Multiple-Preys Pursuit based on Biquadratic Assignment Problem

Lijun Sun*‡, Chao Lyu†‡, Yuhui Shi‡ and Chin-Teng Lin*
*Faculty of Engineering and Information Technology
University of Technology Sydney, Sydney, Australia
Email: 13278371@student.uts.edu.au; Chin-Teng.Lin@uts.edu.au
†Harbin Institute of Technology, China
Email: 11849557@mail.sustech.edu.cn
‡Department of Computer Science and Engineering
Southern University of Science and Technology, Shenzhen, China
Email: shiyh@sustech.edu.cn

***Abstract*—The multiple-preys pursuit (MPP) is the adversarial game between predators and preys. If the capture of a prey is defined as that it cannot move anymore due to the surrounding of predators, there are two kinds of task allocations. One is about assigning which prey to which group of predators so that all preys can be captured. The other is about assigning which capturing position to which predator to encircle the prey simultaneously. In this paper, the MPP is modeled as a dynamic optimization problem and each its time step is solved in two stages. Firstly, the first kind of task allocation problem is modeled as the biquadratic assignment problem (BiQAP) and a MPP fitness function is proposed for the evaluation of such BiQAP task allocations. In this way, the MPP is transformed to several single-prey pursuit (SPP) problems. Secondly, for each SPP, we extend the coordinated SPP 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 predators. Through experiments of the current BiQAP solvers on the task allocation, we improve 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 MPP experiments.**

***Index Terms*—multiple-preys pursuit, biquadratic assignment problem (BiQAP), cooperative coevolutionary algorithm**

I. Introduction

The pursuit, or predator-prey domain is the game between two groups of agents. Typically, one group is called the predators or pursuers, and the other is called the preys or targets. As its name tells, the pursuit domain is about the story that predators want to capture the preys while the preys want to evade from predators. Based on their capabilities, the intelligence can occur between two adversarial groups or just among one group. Under different game setups, variