the problems of size 16 as in [17]. The semi-neighborhood searching in Algorithm 4.3 is used in all the BiQAP solvers.

The experimental results are presented in Table 4.1 and Table 4.2, where the symbol "-" means that the corresponding experiments are not conducted due to the algorithm's high computational cost in terms of the time limit (0.5s here) of the MEP BiQAP task allocation. Note that, for deterministic algorithms FIRST, BEST, and HEIDER, their results are deterministic if the initial solution is given, while for stochastic algorithms GRASP, SIMANN3, and HEIDER-Random, since there are other random factors other than the initial solution, their results over independent runs may be different. Here, the best results and their ratios are listed in the corresponding parentheses; meanwhile, the average results and standard deviations are listed in the "avg." columns and the corre-

Table 4.1: Solutions qualities obtained by different algorithms on the BiQAP task allocation problems over 50 runs

Size	Instance	Initialization	FIRST	BEST	HEIDER	GRASP		SIMANN3		HEIDER-Random	
						best	avg.	best	avg.	best	avg.
12	1	56.984	**47.080**	47.419	**47.080**	**47.080** (0.300)	47.424 (0.318)	47.083 (0.020)	48.697 (0.491)	**47.080** (0.260)	47.364 (0.274)
	2	54.900	49.375	**48.376**	49.375	**48.376** (0.640)	48.735 (0.480)	49.985 (1.000)	49.985 (0.000)	**48.376** (0.660)	48.715 (0.473)
	3	58.935	**53.614**	**53.614**	**53.614**	**53.614** (0.920)	53.692 (0.266)	54.596 (1.000)	54.596 (0.000)	**53.614** (0.680)	53.810 (0.393)
	4	56.641	**54.336**	**54.336**	**54.336**	**54.336** (1.000)	**54.336** (0.000)	54.344 (0.020)	54.713 (0.053)	**54.336** (1.000)	**54.336** (0.000)
	5	58.213	**45.297**	**45.297**	**45.297**	**45.297** (1.000)	**45.297** (0.000)	**45.297** (0.060)	48.891 (0.939)	**45.297** (1.000)	**45.297** (0.000)
16	1	72.757	55.380	55.380	55.380	-	-	60.876 (0.020)	64.701 (0.546)	**54.745** (0.760)	54.898 (0.271)
	2	77.096	60.922	60.922	60.922	-	-	70.776 (0.100)	71.709 (0.311)	**60.689** (0.600)	60.792 (0.129)
	3	71.025	**64.225**	**64.225**	**64.225**	-	-	65.746 (0.060)	66.349 (0.152)	**64.225** (0.200)	**64.225** (0.000)
	4	56.182	46.444	**45.958**	46.444	-	-	48.256 (0.020)	48.451 (0.028)	**45.958** (0.140)	46.251 (0.183)
	5	59.543	50.753	50.753	50.753	-	-	52.562 (0.020)	58.874 (1.127)	**50.118** (0.220)	50.575 (0.285)
Mean rank			1.53	1.67	1.6	1		2.4		1	

sponding parentheses. The "Mean rank" row is calculated according to the algorithms' best results. It can be seen that HEIDER-Random is the best in terms of the probability to find the best known solutions and converge with less computational cost.

In addition, note that, all the searching algorithms here, either deterministic or stochastic, are iterative algorithms. That is, the current iterative procedure depends on the results of the previous iteration, the process of which is sequential. Therefore, the algorithm's runtime is mainly determined by the number of iterations and the time in evaluating a solution per iteration. For the BiQAP task allocation of size $n = 4 \cdot m$, the time to evaluate a permutation solution is $m \cdot t(f^i)$ where m is the number of evaders and $t(f^i)$ is the time in evaluating f^i of Eq (4.6). As shown in Fig. 4.3, $t(f^i) \approx 0.0005$s on a Macbook Pro with a 2.9 GHz quad-core Intel core i7 and a 16 GB memory. So, roughly, at most 1000 sequential calculations of f^i, i.e., $1000/m$ algorithm's iterations or MEP fitness evaluations Eq (4.6) are allowed if only one solution is evaluated per iteration, as listed in the "MaxIter" column in Table 4.2. Hence, for the problems of size 12, all deterministic algorithms FIRST, BEST, HEIDER, and HEIDER-Random can converge, while for the problems of size 16, only HEIDER and HEIDER-Random can converge.

Table 4.2: No. of MEP fitness evaluations (4.6) of algorithms on the BiQAP task allocation problems over 50 runs

Size	MaxIter	Instance	FIRST	BEST	HEIDER	GRASP		SIMANN3		HEIDER-Random	
						best	avg.	best	avg.	best	avg.
12	333	1	221	193	87	1632.667 (0.020)	1736.367 (62.261)	694 (1.000)	694 (0.000)	**68** (0.020)	97.600 (20.134)
		2	165	145	95	1562.667 (0.020)	1742.207 (96.251)	694 (1.000)	694 (0.000)	**57** (0.040)	80.060 (14.026)
		3	223	193	92	1624.667 (0.020)	1826.987 (112.164)	694 (1.000)	694 (0.000)	**61** (0.040)	85.860 (15.242)
		4	253	193	97	1627.667 (0.020)	1866.027 (127.177)	694 (1.000)	694 (0.000)	**65** (0.020)	87.560 (12.384)
		5	225	241	106	1637.667 (0.020)	1912.047 (122.529)	694 (1.000)	694 (0.000)	**63** (0.020)	89.880 (11.117)
16	250	1	1286	865	253	-	-	1261 (1.000)	1261 (0.000)	**138** (0.020)	217.580 (34.527)
		2	1048	673	266	-	-	1261 (1.000)	1261 (0.000)	**134** (0.020)	223.300 (61.364)
		3	691	673	245	-	-	1261 (1.000)	1261 (0.000)	**136** (0.020)	217.040 (40.249)
		4	1214	577	268	-	-	1261 (1.000)	1261 (0.000)	**127** (0.020)	194.380 (41.212)
		5	272	673	196	-	-	1261 (1.000)	1261 (0.000)	**147** (0.020)	210.720 (38.405)
Mean rank			3.86	3.13	2	6		5		1	

In contrast, the computational cost of GRASP and SIMANN3 are too high to converge within 0.5s even for the problems of size 12.

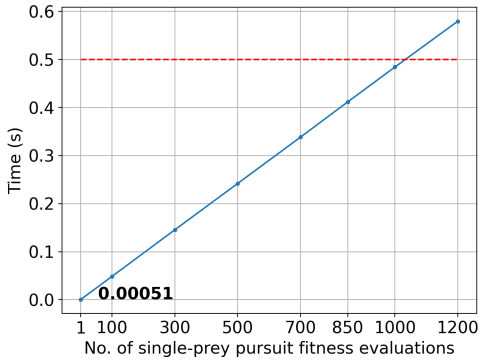


Figure 4.3: The fitness evaluation time.

Take the test instance 5 of the problem size 12 as an example. The BiQAP task allocation fitness values of the permutation solutions over iterations are averaged over the 50 independent runs and plotted in Fig. 4.4. It can be seen that, the greedy initial solution construction stage of GRASP takes too much computational cost and the greedily

constructed initial solution may be not as good as the initial solution constructed from the domain knowledge as proposed in Algorithm 4.2. So, the most contributed component of GRASP, i.e., the solution construction stage, may be unnecessary in real-world applications when better initial solutions can be constructed by practical domain knowledge. On the other hand, SIMANN3 is sensitive to its parameters settings. First, it takes efforts to determine its initial temperature, which additionally may be different with the problem size. Second, although its optimal temperature value can be determined automatically, the automation process itself also takes much computational cost. Therefore, deterministic algorithms are more favorable for the BiQAP task allocation since they are the simplest algorithms with the fastest converging speed when a good enough initial solution can be given, among which HEIDER is the best in terms of the trade-off between the solution quality and the computational cost. Since HEIDER is a special case of HEIDER-Random with a fixed neighborhood searching cyclic order, HEIDER-Random, which adopts a random searching order, has the probability to be superior than the original simplest version of HEIDER.

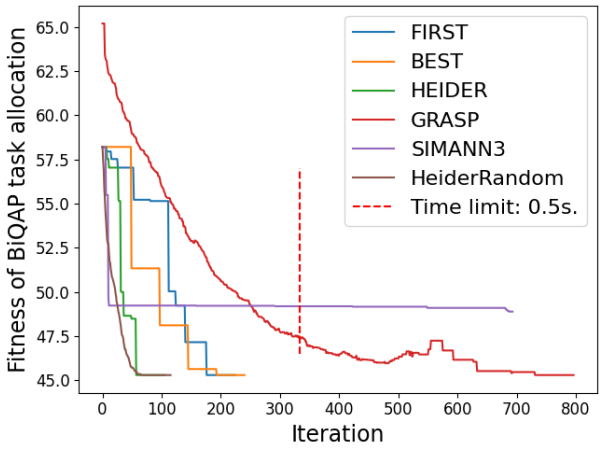


Figure 4.4: The averaged fitness values of different algorithms during the optimization process over 50 runs.

4.4.2 Experiment 2 (PCCPSO-R vs. CCPSO-R)

In this section, the effectiveness of PCCPSO-R over CCPSO-R on the multiple random walking evaders pursuit problems are compared. The environmental instances in Section 4.4.1 is used as the initialization, while the HEIDER-Random algorithm are used as the BiQAP task allocation solver, and maximal 1000 times steps are allowed to capture all evaders. The experimental results are shown in Table 4.3 where the average steps taken

to terminate the game and the standard deviations of the steps in all the 50 independent runs are listed. Both CCPSO-R and PCCPSO-R achieve the 100% capture rate. It can be seen that PCCPSO-R has a higher efficiency than CCPSO-R due to the introducing of the local capture patterns and capturing rules in Section 4.3.2. However, due to the parallel nature in agents' observations, decision makings, and movements, it is not guaranteed that no collisions in using PCCPSO-R.

Table 4.3: Comparison of the multiple-evaders pursuit efficiency of PCCPSO-R and CCPSO-R

Problems	3-evaders pursuit (BiQAP task allocation size: 12)				
	1	2	3	4	5
CCPSO-R	125.38 (65.098)	122.76 (63.864)	97.68 (47.549)	122.76 (89.185)	115.88 (62.588)
PCCPSO-R	56.28 (25.499)	48.18 (12.388)	57.72 (17.266)	62.74 (26.173)	49.32 (23.835)
Problems	4-evaders pursuit (BiQAP task allocation size: 16)				
	1	2	3	4	5
CCPSO-R	120.14 (61.477)	145.54 (85.455)	128.28 (58.538)	126.62 (64.839)	132.36 (74.068)
PCCPSO-R	60.14 (22.107)	49.96 (10.849)	54.22 (30.190)	47.36 (23.606)	53.46 (30.517)

4.5 Summary

In this chapter, the MEP is modeled as a dynamic optimization problem where each time step is solved by the proposed two-stage approach. In particular, the first stage is to centrally assign each evader to 4 pursuers by modeling such task allocation problem as the BiQAP, the solutions of which are evaluated by the proposed multiple-evaders pursuing fitness function. In this way, the MEP is transformed to several SEPs. In the second stage, each SEP is simultaneously solved by the assigned pursuers through PCCPSO-R, which parallelizes CCPSO-R to further improve the scalability of the pursuer's strategy to cooperatively capture the assigned evader. Since the MEP is a slow-changing problem, the BiQAP task allocation solution found in the previous time step is used as the initial solution for the current time step, while the initial solution for the first time step is constructed by the proposed greedy initial solution construction procedure based on the domain knowledge.

As for the solving of the BiQAP task allocation, due to the time limit (0.5s here) in MAS, more complex and more powerful heuristic algorithms are inferior to the simplest deterministic local search algorithms when a good enough initial solution can be given. Even if the population of swarm intelligence algorithms (SIAs) is parallel, enough generations are needed to find good enough solutions, the process of which is sequential and thus hard to be achieved in the time limit. Therefore, based on the domain knowledge, this research has proposed a greedy procedure to construct a good enough initial solution for the BiQAP task allocation, and modified HEIDER as HEIDER-Random by adding tricks and integrating application considerations.

However, there are still limitations in the proposed approach to MEP. First, the centralized BiQAP solution to the task allocation is still not applicable to large-scale problems due to the time limit in MAS. Second, although PCCPSO-R enable the parallel observation, decision making and moving of pursuers by introducing two minimal secure distances in Eq (4.8), local capture patterns, and effective rules, these patterns are not complete to involve all possible local capturing scenarios.

CHAPTER 5

MULTIPLE EVADERS PURSUIT WITH PARTIAL OBSERVATION

5.1 Background

This chapter investigates the dynamic pursuit problem between multiple evaders and a swarm of partially observable pursuers in the bounded grid world without inter-agent communications, i.e., the self-organizing pursuit (SOP) problem. This work proposes a framework for decentralized multi-agent systems to improve the implicit coordination capabilities in search and pursuit. A self-organizing system is modeled as a partially observable Markov game (POMG) featured by large-scale, 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 (MARL) method that enables homogeneous agents to learn natural SOS patterns. Additionally, a fuzzy-based distributed task allocation method is proposed, 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 by decomposing the SOP task, FSC2 achieves superior performance compared with other implicit coordination policies fully trained by general MARL algorithms. The scalability of FSC2 is proved that up to 2048 FSC2 agents perform efficient multi-target SOP with almost 100% capture rates. Empirical analyses and ablation studies verify the interpretability, rationality, and effectiveness of component algorithms in FSC2.

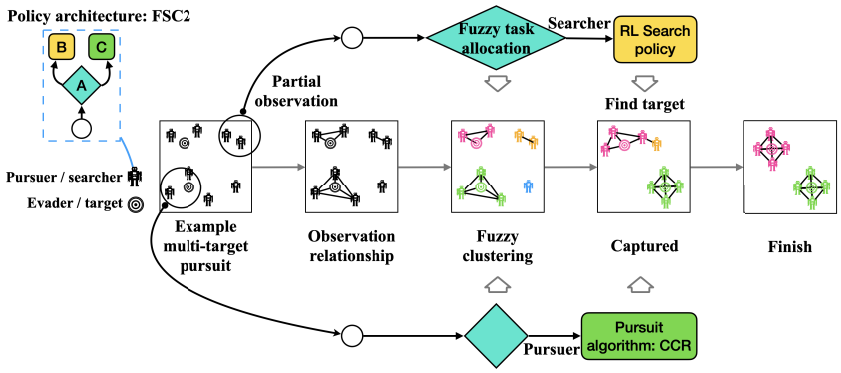


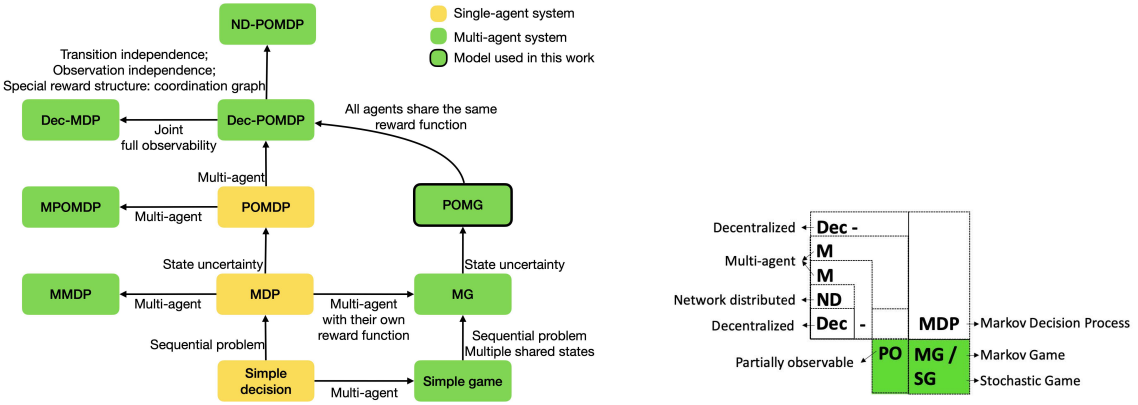
Figure 5.1: **Multi-target self-organizing pursuit task and computational framework of FSC2** (fuzzy self-organizing cooperative coevolution). FSC2 is a distributed framework that consists of three modules: **A. fuzzy task allocation:** takes partial observation as input, computes distributed fuzzy clusters of agents and targets, and determines the current role of agent to be either a searcher or a pursuer; **B. RL search policy:** a searcher searches the space to find targets; and **C. pursuit algorithm (CCR):** a pursuer cooperates with cluster members to pursue the targeted evader of its cluster.

5.2 Problem formulation

5.2.1 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 5.2a [56], and

the common nomenclature for the model name abbreviations is presented in Figure 5.2b.



(a) Common multi-agent problem formulations [56].

(b) Common nomenclature for multi-agent models.

Figure 5.2: Common multi-agent problem formulations and their nomenclature.

A definition of self-organization was given in [19]: 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, a self-organizing system can be formulated as a POMG [56]: $\langle \gamma, \mathscr{I}, \mathscr{S}, \mathscr{A}, \mathscr{O}, P, O, R \rangle$. γ is the discounted factor for return; $\mathscr{I} = \{1, ..., n\}$ represents all total n agents; $\mathscr{S} = \{s\}$ is the true state space; $\mathscr{A} = \mathscr{A}^1 \times ... \times \mathscr{A}^n = \{\vec{a}\}$ is the joint action space; $\mathscr{O} = \mathscr{O}^1 \times ... \times \mathscr{O}^n = \{\vec{o}\}$ 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 the POMG rather than the Dec-POMDP (decentralized partially observable Markov decision process) is used 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.

5.2.2 The problem of self-organizing search and pursuit

A typical multi-target search and pursuit scenario is illustrated in Figure 5.3. Due to the

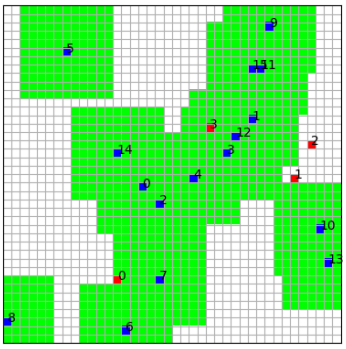


Figure 5.3: A screenshot of self-organizing search and pursuit in a bounded grid world, where red squares are targets or evaders, blue squares are searchers or pursuers, and green background around each agent shows its perception range with an inf-norm radius of 5.

partial observation and communication limitation of agents, this research distinguishes 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.

- **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 5.1, i.e., the space exploring ability that will be used in the SOP task. The SOS task is designed 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, this research specially defines 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, this research investigates 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.

5.3 Fuzzy self-organizing cooperative coevolution (FSC2) algorithm

In this section, this research introduces in detail the proposed distributed hierarchical framework: fuzzy self-organizing cooperative coevolution (FSC2) in Figure 5.1. 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.

The whole algorithm of FSC2 is given in Algorithm 5.1. In the multi-target pursuit environment, targets and partial observable agents are distributed in the space. First, two alternate basic roles of an agent are assumed: searcher or pursuer, based on the existence of free targets that are not captured in the agent's neighborhood. Then, agents are clustered in a distributed way 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 5.3.1, and Figure 5.1 gives an illustrative clustering result. After clustering, an agent alternates between the second module: search policy in Section 5.3.2 and the third module: pursuit algorithm (CCR) in Section 5.3.3, based on its real-time neighborhood.

Algorithm 5.1: FSC2 for each agent in the SOP

```
while the termination conditions are not satisfied do
    role, cluster_center, cluster_members, Memory ← Fuzzy clustering
     (Algorithm 5.2 in Section 5.3.1).
    if role is a searcher then
        As an SOS agent (Section 5.3.2), find free targets.
    else if role is a pursuer then
        As a CCR agent (Section 5.3.3), cooperate with cluster_members in
         pursuing cluster_center.
```

5.3.1 Distributed fuzzy clustering for task allocation

A pursuer is defined to be 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 are required to capture a 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, this research adopts the fuzzy clustering and utilizes 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, its cluster center shall be one of all its m^k local free targets $T = \{T_1, ..., T_{m^k}\}$, while its 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

$$\mu_{ij}^k = \frac{(||A_i - T_j||_1^2)^{\frac{1}{1-\alpha}}}{\sum_{j=1}^{m^k}(||A_i - T_j||_1^2)^{\frac{1}{1-\alpha}}} \in [0,1], \tag{5.1}$$

where $\alpha > 1$ is the fuzzifier [12], the value of which is 1.5 in our experiments. Thus, agent k can obtain its fuzzy membership matrix

$$M^k = [\mu_{ij}^k] \in R^{n^k \times m^k}, \tag{5.2}$$

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

$$\hat{M}^k \sim M^k, \tag{5.3}$$

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

$$T_c|_{\hat{M}_{kc}^k \neq 0, c=1,...,m^k}, \tag{5.4}$$

while agent k's cluster members are the pursuers

$$\{A_i | \hat{M}_{ic}^k \neq 0, i = 1,...,n^k\}. \tag{5.5}$$

The distributed fuzzy clustering based task allocation process in Equation (5.1) to (5.5) is summarized in Algorithm 5.2.

Agent memory. Note that, each agent's $Memory$ of the environment (line 1 of Algorithm 5.2) 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 (5.1), the number of members in each cluster is not specified in Equation (5.5). 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 5.2: 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 (5.4).
    $cluster\_members \leftarrow$ Equation (5.5).

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

$$DC \doteq \frac{2}{n \cdot (n-1)} \sum_{i=1}^{n-1} \sum_{j=i}^{n} \frac{|\{k | k \in \hat{C}_i \cap \hat{C}_j, \text{and } c_{ik} == c_{jk}\}|}{|\hat{C}_i \cap \hat{C}_j|} \in [0, 1], \tag{5.6}$$

where $|\cdot|$ is 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 (5.6) 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, $0/0 = 1$ is defined for Equation (5.6), which means that two agents without local physical interactions have fully consistent decisions.

5.3.2 Self-organizing search (SOS) policy

In the self-organizing 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 [67];
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, this research uses the actor-critic algorithm [43] 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, respectively) according to

$$\theta = \theta + \alpha_1 \nabla_\theta J(\pi_\theta), \tag{5.7}$$

where

$$\nabla_\theta J(\pi_\theta) = E_{\tau \sim \pi_\theta}[\sum_{t=0}^{tmax} \nabla_\theta \log \pi_\theta(a_t|s_t) A_t], \tag{5.8}$$

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) [74] in the form of

$$A_t = \sum_{l=0}^{tmax-t} (\gamma\lambda)^l \delta_{t+l}^V, \tag{5.9}$$

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

$$\delta_t^V = R(s_t, \vec{a}_t) + \gamma V_\phi(s_{t+1}) - V_\phi(s_t). \tag{5.10}$$

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):

$$\phi = \arg\min_{\phi} E_{s_t, \hat{R}_t \sim \pi_\theta}[(V_\phi(s_t) - \hat{R}_t)^2]. \tag{5.11}$$

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 5.1. 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 5.1: Reward function $R^i(s_t, \vec{a}_t)$ for self-organizing 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

This research once tries 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. It is assumed 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 a reward is only given to the contributing agent that finds the target, as in Table 5.1, 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.

5.3.3 Cooperative coevolutuion algorithm for robots (CCR)

According to FSC2 (Algorithm 5.1), 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, the CCR (cooperative coevolution for robots) algorithm is proposed based on CCPSO-R [83, 84], which further improves the cooperation of pursuers in their simultaneously decision making and execution process.

Cooperative coevolutionary evaluation scheme. Similar to CCPSO-R [83], 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 by 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 [83] as follows:

$$f_{stp}^{ij} = f_{closure}^{ij} + f_{expanse}^{ij} + f_{uniformity}^{ij}, \tag{5.12}$$

where

$$f_{closure}^{ij} = inconv(T_c, A_1, ..., A_i^j, ..., A_{n^i}) \tag{5.13}$$

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;

$$f_{expanse}^{ij} = \frac{1}{n^i}\Big(\sum_{k=1,k\neq i}^{n^i} ||A_k - T_c||_1 + ||A_i^j - T_c||_1\Big) \tag{5.14}$$

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

$$f_{uniformity}^{ij} = std\left(\begin{bmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{bmatrix}\right) \tag{5.15}$$

or

$$f^{ij}_{uniformity} = std([N_{12}, N_{21}, N_{23}, N_{32}]) + std([N_{11}, N_{13}, N_{31}, N_{33}]). \tag{5.16}$$

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 [83]).

However, f^{ij}_{stp} only solves the cooperative single-target pursuit problem by letting agents make decisions sequentially, while its parallel decision-making version PCCPSO-R [84] can only resolve partial conflicts by introducing two secure distances in the fitness function. Hence, a new fitness function is proposed based on f^{ij}_{stp} 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^j_i can be defined as

$$f^{ij} = \begin{cases} \infty, & \text{if } nnd^{ij}_{entity} == 0 \text{ or} \\ & (nnd^{ij}_{target} \neq 1 \ \& \ nnd^{ij}_{pursuer} == 1) \\ f^{ij}_{convention}, & \text{else if } nnd^{ij}_{target} == 1 \ \& \\ & nnd^{ij}_{pursuer} == 1 \\ f^{ij}_{stp}, & \text{else} \end{cases} \tag{5.17}$$

where nnd^{ij}_{entity} 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 ($nnd^{ij}_{target} == 1$ & $nnd^{ij}_{pursuer} == 1$) 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 $f^{ij}_{convention}$ 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.

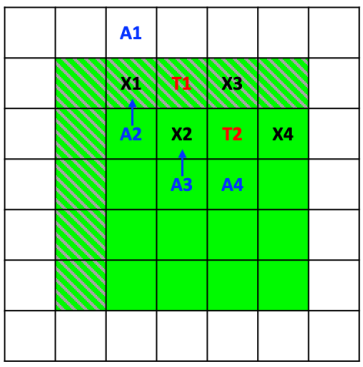
If the next candidate position or virtual agent A_i^j of the current real pursuer A_i is its assigned capturing position under a certain partial observation, $f_{convention} = -1$; otherwise, $f_{convention} = \infty$, which means that the choice not satisfying the lexicographic convention is not allowed.

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 5.4. 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, the current agent is prevented 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.

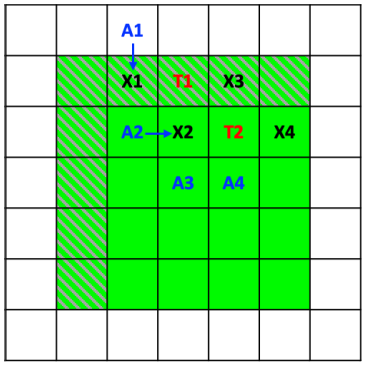
5.4 Experiments

5.4.1 Environments, baselines, and experimental setups

Environments First, for the convenience in comparing the self-organizing search (SOS) agents trained by different MARL algorithms with their official public code, several changes were made to the PettingZoo Pursuit-V3 environment [89], including the initialization, reward function, some utility functions, and bugs. Second, for the multi-target self-organizing pursuit (SOP), the environment is implemented by ourselves with more compact code and adjusted to the self-organizing game setups. 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.



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



(b) Actual decisions of pursuers.

Figure 5.4: Illustration of uncertain partial observation under the lexicographic convention of Section 5.3.3; 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 5.3.3. 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.

Baselines for SOS In the performance comparison of self-organizing search (SOS), the actor-critic trained search policy are compared with the following search strategies.

- A swarm of independent random-walk searchers: Each searcher randomly walks in the space, taking no account of its surroundings and past history.
- 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, which is NP-complete [30]. For simplicity, an intuitive systematic strategy is employ, 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. 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.

- A swarm of ApeX-DQN searchers, the current documented best performing MARL in pursuit [90]: This work tested the learning rates {10^{-6}, $\mathbf{10^{-5}}$, 10^{-4}, 10^{-3}}; the batch sizes {128, 256, 512, **1024**}; the rollout fragment lengths {**32**, 128}; and Adam epsilons {**0.00015**, 10^{-8}}, where the best values are shown in bold, and the other parameter values are the same as in [90].

- A swarm of coordinated MADDPG searchers: The OpenAI MADDPG implementation [1] 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 $Q(\vec{o},\vec{a})$. This work tested the learning rates {10^{-4}, $\mathbf{10^{-3}}$, 10^{-2}}; the batch sizes {256, **512**, 1024}; and the model update rates {4, **100**, 500}, where the best values are shown in bold.

Baselines for SOP In the overall performance of the multi-target self-organizing pursuit (SOP), this research compares tree implicit coordination methods: FSC2 and three others trained by the following MARL algorithms.

- Actor-critic [85] (with parameter sharing): practical well-performed RL algorithm which is suitable for large-scale homogeneous agents. This work tested the learning rates {$\mathbf{10^{-4}}$, 5×10^{-4}} (best value in bold) and three reward functions. Besides, this work tested trained the value function {10, **80**} times every training epoch and got similar final performance. The other hyperparameters are the same as those for the actor-critic algorithm in the self-organizing search experiments.

- MAPPO [98]: state-of-the-art on-policy MARL algorithm, which has the potential for large-scale applications. This work tested two inputs to the centralized value function: {**concentration of all agents' observations**, agent-specific global state} (similar), the learning rates {$\mathbf{10^{-4}}$, 5×10^{-4}} (best value in bold), three reward functions, and train both policy and value functions 10 times per training epoch. The implementation is based on the official code [2], and other hyperparameters are the default values provided by [98] for the MPE environments, which are verified with small experiments.

[1] https://github.com/openai/maddpg
[2] https://github.com/marlbenchmark/on-policy

- IPPO [75] (with parameter sharing): independent proximal policy optimization (PPO) algorithm with the same local observation as input for both the policy and value functions like the actor-critic baseline. All the other hyperparameters are the same with MAPPO.

Common experimental setup The policy and value models in all MARL algorithms use the same architecture: two-layer ReLU multi-layer perceptions (MLP) with hidden layers of size 400 and 300. In the SOP task, layer normalization [1] is added to each hidden and output layer for the three baseline algorithms: actor-critic, PPO, and MAPPO.

- **Reward function:** For self-organizing search (SOS) tasks, all MARL algorithms use the same reward function in Table 5.1. For self-organizing pursuit (SOP) tasks, all MARL algorithms use the same reward function in Table 5.2.

Table 5.2: Reward function $R^i(s_t, \vec{a}_t)$ for self-organizing pursuit (SOP)

Action	Reward
Capture a target	10
Neighbor a target	0.1
Collide	-12
Move before termination	-0.05

Table 5.3: Performance comparison on multi-target self-organizing pursuit (SOP) with 16 agents and 4 targets in 40×40 grid worlds. FSC2-HC: replace FSC2 fuzzy clustering with hard clustering. FSC2-NM: remove agent's memory in fuzzy clustering. FSC2-RC: replace FSC2 fuzzy clustering with random clustering. FSC2-FS: replace FSC2 search with fish flocking rules (no migratory urge). * represents the statistical significance by student's t-test at the significance level 0.01.

Algorithm		FSC2	FSC2-HC	FSC2-NM	FSC2-RC	FSC2-FS	Actor-critic	IPPO	MAPPO
Clustering	Memory	✓	✓	✗	✗	✓	-	-	-
	Fuzzy membership	✓	✗	✓	✗	✓	-	-	-
Self-organizing search		✓	✓	✓	✓	✗	-	-	-
Capture rate		1 (0)	1 (0)	0.975* (0.075)	0.965* (0.086)	0.885* (0.155)	0.91* (0.139)	0.452* (0.228)	0.6475* (0.223)
Episode length		108.09 (50.256)	119.35 (75.99)	156.39* (133.278)	178.77* (147.369)	308.94* (181.485)	281.97* (176.28)	496.06* (38.505)	478.84* (63.689)
Collisions		0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	6.59* (6.935)	31.14* (25.285)	12.47* (20.991)

Table 5.4: Episode length (efficiency) comparison of FSC2 with fuzzy clustering and hard clustering on multi-target pursuit (SOP) in 40×40 grid worlds. FSC2-HC: replace FSC2 fuzzy clustering with hard clustering.

No. of agents	16	32	64	128	256	512	1024
FSC2	**108.09** (**50.256**)	**114.51** (**69.885**)	134.22 (102.551)	**107.15** (**78.498**)	**112.83** (**119.471**)	**285.51** (**216.59**)	**338.18** (230.582)
FSC2-HC	119.35 (75.99)	115.09 (83.229)	**115.32** (**90.802**)	126.77 (118.194)	133.85 (141.501)	289.65 (216.986)	353.1 (**224.407**)

5.4.2 Self-organizing pursuit (SOP) experiments

For the overall performance in multi-target self-organizing pursuit (SOP), the proposed FSC2 method is compared with three implicit coordination policies trained by the CTDE parameter sharing based actor-critic algorithm, PPO, and MAPPO, respectively. These methods solve the large-scale implicit multi-agent coordination problem constrained by partial observation and no inter-agent communications in three ways: hierarchical decomposition, parameter sharing based coordinated reinforcement learning, and centralized value function enhanced coordinated reinforcement learning.

The results are shown in Figure 5.5 and Table 5.3. It can been seen that FSC2 significantly outperforms the other methods over all metrics. Compared with PPO-based methods, the actor-critic algorithm achieves better results even with less model updates in the training. Compared with IPPO, MAPPO performs better most of the time but its centralized value function does not achieve better multi-agent collision avoidance when the swarm density is extremely higher than that in its training.

From the video rendering results, the swarm search strategy, especially the swarm migration ability (see Section 5.3.2), plays an vital role in the overall performance, the ineffectiveness of which contributes to the inferior performances of general MARL policies. Besides, another main challenge of general MARL algorithms is the multi-agent safety issue, such as the collisions. It is very hard to achieve the safety guarantee by a reward function, which is especially challenging with more agents and conflicts of interests being involved [99]. The conflicts are non-trivial to be resolved since agents make decisions and execute actions simultaneously in POMG. In contrast, FSC2 employs the CCR algorithm as the third module in its framework for the close coordination of agents, the safety of which is guaranteed by the fitness function in the online planning.

Scalability and swarm performance of FSC2 The swarm performance and scalability of up to 2048 FSC2 agents are tested in multi-target SOP in 40×40 and 80×80 grid worlds, as shown in Figures 5.5 and 5.6, respectively. Almost all experiments achieve a nearly 100% average capture rate except that when 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 5.6. However, the more than 68% average capture rate proves the efficient search ability of FSC2 agents in such trials.

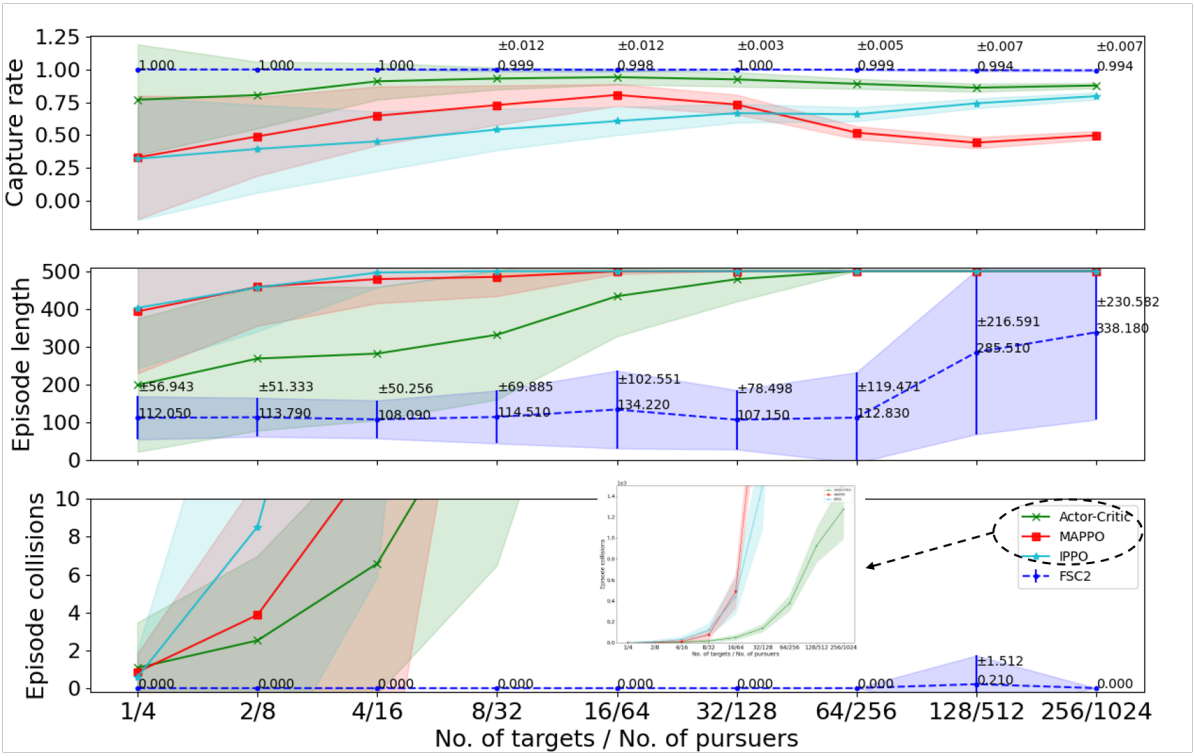


Figure 5.5: Swarm performance in the multi-target self-organizing pursuit (SOP) in 40×40 grid wolds with different numbers of targets and pursuers, where the mean and standard deviation of the experimental results in 100 independent runs are plotted.

Note that, the collisions in 0.22% and 2.1% of the trials in Figures 5.5 and 5.6 occur when the FSC2 agent is a searcher, i.e., the SOS agent in Algorithm 5.1. This does not mean a performance degradation of SOS agents in SOP tasks. Rather, it reveals the weak safety guarantee of RL algorithms. Figure 5.7 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

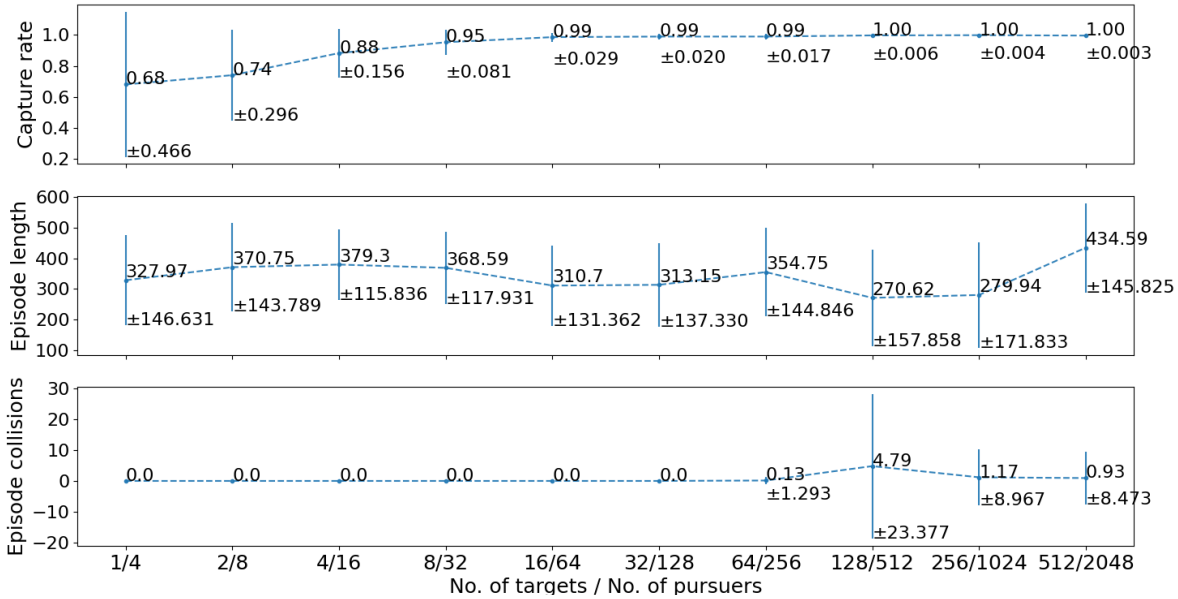


Figure 5.6: Swarm performance in the multi-target self-organizing pursuit (SOP) in 80×80 grid wolds with different numbers of targets and pursuers, where the mean and standard deviation of the experimental results in 100 independent runs are plotted.

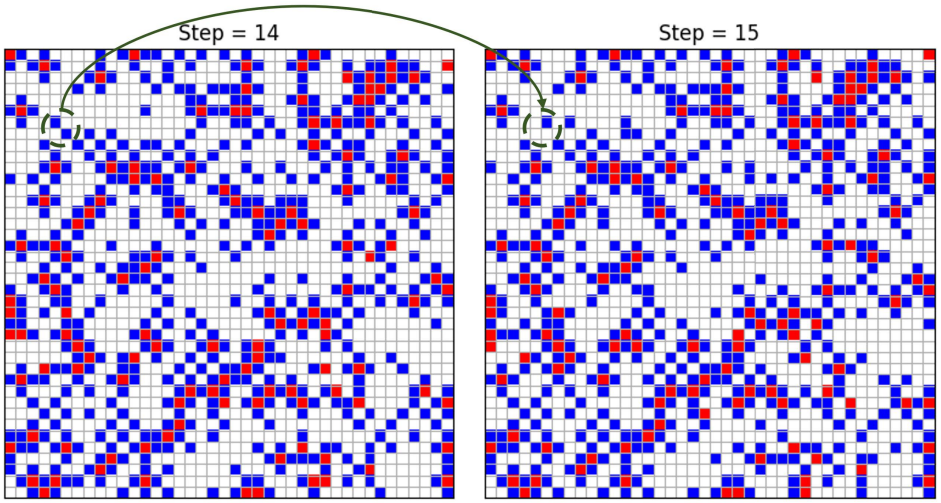


Figure 5.7: Multi-agent collision scenario illustration in the multi-target SOP with 128 targets (red squares) and 512 agents (blue squares) in 40×40 grid world: the two circled agents in step 14 are two searchers that collide with each other in step 15.

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 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 5.5 and 5.6.

In addition, the relatively stable swarm performance of FSC2 agents indicates that the three proposed subsolutions in FSC2, i.e., the MARL-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 performance are not restricted by the swarm size.

5.4.3 Self-organizing search (SOS) experiments

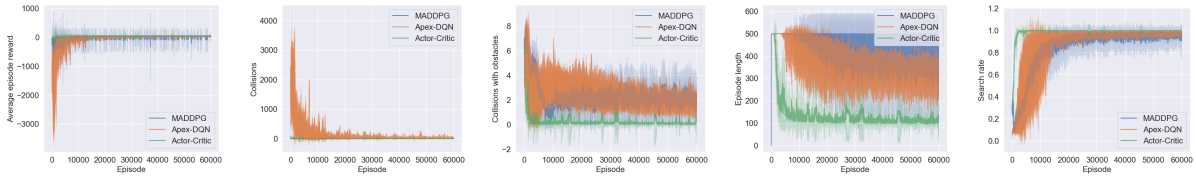


Figure 5.8: Training performance comparison on self-organizing search (SOS) over 10 random seeds, where the solid lines and shaded areas represent the mean and standard deviation of the corresponding performance, respectively.

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 5.8. The average episode 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 5.1 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 to ApeX-DQN, it is 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, the parameters are chosen to 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.

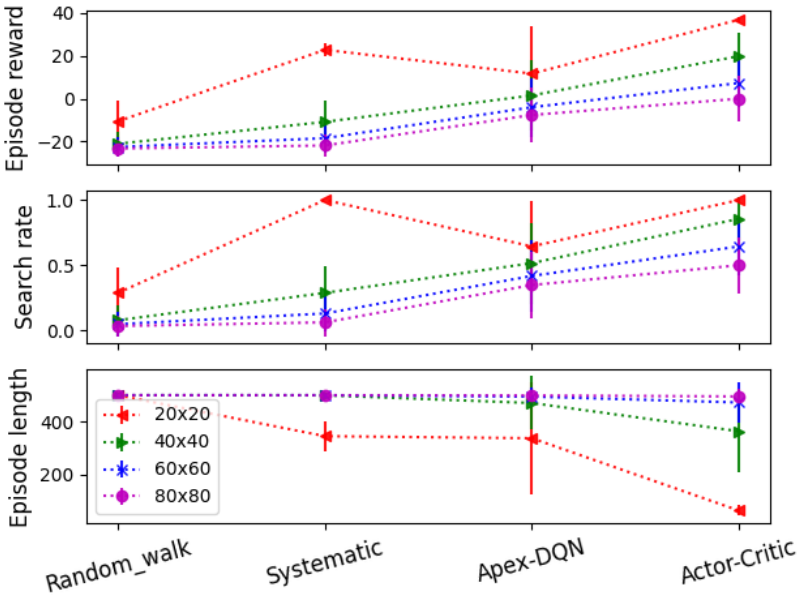


Figure 5.9: Single SOS agent performance comparison in grid worlds of different sizes, where the mean and standard deviation of the experimental results in 100 independent runs are plotted

Second, a single agent's searching performance is compared in the $20 \times 20, 40 \times 40, 60 \times 60$, and 80×80 grid worlds with 5 targets in Figure 5.9. With the increase of the environment size and 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.

Third, the swarm performance of different policies are compared 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 5.10. The smaller the environment is, the larger the swarm density is, and the more challenging the multi-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

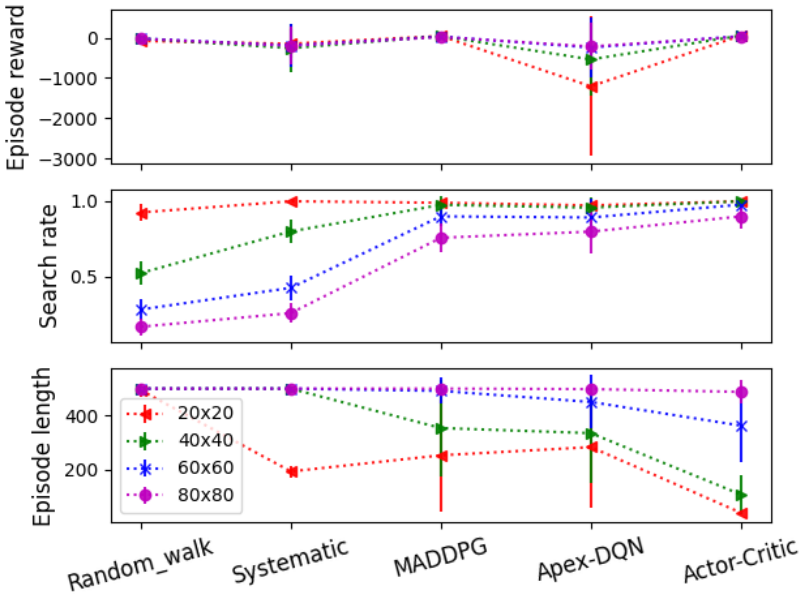


Figure 5.10: Swarm performance comparison of 8 SOS agents searching 50 targets in grid worlds of different sizes, where the mean and standard deviation of the experimental results in 100 independent runs are plotted

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 5.9 and 5.10 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.

Explainable search behavior analysis and sparse targets exploration One basic problem to be solved in self-organizing 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 [70] 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 [67]. However, as indicated in [70], these three behaviors can only support aimless flocking; it is also observed in our experiments that if only these three rules are applied, 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 [70], it is observed that the successfully trained self-organizing searchers learn similar behaviors by themselves. As shown in Figure 5.11, the actor-critic searcher's behavior is tested 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 5.11, 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 5.9.

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.

5.4.4 Consistency analysis 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 5.12a, 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

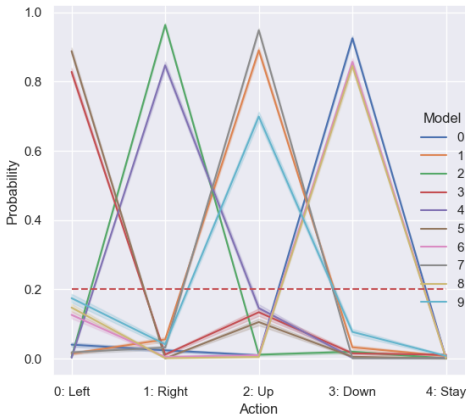
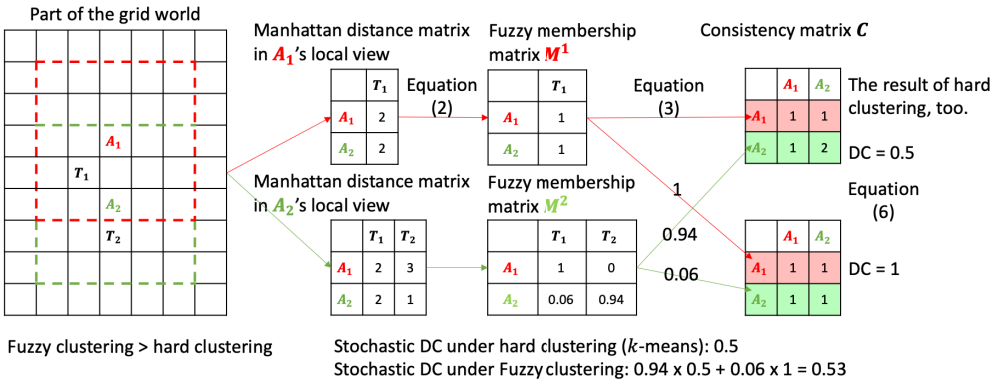


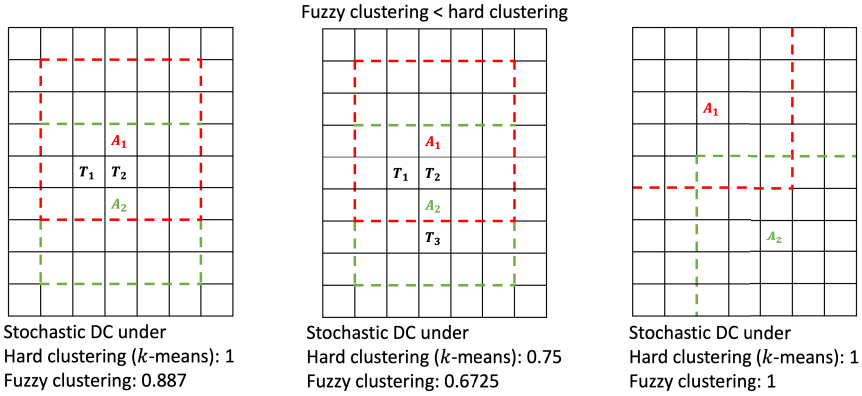
Figure 5.11: Behavior probability or action distribution of actor-critic trained self-organizing search (SOS) policy with the empty observation, which is estimated from 100 independent runs with 1000 steps per run. The different models are actor-critic policies trained with different random seeds.

nearest targets in hard clustering and fuzzy membership values in the fuzzy clustering may all stochastically result in different consistency matrices C. The DC value of each matrix C is multiplied with its corresponding probability and obtain the stochastic DC value. Figure 5.12a 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 5.12a. In contrast, as illustrated in Figure 5.12b, 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.



(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 5.12: The computational process of DC in Equation (5.6) 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.

5.4.5 Ablation studies

Influence of fuzziness: fuzzy clustering vs. hard clustering The fuzzy membership value calculation of Equation (5.1) to (5.4) are replaced in the fuzzy clustering with a hard clustering method. Since the cluster centers are known to be local free targets as introduced in Section 5.3.1, there is no need to calculate the k cluster centers as k-means. But, similar to the hard-clustering in k-means, an agent greedily selects the nearest cluster center and joins that cluster. The result is shown in the column of FSC2-HC of Table 5.3. In the 100 experiments, the efficiency degradation of FSC2-HC is observed in

terms of the episode length, but the statistical significance evidence is not obtained from student t-test. Following the same statistical comparison of Table 5.3, the efficiency of FSC2 and FSC2-HC are compared in 40×40 grid worlds with the 16, 32, 64, 128, 256, 512, and 1024 agents and get the same conclusion, as shown in Table 5.4. The conclusion here is in accordance with the clustering consistency analysis that the fuzziness of fuzzy clustering and its stochastic clustering enable the fuzzy clustering to be as good as or even better than hard clustering. In other words, the task time is extended due to the inconsistent distributed hard clustering. Besides, another reason that the stochastic significance is not achieved may be that the fuzzifier parameter α in Equation (5.1) is not optimized and our test experiments are not large enough to observe the difference.

Influence of memory The agent memory in the fuzzy clustering of Algorithm 5.2 is removed. The comparison result is shown in the FSC2-NM column of Table 5.3. It is seen that without the agent memory, the multi-target pursuit performance degrade significantly. As introduced in Section 5.3.1, without the memory, an agent is hard to cope with temporal uncertainty due to the partial observation and may switch the roles between searcher and pursuer. It causes the inconsistent or unstable successive decision-making of agents in the time scale and thus reduce the overall task performance.

Influence of clustering In this part, the first module of FSC2: fuzzy clustering is totally replaced with the random clustering to see to what extent an effective clustering method can influence the overall multi-target pursuit. The result is shown in the FSC2-RC of Table 5.3. It can be seen that, a random clustering performs significantly worse than FSC2, FSC2-HC, and FSC2-NM, which prove the necessity of effective clustering to the whole task completion.

Influence of “migration" ability in search In Section 5.3.2, three abilities for a successful self-organizing searcher are proposed: (1) the ability to effective search as a single agent; (2) the ability to form flocks and get benefits from the swarm; and (3) the ability to perform effective “migration" like actions in order to realize the swarm potential. In this part, the third ability is removed by replacing the second module of FSC2: RL trained search policy, with the three behavioral rules proposed by Reynolds [70] in simulating natural flocking and school of fish [67]. The three rules are: (1) avoid collisions with neighbors; (2) match velocity with neighbors, and (3) stay close to neighbors. As indicated in [70], these three behaviors can only support aimless flocking, i.e., no “migration" ability. The result is shown in the FSC2-FS column of Table 5.3, which significantly perform worse than others in terms of both capture rate and efficiency. It proves the importance of “migration" ability in the self-organizing search.

5.4.6 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 5.3.1, the time complexity in terms of Equations (5.2) to (5.5) 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 5.3.2, the time complexity of the policy model with input size $3r^2$ is $O(r^2)$. Third, for the single-target pursuit in Section 5.3.3, the time complexity [3] of Equation (5.12) 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 (5.17) is $O(n^2) + O(m^2) + O(n \cdot m) = O(max(n,m)^2)$ in the worst case. Therefore, based on Algorithm 5.1, 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 Chapter 2, 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 [101]. In FSC2, when pursuers surround the target, 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 [101], four pursuers take charge of each target and thus there are many distributed mass captures in the FSC2. Therefore, compared with the mean field reinforcement learning of Zhou et al. [101], 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 occupation-based and the number of pursuers needed for a target is not greater than 4, as in MPE [49]. FSC2 agents only need to walk towards the target one more step after they surround the target and the target cannot move. Actually, in addition to the occupation-based pursuit, pursuers can do many things as long as the target is surrounded, such as tagging the target as in MAgent [100]. In the proposed fuzzy-based distributed task allocation, the number of agents is not limited 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 occupation-based capture yet with more than 4 pursuers for each

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

target. In addition, the fitness function, i.e., Equation (5.17), 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 [83]. 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.

5.5 Summary

This chapter investigated the large-scale partial observable 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 dealt with two kinds of uncertainties in SOP: observation uncertainty and interaction uncertainty. By comparing with other implicit coordination policies, the superior performance of FSC2 and the benefits of the hierarchical framework by decomposing the task are proved. The scalability, interpretability, and rationality of FSC2 have been verified through experiments, empirical analyses, and ablation studies.

However, the safety of interagent collision avoidance is difficult to be guaranteed by MARL without explicit communications, which has also been verified by our experiments. This was one motivation that MARL is only applied in the search sub-task, not the target pursuit task 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, especially in terms of the multi-agent safety issue.

CHAPTER 6

ADVERSARIAL PURSUIT-EVASION WITH PARTIAL OBSERVATION

6.1 Background

This chapter investigates the adversarial pursuit-evasion game, where both pursuers and evaders co-evolve. First, this chapter presents a brief survey and clarifies the relationships between several similar concepts: co-evolution, self-play, autocurricula, adaptation, arms races, and adversarial learning, as shown in Figure 6.1. The aim is to better understand and highlight the role of co-evolution in multi-agent settings, more accurately use related terminologies, and discuss some common misleading expectations and magics, especially for researchers who are new to this field. Based on this, it is argued that MatrixWorld can serve as the first environment for autocurriculum research, where ideas can be quickly verified and well understood. Then, this research designs three adversarial learning algorithms based on different co-evolution mechanisms. Based on the experiments, various arms race outcomes are achieved. In particular, arms races with steady and converging improvement are more practical for increasingly complex behaviors, while policy cycles between two rival sides are useful for producing diverse policies. Besides, this research finds that the passive (evasive) policy learning benefits more from co-evolution than active (pursuing) policy learning in an asymmetric adversarial game.

6.2 Brief survey on co-evolution, autocurriculum, and arms race

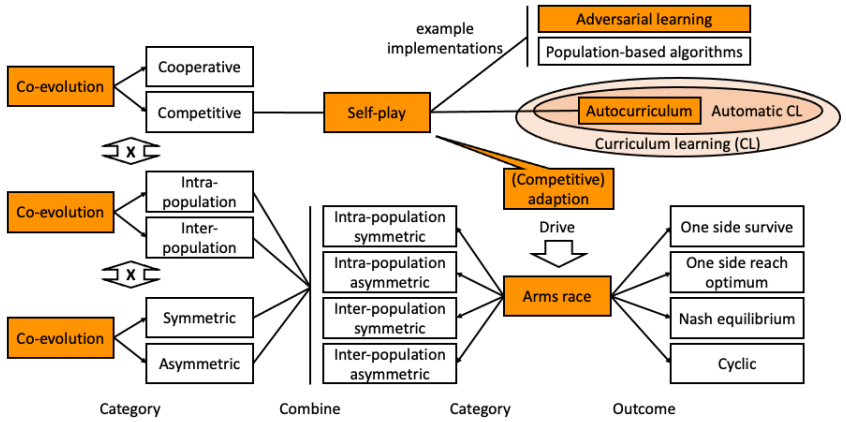


Figure 6.1: Relationships between co-evolution, self-play, autocurricula, arms races, and adversarial learning.

6.2.1 Co-evolution

Co-evolution is an natural phenomenon that witnesses the evolutionary progress and co-adaptations of, such as biological species. As an evolutionary and learning mechanism, its appealing properties, including autonomous co-adaptation and arms races, the natural modeling of coupled relationships, and performance improvement, contribute to its sustained success in diverse research fields and applications.

The most distinct feature of co-evolution is that the fitness evaluation of one individual depends on other individuals, where each individual is a separately evolving or learning agent. Co-evolution happens when adaptations occur between different agents, such as pursuers and evaders, environmental configurations and the agents therein. Its effectiveness is promoted by iterative coordinated learning among diverse individuals, which may induce a steadier and more guaranteed improvement. Therefore, co-evolution is more akin to a framework that can integrate different evolutionary computations (ECs), reinforcement learning (RL) algorithms, etc.

The available co-evolution methods can be broadly classified into three categories. First, there are competitive and cooperative co-evolution, which describe the relationships between individuals. However, they do not determine the number of populations that co-evolve. Individuals in one population can be competitive, while individuals from

multiple populations may be cooperative. The number of co-evolving populations defines the second category: intra-population and inter-population co-evolution methods. Third, co-evolution can be classified into symmetric and asymmetric contests. Symmetric co-evolution is for homogeneous agents in the sense that they learn similar skills for the same tasks. In contrast, asymmetric co-evolution is for heterogeneous agents that target different skills for different, perhaps competitive, tasks.

6.2.2 Self-play

Self-play is a kind of competitive co-evolution process. It generally refers to the competitive interactions of multiple agents but can also occur between different “minds" (policy models) of the same physical agent, such as Alice and Bob in the research of Sukhbaatar et al. [81]. Its idea is that learning progress is only achieved by the interplay or interaction between agents without external interference. For example, Jaderberg et al. [41] trained symmetric adversarial agents in a competitive game (capture the flag) by letting the agents in a population play with each other. Baker et al. [2] trained asymmetric adversarial agents in another competitive game: hide-and-seek through multi-agent competitions.

6.2.3 Adaptation and arms races

Arms races between two rival sides are common in nature, such as those between predators and prey, and between parasites and hosts. Biologically, Dawkins et al. [23] discussed an arm race as an evolutionary process of reciprocal counter-adaptations and the resultant challenges faced by two sides. Based on this definition, it seems that the most crucial feature for identifying an arms race in a co-evolution process is the driving force derived from mutual counter-adaptations. However, note that, adaptations and challenges do not necessarily mean more complex, stronger, or more general skills, which will be further explored in Section 6.2.4.

In addition, Dawkins et al. [23] proposed a two-way classification paradigms for arms races, i.e., symmetric or asymmetric, and interspecific or intraspcific, which form four possible combinations with biological examples. This is also consistent with the categories of co-evolution. In addition, they summarized four classes of arms race endpoints: one side wins by driving the other to extinction, one side wins by first reaching its optimum, equilibrium endpoints, and cyclic endings. One of the interesting discussions provided by Dawkins et al. [23] indicates that the average success of one side, such as the capture rate

of predators, does not necessarily increase with the evolutionary time in an arms race. This is because both rival sides are improving. Then, the question of how to evaluate an arms race is progressing is actually the same as asking how the arms race will terminate, as described by the four possible outcomes of arms races.

6.2.4 Curriculum learning (CL), automatic CL, and autocurricula

Curriculum learning (CL) [10] refers to a training strategy that learns from a sequence of related tasks organized with increasing difficulty, which generally achieves better results faster than normal learning methods. Rather than manually designing a set of tasks and the time required to switch between them, automatic CL automates part or all of the training process [93]. In particular, the autocurriculum concept proposed by Leibo et al. [46] specifically refers to automatic CL in multi-agent settings, where the sequence of tasks or challenges is self-generated from mutual counter-adaptations in multi-agent interactions, i.e., an arms race. In this sense, an autocurriculum can be seen as equivalent to self-play.

However, based on both biological observations [23] and artificial intelligence (AI) investigations [46], it is noted that challenges do not definitely mean more difficult or complex tasks, and adaptations to these challenges do not necessarily involve increasing strong abilities. Even the commonly adopted difficulty ranking of tasks, i.e., from easy to hard, in curriculum learning is not guaranteed to be optimal in all cases [93]. The conditions for obtaining different arms race outcomes have attracted researchers' attention. Particularly, people are more interested in generating stronger models, such as agents with more complex behaviors. Compared with the various optimal rankings of task difficulties, seemingly, the conclusions on how to avoid policy cycles are more consistent. It has been shown that the policy cycling issue may occur in both symmetric and asymmetric arms races, for example, in a symmetric intra-population adversarial game of rock-paper-scissors [46] where the same agent strategy can play two different roles, and an asymmetric game [58]. Nolfi et al. [58] investigated the policy cycling problem in the co-evolution and proposed that it could be reduced by evaluating the current policy with all past policies, which successfully drives the evolutionary process to an arms race with increasing complexity. The same conclusion was also identified by Leibo et al. [46] and Vinyals et al. [92]: self-play algorithms that do forgetting past policies provide a way to break out the policy cycling problem. In addition, Leibo et al.

[46] pointed out that the ceiling of the intelligence that can be achieved by an arms race is determined by the “environment’s carrying capacity", or the task definition that the autocuriculum is trying to solve. This research actually states that optima are defined by the problem itself. For example, in a pursuit-evasion game without tools, novel tool usages will never emerge.

6.2.5 Adversarial learning

Adversarial learning (AL) is a training framework that optimizes adversarial models with conflicting objectives and is typically implemented in an alternative learning mode. Different from minimax search algorithms [72] in which an agent optimizes itself against an optimal adversary, AL can be more general, as will be shown in our experiments (Section 6.4). Subsequently, AL is not the only way to co-evolve adversarial agents and is more of an example implementation of self-play. For example, in the symmetric adversarial game “capture the flag" [41], population-based RL was used to train a population of agents with similar skills that could play two adversarial game roles.

In addition, one main application of AL is to target an ultimate protagonist model rather than an adversary model, such as the generator model in generative adversarial networks (GANs) [32] and primary agents in the robust adversarial reinforcement learning (RARL) [50, 62, 66]. Last, note that not all adversarial work refers to co-evolutionary adversarial learning, and they may be merely normal learning approaches in adversarial settings. In these works, only one side in the adversarial setting is intentionally trained to be an opponent of the other, such as the adversarial policies in RL [31] and adversarial examples.

6.2.6 MatrixWorld: A lightweight co-evolution environment

Biologically, predators pursue their prey for dinner, while prey evade predators for life, and their co-evolutionary arms race and co-adaptations never stop. In AI, the co-evolutionary pursuit-evasion has been studied since 1994 [51, 53, 54, 71], which is slightly later than the research on pursuit or evasion alone, which has taken place since 1953 [48]. Compared with complex 3D first-person video games such as Quake III Arena [41], real-time strategy games such as StarCraft II [73], or adversarial games that need millions of episodes for arms races [2], MatrixWorld is a lightweight and simple co-evolution framework based on the pursuit-evasion game.

As discussed above, research on co-evolution is still open and it is supposed to be promising for automatically generating more complex agent behaviors and intelligence from the arms races in multi-agent interactions. This research suggests that MatrixWorld can serve as the first environment for quickly verifying and effectively understanding research ideas in autocurricula.

6.3 Adversarial learning algorithms for pursuit-evasion

This research particularly explores three representative co-evolution frameworks and their influences on the arms race process and outcomes.

- Pursuer specialist vs. evader specialist (Algorithm 6.1): Pursuers are trained to be specialized for the current evaders, and so are the evaders.
- Pursuer generalist vs. evader specialist (Algorithm 6.2): Pursuers are trained to be general, while the evaders are trained to be specialized to their opponents, which is an unfair training system but may be more practical in the real world. For example, police can learn from all past criminal data, while criminals may only access the current police data.
- Pursuer generalist vs. evader generalist (Algorithm 6.3): Pursuers are trained to be general and robust for all past evaders, and so are the evaders.

The alternative learning scheme of adversarial learning is applied that when one side learns the other side is fixed.

6.4 Experiments

In this section, through adversarial learning, various arms race outcomes of different co-evolution mechanisms are achieved in MatrixWorld. Based on experiments, arms races with steady and converging improvement are more practical for increasingly complex behaviors, while policy cycles between two rival sides are useful for producing diverse policies. In particular, this research finds that the passive (evasive) policy learning benefits more from co-evolution than active (pursuing) policy learning in an asymmetric adversarial game. An arms race can drive the passive policy to a higher level than that

Algorithm 6.1: Adversarial learning: Pursuer specialist vs. evader specialist

Initialize pursuer model θ_0^p and evader model θ_0^e.
for *generation* $k = 1 : N$ **do**
  $\theta^p_{k,k^p=0} = \theta^p_{k-1}$.
  for $k^p = 1 : N^p$ **do**
    Collect experiences by θ^p_{k,k^p-1} and θ^e_{k-1}.
    Pursuer model update $\theta^p_{k,k^p-1} \rightarrow \theta^p_{k,k^p}$.
  $\theta^p_k = \theta^p_{k,N^p}$.
  $\theta^e_{k,k^e=0} = \theta^e_{k-1}$.
  for $k^e = 1 : N^e$ **do**
    Collect experiences by θ^p_k and θ^e_{k,k^e-1}.
    Evader model update $\theta^e_{k,k^e-1} \rightarrow \theta^e_{k,k^e}$.
  $\theta^e_k = \theta^e_{k,N^e}$.
  return θ^p_N, θ^e_N.

Algorithm 6.2: Adversarial learning: Pursuer generalist vs. evader specialist

Initialize pursuer model θ_0^p and evader model θ_0^e.
for *generation* $k = 1 : N$ **do**
  $\theta^p_{k,k^p=0} = \theta^p_{k-1}$.
  for $k^p = 1 : N^p$ **do**
    $k^e = k^p \bmod k$.
    Collect experiences by θ^p_{k,k^p-1} and $\theta^e_{k^e}$.
    Pursuer model update $\theta^p_{k,k^p-1} \rightarrow \theta^p_{k,k^p}$.
  $\theta^p_k = \theta^p_{k,N^p}$.
  $\theta^e_{k,k^e=0} = \theta^e_{k-1}$.
  for $k^e = 1 : N^e$ **do**
    Collect experiences by θ^p_k and θ^e_{k,k^e-1}.
    Evader model update $\theta^e_{k,k^e-1} \rightarrow \theta^e_{k,k^e}$.
  $\theta^e_k = \theta^e_{k,N^e}$.
  return θ^p_N, θ^e_N.

in normal RL. Finally, based on experiments in Pursuit-Evasion-S, this research shows the demanding safety assurance regarding the reward shaping or other techniques in safe MARL.

Algorithm 6.3: Adversarial learning: Pursuer generalist vs. evader generalist

Initialize pursuer model θ_0^p and evader model θ_0^e.
for *generation* $k = 1 : N$ **do**
  $\theta^p_{k,k^p=0} = \theta^p_{k-1}$.
  for $k^p = 1 : N^p$ **do**
    $k^e = k^p \bmod k$.
    Collect experiences by θ^p_{k,k^p-1} and $\theta^e_{k^e}$.
    Pursuer model update $\theta^p_{k,k^p-1} \rightarrow \theta^p_{k,k^p}$.
  $\theta^p_k = \theta^p_{k,N^p}$.
  $\theta^e_{k,k^e=0} = \theta^e_{k-1}$.
  for $k^e = 1 : N^e$ **do**
    $k^p = k^e \bmod (k+1)$.
    Collect experiences by $\theta^p_{k^p}$ and θ^e_{k,k^e-1}.
    Evader model update $\theta^e_{k,k^e-1} \rightarrow \theta^e_{k,k^e}$.
  $\theta^e_k = \theta^e_{k,N^e}$.
  return θ^p_N, θ^e_N.

6.4.1 Experimental setup

The policy models for the pursuers and evaders are both two-layer ReLU multi-layer perceptrons (MLPs) with hidden sizes of [400, 300]. The actor-critic [85] algorithm is adopted in the centralized training and decentralized learning (CTDE) scheme, with the policy and value learning rates set to 3×10^{-4} and 10^{-3}, respectively. To stabilize the learning process, the actor model is trained 5 times, while value function is trained 1 time per epoch. $N = 30$ generations of co-evolution are conducted for Algorithms 6.1, 6.2, and 6.3, with $N_p = N_e = 400$ epochs in each generation. For the Pursuit-Evasion-O task, 8 pursuers compete with 30 evaders in 40×40 grid worlds, while for Pursuit-Evasion-S, 20 pursuers compete with 5 evaders since 4 pursuers are required for every evader in the surrounding-based capture paradigm. The reward functions are shown in Table 6.1.

Suggestion: Set the hyperparameters N_p and N_e to allow the (moving average) learning performance improve to some extent. In this way, an arms race can be observed in the adversarial learning; otherwise, neither rival side can learn effectively.

6.4.2 Autocurricula in co-evolutionary pursuit-evasion

Adaptation and arms race occur between pursuers and evaders. As shown in the training performance presented in Figure 6.2, the learning of pursuers brings challenges

Table 6.1: Reward function for all pursuit-evasion tasks. "–": the same.

Pursuer		Evader	
Action	Reward	Action	Reward
Capture an evader	10	Being captured	-10
Neighbor an evader	0.1	Being neighbored	-0.1
Collide	-12	–	-12
Move before termination	-0.05	–	-0.05

to the evaders' policis, the capture rate increases, and vice versa. The performance of the agents (pursuers and evaders) continues to improve with iterative generations. Their performance variance brought by policy adaptations tends to converge as the number of generations increases, but this is not the end of the arms race. When many more generations than $N = 30$ are observed, say 100 generations, this research finds that the performance variance diverges again, which demonstrates the sustained adaptations during the arms race.

Policy cycles occur in the specialist-vs.-specialist framework, but they are avoided in the generalist-vs.-generalist scheme. As shown in Figure 6.3, the evolutionary performance (capture rate) of the evaders are tested against the pursuers in each generation. The policy cycles are observed in Figure 6.3a, where both the pursuers and evaders learn only against their contemporary opponents. The evaders' performances against the pursuers in a specific generation, i.e., one row in the figure, fluctuate periodically from generation to generation, or even decrease sometimes. This indicates that due to the mutual adaptations of pursuers and evaders, their skills periodically appear and disappear, which is called a policy cycle [58]. On the other hand, such policy cycles from counter-adaptations are useful for producing diverse policies with similar complexity levels. In contrast, when the pursuers are trained to be general in Algorithm 6.2, the policy cycle problem is less severe in Figure 6.3b. Furthermore, when both the pursuers and evaders learn against all past opponents, the skill of competing with a specific pursuer is preserved in later generations, which can be seen from the stable performance in Figure 6.3c after the evaders first learn against that pursuer. This indicates that learning against past opponents helps avoid forgetting already learned skills. This result is consistent with the idea of past literature works on eliminating policy cycles, such as [46, 58, 92].

An arms race is more useful for learning passive policies. In Figure 6.4, the generalization performance of the agent in each generation is tested by averaging its

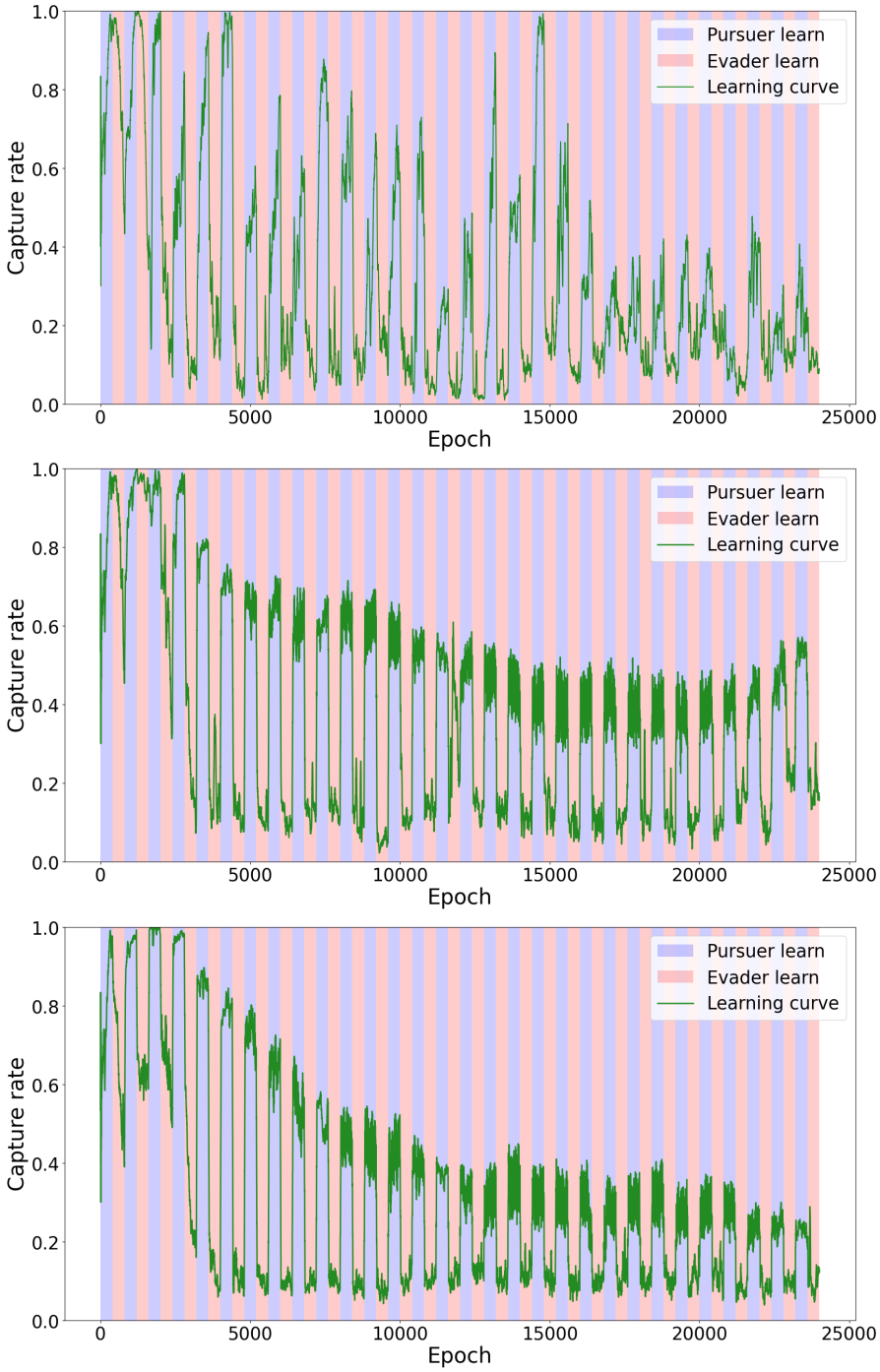
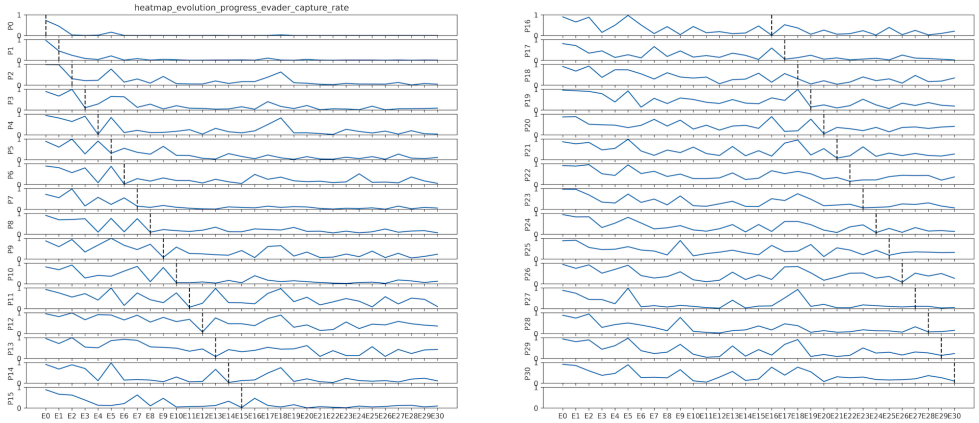
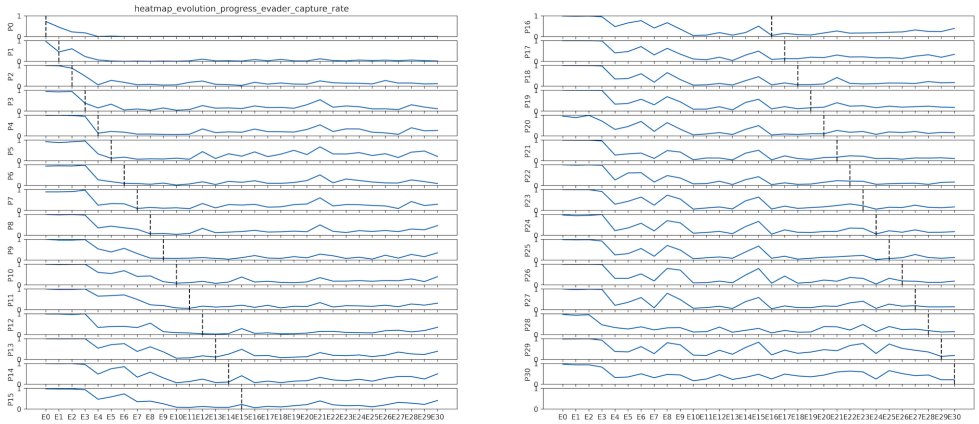


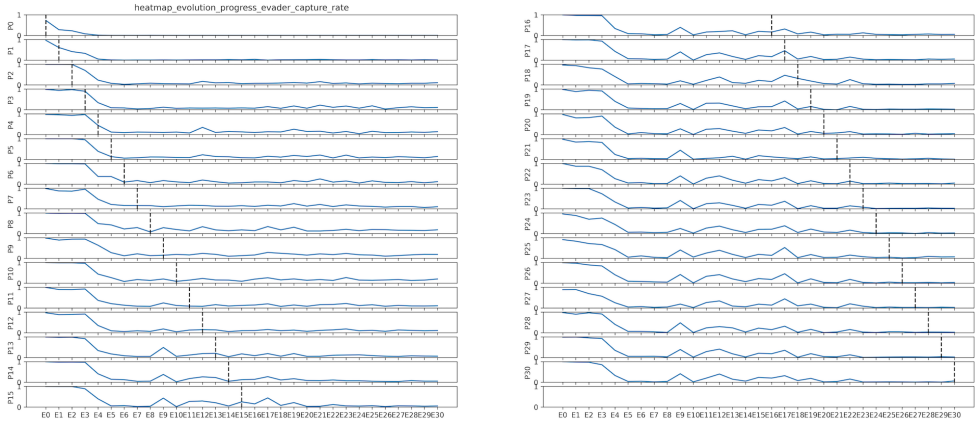
Figure 6.2: Training performance achieved for Pursuit-Evasion-O by Algorithms 6.1, 6.2, and 6.3 (from top to bottom). The curves are smoothed over 30 points.



(a) Pursuer specialist vs. evader specialist (Algorithm 6.1)



(b) Pursuer generalist vs. evader specialist (Algorithm 6.2)



(c) Pursuer generalist vs. evader generalist (Algorithm 6.3)

Figure 6.3: Evolutionary performance (capture rate) of evaders based on the testing performance achieved on Pursuit-Evasion-O, which is averaged over 10 independent runs. Horizontal axis: E0 - E30, the 30 generations of evaders. Vertical axis: P0 - P15 (left), P16 - P30 (right), the 30 generations of pursuers. E0 and P0 are the initial evaders and pursuers. A black dashed line indicates the associated pursuer and evader are in the same generation.

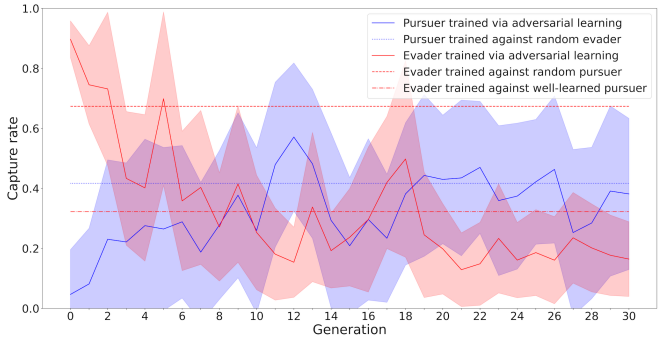
performance against the opponents across all generations. At the same time, this research compares it with three baselines.

- Baseline 1: the pursuer that is trained by competing with randomly walking evaders.
- Baseline 2: the evader that is trained by competing with randomly walking pursuers.
- Baseline 3: the evader that is trained by competing with well-learned pursuers. The well-learned pursuer is trained by competing with randomly walking evaders. That is, the baseline 3 is trained via manual curriculum learning.

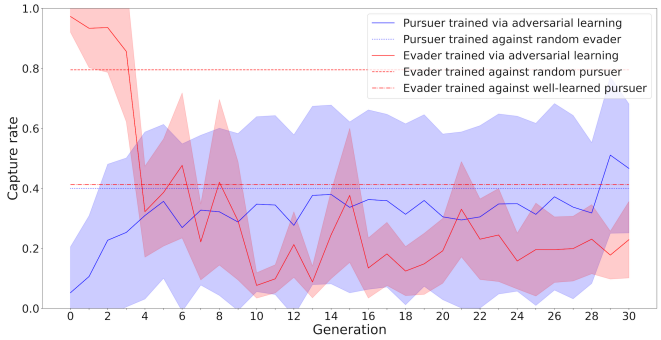
The conclusion that an arms race is more useful for learning passive (evasive) policies can be drawn based on two consistent observations from Figure 6.4a to 6.4c. First, adversarial learned evaders have better generalization performance than the baseline evaders, while adversarial learned pursuers do not achieve significantly better generalization performance than the baseline pursuers. Second, it can be seen that the complexity of a passive evasive policy highly depends on opponents' ability. When trained only against random pursuers, the evaders' performance is hard to improve once it reaches a certain level, which is the worst in Figure 6.4. If trained against stronger pursuers, i.e., well-learned pursuer, their performance is significantly enhanced. The best evader policies are obtained by adversarial learning. This indicates that autocurriculum learning is achieved and increasingly strong pursuers drive the continuous learning of evaders.

6.4.3 General MARL in safe multi-agent coordination scenarios

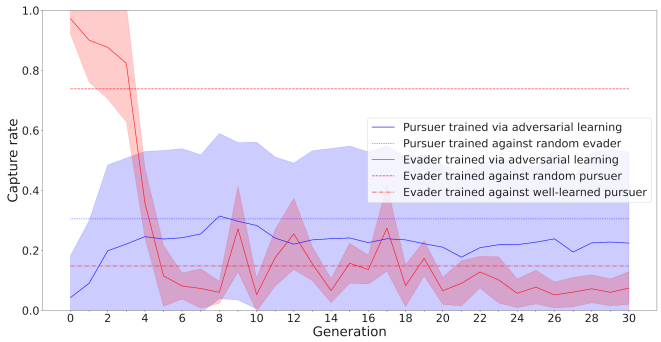
Compared with Pursuit-Evasion-O, more multi-agent conflicts of interest may occur in Pursuit-Evasion-S since more explicit multi-agent coordination is required to capture each evader with four pursuers. Therefore, this research demonstrates the difficulties faced by general MARL in guaranteeing safe multi-agent coordination with only negative rewards for collisions in Pursuit-Evasion-S. As shown in Figure 6.5, after training for 1000+ epochs, the reward and capture rates improve significantly. However, more conflicts of interest occur with increasingly efficient capture behavior, as shown in the collision statuses. The multi-agent collisions do not absolutely vanish with a longer training time, especially in test scenarios, although the costs of collisions are larger than the benefits of capturing an evader in the reward structure (Table 6.1).



(a) Pursuer specialist vs. evader specialist (Algorithm 6.1)



(b) Pursuer generalist vs. evader specialist (Algorithm 6.2)



(c) Pursuer generalist vs. evader generalist (Algorithm 6.3)

Figure 6.4: Generalization performance achieved for Pursuit-Evasion-O.

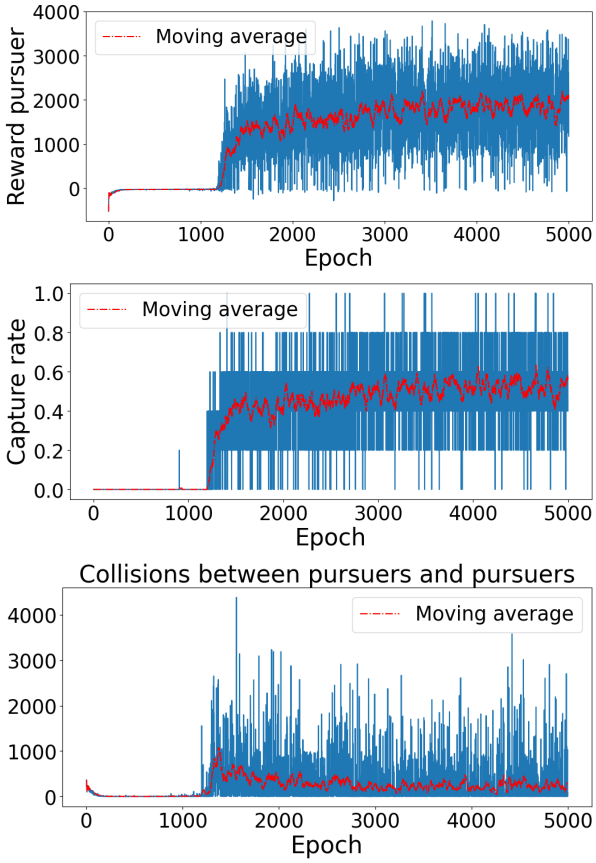


Figure 6.5: Training performance achieved for Pursuit-Evasion-S.

6.5 Summary

This chapter has revisited the concepts of co-evolution, autocurricula, self-play, arms races, and adversarial learning, and has clarified their relationships for better understanding and more accurate terminology usage. Through experiments, this research shows that ideas of autocurricula can be quickly verified and well understood in MatrixWorld. Finally, this research demonstrate the difficulties of general MARL algorithms in the safe multiagent coordination with only negative reward for collisions, which is one motivation of our work and challenge to the safe MARL community.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

This thesis has investigated the multi-agent coordination algorithms for pursuit-evasion games. The research objective is to achieve emerging swarm intelligence in the distributed implicit coordination of large-scale multi-agent systems (MAS), with only observations and no communications, as some of self-organizing systems in nature. To this end, five research questions have been answered in the thesis: how to resolve collisions in the multi-agent environment; how to be cooperative in a distributed way; how to allocate the multi-agent task; how to overcome the partial observation; how to get the arms race from the competitive co-evolution.

First, a safety-constrained multi-agent pursuit-evasion platform: MatrixWorld, has been proposed in Chapter 2. It addresses the limited safety support of popular multi-agent environments and contributes to the safe multi-agent coordination and autocurriculum learning in four ways. (1) The proposed safety-constrained multi-agent action-execution model is general for the software implementation of safe multi-agent environments. Its rationality, feasibility, and diversity in terms of the safety definition will guide algorithms learn correctly from right feedback. (2) Nine pursuit-evasion variants has been defined for example scenarios like real-world vehicle swarm, multi-agent path finding (MAPF), popular pursuit-evasion setups, and classic cops-and-robbers problem. (3) Safety related environmental information has been provided in the API for the open safe MARL research, such as performance evaluation, reward shaping, and safety con-

straints construction. (4) MatrixWorld can serve as a lightweight co-evolution framework for autocurriculum research, where ideas can be quickly verified and well understood.

Second, for the distributed coordination problem, this research finds that the cooperative co-evolution performance evaluation mechanism is useful to reach the swarm benefits from the distributed agent's perspective, and scalable with the swarm size. Besides, among the swarm intelligence (SI) algorithms, the position concept and update rule in particle swarm optimization algorithm (PSO) naturally fit the multi-agent scenario. Based on these findings, this research has proposed the cooperative co-evolutionary particle swarm optimization algorithm for robots (CCPSO-R) for the full observable pursuit of a single evader (Chapter 3). Further, the concept of virtual agents has been introduced along with the real agents to allow the concurrent search of swarm and individual benefits. A modular fitness functions has been heuristically designed, which separates the collision avoidance mechanism from SI itself. Experiments have shown the better reliability, generality, scalability, and effectiveness of CCPSO-R, compared with the repsentative dynamic path planning method.

Third, for the multi-agent task allocation problem, this research finds that the main limitation of centralized task allocation methods is that its optimization time is hard to match the real-time requirement in dynamic MAS, even when the search space is reduced significantly by utilizing the domain knowledge. For example, in the full observable pursuit of multiple evaders (Chapter 4), the proposed BiPCCR algorithm is struggle to deal with the swarm size larger than 16 agents. Besides, this research also finds that the fuzziness and soft clustering in fuzzy logic provide a feasible way to cope with the consensus issue in the distributed implicit task allocation. For example, in the partial observable pursuit of multiple evaders (Chapter 5), the necessity and effectiveness of the FSC2 component algorithm: fuzzy clustering have been proved by the superior overall performance of FSC2, empirical analyses, and ablation studies.

Fourth, for the partial observation problem, this research finds that it challenges all sub-tasks in the partial observable pursuit (Chapter 5). Accordingly, this research has proposed the FSC2 (fuzzy self-organizing cooperative co-evolution) algorithm. To overcome the frequent switch of agent roles in the distributed task allocation, FSC2 introduces the incremental agent memory. To achieve global coordination in the sparse targets exploration, FSC2 utilizes the reinforcement learning to learn the unknown local search policy. To address the safety risk arsing from the observation uncertainty in decisions, FSC2 designs the lexicographic convention and defines the concept of certain partial observation in the close single evader capture. Empirical analyses and ablation

studies have verified the interpretability, rationality, and effectiveness of the above designs. Experiments have demonstrated the superior performance of FSC2 compared with policies fully trained by general MARL algorithms. In this case, the pursuit by 2048 agents are demonstrated.

Fifth, for the competitive co-evolution of adversarial agents, this research finds that the co-evolution is a bigger framework, self-play is a kind of competitive co-evolution, autocurricula can be seen as equivalent to self-play, an arms race is an evolutionary process driven by mutual counter-adaptations and not necessarily means increasingly complex multi-agent behaviors, and adversarial learning is an alternative learning mode in the current practice, which is more of an example implementation of self-play. Through adversarial learning experiments, various arms race outcomes have been achieved by different co-evolution frameworks. In particular, this research finds that the arms race in an asymmetric adversarial game can drive the passive (evadive) policy to increasing complexities, the learning of which highly depends on its opponents' ability.

7.2 Limitations and future work

There are limitations in this work. For example, to keep safe coordination, this research integrates the collision avoidance constraint in the cooperative co-evolution fitness function and designs the lexicographic convention for close multi-agent coordination. Although these methods work well, they are inefficient sometimes and even trapped in rare deadlocks, which need to be optimized by, such as learning algorithms, in the future. Another limitation of this research is that the large-scale multi-agent coordination is achieved because the swarm agents are homogeneous and the proposed algorithms are not optimized based on the swarm size. However, the homogeneity assumption is not always true in real-world applications. In the future, more efforts can be devoted to achieve the scalable coordination for heterogeneous multi-agent swarm.

Finally, this research recommends the proposed safety-constrained multi-agent action execution model to be used in other multi-agent environments for safe coordination research. The pursuit-evasion based MatrixWorld can be used as the first lightweight co-evolution environment to quickly verify and well understand autocurriculum ideas, which can be then applied in more complex but heavy multi-agent problems. Besides, MatrixWorld can be also utilized to investigate many other grid world based tasks, including the multi-agent path finding (MAPF), etc. Last, the proposed FSC2 algorithms can serve as a strong baseline for the large-scale distributed pursuit problem.

BIBLIOGRAPHY

[1] J. L. BA, J. R. KIROS, AND G. E. HINTON, *Layer normalization*, arXiv preprint arXiv:1607.06450, (2016).

[2] B. BAKER, I. KANITSCHEIDER, T. MARKOV, Y. WU, G. POWELL, B. MCGREW, AND I. MORDATCH, *Emergent tool use from multi-agent autocurricula*, in International Conference on Learning Representations, 2019.

[3] N. BALAJI, S. KIEFER, P. NOVOTNÝ, G. A. PÉREZ, AND M. SHIRMOHAMMADI, *On the complexity of value iteration*, arXiv preprint arXiv:1807.04920, (2018).

[4] S. BARRETT, A. ROSENFELD, S. KRAUS, AND P. STONE, *Making friends on the fly: Cooperating with new teammates*, Artificial Intelligence, 242 (2017), pp. 132 – 171.

[5] S. BARRETT AND P. STONE, *An analysis framework for ad hoc teamwork tasks*, in Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, AAMAS 12, Richland, SC, 2012, International Foundation for Autonomous Agents and Multiagent Systems, pp. 357–364.

[6] ——, *Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork*, in Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI 15, AAAI Press, 2015, pp. 2010–2016.

[7] S. BARRETT, P. STONE, AND S. KRAUS, *Empirical evaluation of ad hoc teamwork in the pursuit domain*, in The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, AAMAS '11, Richland, SC, 2011, International Foundation for Autonomous Agents and Multiagent Systems, pp. 567–574.

[8] S. BARRETT, P. STONE, S. KRAUS, AND A. ROSENFELD, *Teamwork with limited knowledge of teammates*, in Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, AAAI 13, AAAI Press, 2013, pp. 102–108.

[9] M. BENDA, V. JAGANNATHAN, AND 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.

[10] Y. BENGIO, J. LOURADOUR, R. COLLOBERT, AND J. WESTON, *Curriculum learning*, in Proceedings of the 26th annual international conference on machine learning, 2009, pp. 41–48.

[11] D. S. BERNSTEIN, R. GIVAN, N. IMMERMAN, AND S. ZILBERSTEIN, *The complexity of decentralized control of markov decision processes*, Mathematics of operations research, 27 (2002), pp. 819–840.

[12] J. C. BEZDEK, *Pattern recognition with fuzzy objective function algorithms*, Springer, Boston, MA, 2013.

[13] A. BONATO, *The game of cops and robbers on graphs*, American Mathematical Soc., 2011.

[14] C. BOUTILIER, *Planning, learning and coordination in multiagent decision processes*, in TARK, vol. 96, Citeseer, 1996, pp. 195–210.

[15] ——, *Sequential optimality and coordination in multiagent systems*, in IJCAI, vol. 99, 1999, pp. 478–485.

[16] R. BURKARD, M. DELL'AMICO, AND S. MARTELLO, *Assignment Problems*, Society for Industrial and Applied Mathematics, 2012.

[17] R. E. BURKARD AND E. ÇELA, *Heuristics for biquadratic assignment problems and their computational comparison*, European Journal of Operational Research, 83 (1995), pp. 283–300.

[18] R. E. BURKARD, E. CELA, AND B. KLINZ, *On the biquadratic assignment problem*, in Quadratic Assignment and Related Problems: DIMACS Workshop, May 20-21, 1993, vol. 16, American Mathematical Soc., 1994, pp. 117–146.

[19] S. CAMAZINE, J.-L. DENEUBOURG, N. R. FRANKS, J. SNEYD, G. THERAULA, AND E. BONABEAU, *Self-organization in biological systems*, Princeton university press, 2001.

[20] E. CELA, *The quadratic assignment problem: theory and algorithms*, vol. 1, Springer Science & Business Media, 2013.

[21] T. H. CHUNG, G. A. HOLLINGER, AND V. ISLER, *Search and pursuit-evasion in mobile robotics*, Autonomous robots, 31 (2011), p. 299.

[22] M. S. COUCEIRO, R. P. ROCHA, AND N. M. F. FERREIRA, *A novel multi-robot exploration approach based on particle swarm optimization algorithms*, in 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, Nov 2011, pp. 327–332.

[23] R. DAWKINS AND J. R. KREBS, *Arms races between and within species*, Proceedings of the Royal Society of London. Series B. Biological Sciences, 205 (1979), pp. 489–511.

[24] C. DE SOUZA, R. NEWBURY, A. COSGUN, P. CASTILLO, B. VIDOLOV, AND D. KULIFÁ, *Decentralized multi-agent pursuit using deep reinforcement learning*, IEEE Robotics and Automation Letters, 6 (2021), pp. 4552–4559.

[25] S. L. DEVADOSS AND J. O'ROURKE, *Discrete and computational geometry*, Princeton University Press, 2011.

[26] M. EGOROV, *Multi-agent deep reinforcement learning*, CS231n: Convolutional Neural Networks for Visual Recognition, (2016).

[27] A. EIBEN AND J. SMITH, *Introduction to evolutionary computing*, Springer, 2015.

[28] J. FOERSTER, G. FARQUHAR, T. AFOURAS, N. NARDELLI, AND S. WHITESON, *Counterfactual multi-agent policy gradients*, in Proceedings of the AAAI conference on artificial intelligence, vol. 32, 2018.

[29] F. V. FOMIN, P. A. GOLOVACH, AND J. KRATOCHVÍL, *On tractability of cops and robbers game*, in Fifth Ifip International Conference On Theoretical Computer Science – Tcs 2008, G. Ausiello, J. Karhumäki, G. Mauri, and L. Ong, eds., Boston, MA, 2008, Springer US, pp. 171–185.

[30] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, USA, 1979.

[31] A. GLEAVE, M. DENNIS, C. WILD, N. KANT, S. LEVINE, AND S. RUSSELL, *Adversarial policies: Attacking deep reinforcement learning*, arXiv preprint arXiv:1905.10615, (2019).

[32] I. GOODFELLOW, J. POUGET-ABADIE, M. MIRZA, B. XU, D. WARDE-FARLEY, S. OZAIR, A. COURVILLE, AND Y. BENGIO, *Generative adversarial nets*, in Advances in Neural Information Processing Systems, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds., vol. 27, Curran Associates, Inc., 2014.

[33] Y. GUAN, D. MAITY, C. M. KRONINGER, AND P. TSIOTRAS, *Bounded-rational pursuit-evasion games*, in 2021 American Control Conference (ACC), 2021, pp. 3216–3221.

[34] J. K. GUPTA, M. EGOROV, AND M. KOCHENDERFER, *Cooperative multi-agent control using deep reinforcement learning*, in Autonomous Agents and Multiagent Systems, G. Sukthankar and J. A. Rodriguez-Aguilar, eds., Cham, 2017, Springer International Publishing, pp. 66–83.

[35] T. HAYNES AND S. SEN, *Evolving behavioral strategies in predators and prey*, in Adaption and Learning in Multi-Agent Systems, G. Weiß and S. Sen, eds., Berlin, Heidelberg, 1996, Springer Berlin Heidelberg, pp. 113–126.

[36] ——, *Learning cases to resolve conflicts and improve group behavior*, International Journal of Human-Computer Studies, 48 (1998), pp. 31–49.

[37] T. HAYNES, R. L. WAINWRIGHT, AND S. SEN, *Evolving cooperation strategies.*, in ICMAS, 1995, p. 450.

[38] T. HAYNES, R. L. WAINWRIGHT, S. SEN, AND D. A. SCHOENEFELD, *Strongly typed genetic programming in evolving cooperation strategies.*, in ICGA, vol. 95, 1995, pp. 271–278.

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

[40] Y. ISHIWAKA, T. SATO, AND Y. KAKAZU, *An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning*, Robotics and Autonomous Systems, 43 (2003), pp. 245 – 256.

[41] M. JADERBERG, W. M. CZARNECKI, I. DUNNING, L. MARRIS, G. LEVER, A. G. CASTANEDA, C. BEATTIE, N. C. RABINOWITZ, A. S. MORCOS, A. RUDERMAN, ET AL., *Human-level performance in 3d multiplayer games with population-based reinforcement learning*, Science, 364 (2019), pp. 859–865.

[42] J. JIANG, C. DUN, T. HUANG, AND Z. LU, *Graph convolutional reinforcement learning*, in International Conference on Learning Representations, 2019.

[43] V. KONDA AND J. TSITSIKLIS, *Actor-critic algorithms*, Advances in neural information processing systems, 12 (1999).

[44] R. E. KORF, *A simple solution to pursuit games*, in Working Papers of The 11th International Workshop on Distributed Artificial Intelligence, 1992, pp. 183–194.

[45] J. R. KOZA, *Genetic programming*, MIT Press, Cambridge, MA, 1992.

[46] J. Z. LEIBO, E. HUGHES, M. LANCTOT, AND T. GRAEPEL, *Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research*, arXiv preprint arXiv:1903.00742, (2019).

[47] R. LEVY AND J. S. ROSENSCHEIN, *A game theoretic approach to distributed artificial intelligence and the pursuit problem*, ACM SIGOIS Bulletin, 13 (1992), p. 11.

[48] J. E. LITTLEWOOD, *A mathematician's miscellany*, Methuen & Co. Ltd., London, 1953.

[49] R. LOWE, Y. WU, A. TAMAR, J. HARB, P. ABBEEL, AND I. MORDATCH, *Multi-agent actor-critic for mixed cooperative-competitive environments*, Neural Information Processing Systems (NIPS), (2017).

[50] X. MA, K. DRIGGS-CAMPBELL, AND M. J. KOCHENDERFER, *Improved robustness and safety for autonomous vehicle control with adversarial reinforcement learning*, in 2018 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2018, pp. 1665–1671.

[51] P. MAES, M. J. MATARIC, J.-A. MEYER, J. POLLACK, AND S. W. WILSON, *Co-evolution of pursuit and evasion ii: Simulation methods and results*, (1996).

[52] T. MAVRIDOU, P. PARDALOS, L. PITSOULIS, AND M. G. RESENDE, *A grasp for the biquadratic assignment problem*, European Journal of Operational Research, 105 (1998), pp. 613 – 621.

[53] G. F. MILLER AND D. CLIFF, *Co-evolution of pursuit and evasion I: Biological and game-theoretic foundations*, School of Cognitive and Computing Sciences, University of Sussex Brighton, 1994.

[54] ——, *Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics*, From animals to animats, 3 (1994), pp. 411–420.

[55] D. J. MONTANA, *Strongly typed genetic programming*, Evolutionary computation, 3 (1995), pp. 199–230.

[56] K. H. W. MYKEL J. KOCHENDERFER, TIM A. WHEELER, *Algorithms for Decision Making*, MIT Press, 2022.

[57] G. NITSCHKE, *Co-evolution of cooperation in a pursuit evasion game*, in Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453), vol. 2, IEEE, 2003, pp. 2037–2042.

[58] S. NOLFI AND D. FLOREANO, *How co-evolution can enhance the adaptive power of artificial evolution: Implications for evolutionary robotics*, in Evolutionary Robotics: First European Workshop, EvoRobot98 Paris, France, April 16–17, 1998 Proceedings 1, Springer, 1998, pp. 22–38.

[59] R. NOWAKOWSKI AND P. WINKLER, *Vertex-to-vertex pursuit in a graph*, Discrete Mathematics, 43 (1983), pp. 235 – 239.

[60] A. OROOJLOOY AND D. HAJINEZHAD, *A review of cooperative multi-agent deep reinforcement learning*, Applied Intelligence, (2022), pp. 1–46.

[61] E. OSAWA, *A metalevel coordination strategy for reactive cooperative planning.*, in ICMAS, vol. 95, 1995, pp. 297–303.

[62] X. PAN, D. SEITA, Y. GAO, AND J. CANNY, *Risk averse robust adversarial reinforcement learning*, in 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 8522–8528.

[63] C. H. PAPADIMITRIOU AND J. N. TSITSIKLIS, *The complexity of markov decision processes*, Mathematics of operations research, 12 (1987), pp. 441–450.

[64] P. M. PARDALOS AND L. S. PITSOULIS, *Nonlinear assignment problems: algorithms and applications*, vol. 7, Springer Science & Business Media, 2013.

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

[66] L. PINTO, J. DAVIDSON, R. SUKTHANKAR, AND A. GUPTA, *Robust adversarial reinforcement learning*, in International Conference on Machine Learning, PMLR, 2017, pp. 2817–2826.

[67] T. PITCHER, A. MAGURRAN, AND I. WINFIELD, *Fish in larger shoals find food faster*, Behavioral Ecology and Sociobiology, 10 (1982), pp. 149–151.

[68] A. QUILLIOT, *Jeux et pointes fixes sur les graphes*, PhD thesis, Université de Paris VI, 1978.

[69] T. RASHID, M. SAMVELYAN, C. SCHROEDER, G. FARQUHAR, J. FOERSTER, AND S. WHITESON, *Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning*, in International Conference on Machine Learning, PMLR, 2018, pp. 4295–4304.

[70] 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.

[71] ——, *Competition, coevolution and the game of tag*, in Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, 1994, pp. 59–69.

[72] S. RUSSELL AND P. NORVIG, *Artificial Intelligence: A Modern Approach, 4th Edition*, Pearson Education, 2021.

[73] M. SAMVELYAN, T. RASHID, C. SCHROEDER DE WITT, G. FARQUHAR, N. NARDELLI, T. G. RUDNER, C.-M. HUNG, P. H. TORR, J. FOERSTER, AND S. WHITESON, *The starcraft multi-agent challenge*, in Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019, pp. 2186–2188.

[74] J. SCHULMAN, P. MORITZ, S. LEVINE, M. JORDAN, AND P. ABBEEL, *High-dimensional continuous control using generalized advantage estimation*, arXiv preprint arXiv:1506.02438, (2015).

[75] J. SCHULMAN, F. WOLSKI, P. DHARIWAL, A. RADFORD, AND O. KLIMOV, *Proximal policy optimization algorithms*, arXiv preprint arXiv:1707.06347, (2017).

[76] S. SEUKEN AND S. ZILBERSTEIN, *Formal models and algorithms for decentralized control of multiple agents*, tech. rep., Technical Report 05-68, Department of Computer Science, University of Massachusetts Amherst, 2005.

[77] Y. SHI AND R. EBERHART, *A modified particle swarm optimizer*, in 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), May 1998, pp. 69–73.

[78] L. STEPHENS, *Agent organization as an effector of dai system performance*, in Proceedings of the 9th Workshop on Distributed Artificial Intelligence, 1989, 1989.

[79] L. M. STEPHENS AND M. B. MERX, *The effect of agent control strategy on the performance of a dai pursuit problem*, in Proceedings of the 10th International Workshop on Distributed Artificial Intelligence, 1990.

[80] P. STONE AND M. VELOSO, *Multiagent systems: A survey from a machine learning perspective*, Autonomous Robots, 8 (2000), pp. 345–383.

[81] S. SUKHBAATAR, Z. LIN, I. KOSTRIKOV, G. SYNNAEVE, A. SZLAM, AND R. FERGUS, *Intrinsic motivation and automatic curricula via asymmetric self-play*, arXiv preprint arXiv:1703.05407, (2017).

[82] L. SUN, Y.-C. CHANG, C. LYU, Y. SHI, Y. SHI, AND C.-T. LIN, *Toward multi-target self-organizing pursuit in a partially observable markov game*, arXiv preprint arXiv:2206.12330, (2022).

[83] L. SUN, C. LYU, AND Y. SHI, *Cooperative coevolution of real predator robots and virtual robots in the pursuit domain*, Applied Soft Computing, 89 (2020), p. 106098.

[84] L. SUN, C. LYU, Y. SHI, AND C.-T. LIN, *Multiple-preys pursuit based on bi-quadratic assignment problem*, in 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 1585–1592.

[85] R. S. SUTTON AND A. G. BARTO, *Reinforcement learning: An introduction*, MIT press, 2018.

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

[87] X. TANG, D. YE, L. HUANG, Z. SUN, AND J. SUN, *Pursuit-evasion game switching strategies for spacecraft with incomplete-information*, Aerospace Science and Technology, 119 (2021), p. 107112.

[88] J. TERRY, B. BLACK, N. GRAMMEL, M. JAYAKUMAR, A. HARI, R. SULLIVAN, L. S. SANTOS, C. DIEFFENDAHL, C. HORSCH, R. PEREZ-VICENTE, ET AL., *Pettingzoo: Gym for multi-agent reinforcement learning*, Advances in Neural Information Processing Systems, 34 (2021), pp. 15032–15043.

[89] 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, (2020).

[90] J. K. TERRY, N. GRAMMEL, A. HARI, L. SANTOS, AND B. BLACK, *Revisiting parameter sharing in multi-agent deep reinforcement learning*, arXiv preprint arXiv:2005.13625, (2020).

[91] C. UNDEGER AND F. POLAT, *Multi-agent real-time pursuit*, Autonomous Agents and Multi-Agent Systems, 21 (2010), pp. 69–107.

[92] O. VINYALS, I. BABUSCHKIN, W. M. CZARNECKI, M. MATHIEU, A. DUDZIK, J. CHUNG, D. H. CHOI, R. POWELL, T. EWALDS, P. GEORGIEV, ET AL., *Grandmaster level in starcraft ii using multi-agent reinforcement learning*, Nature, 575 (2019), pp. 350–354.

[93] X. WANG, Y. CHEN, AND W. ZHU, *A survey on curriculum learning*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 44 (2021), pp. 4555–4576.

[94] Z. WANG, B. GONG, Y. YUAN, AND X. DING, *Incomplete information pursuit-evasion game control for a space non-cooperative target*, Aerospace, 8 (2021).

[95] C. WU, A. R. KREIDIEH, K. PARVATE, E. VINITSKY, AND A. M. BAYEN, *Flow: A modular learning framework for mixed autonomy traffic*, IEEE Transactions on Robotics, 38 (2021), pp. 1270–1286.

[96] Y. YANG, R. LUO, M. LI, M. ZHOU, W. ZHANG, AND J. WANG, *Mean field multi-agent reinforcement learning*, in International conference on machine learning, PMLR, 2018, pp. 5571–5580.

[97] D. YE, M. SHI, AND Z. SUN, *Satellite proximate pursuit-evasion game with different thrust configurations*, Aerospace Science and Technology, 99 (2020), p. 105715.

[98] C. YU, A. VELU, E. VINITSKY, J. GAO, Y. WANG, A. BAYEN, AND Y. WU, *The surprising effectiveness of ppo in cooperative multi-agent games*, Advances in Neural Information Processing Systems, 35 (2022), pp. 24611–24624.

[99] K. ZHANG, Z. YANG, AND T. BAŞAR, *Multi-agent reinforcement learning: A selective overview of theories and algorithms*, Handbook of reinforcement learning and control, (2021), pp. 321–384.

[100] L. ZHENG, J. YANG, H. CAI, M. ZHOU, W. ZHANG, J. WANG, AND Y. YU, *Magent: A many-agent reinforcement learning platform for artificial collective intelligence*, Proceedings of the AAAI Conference on Artificial Intelligence, 32 (2018).

[101] Z. ZHOU AND H. XU, *Decentralized optimal large scale multi-player pursuit-evasion strategies: A mean field game approach with reinforcement learning*, Neurocomputing, (2021).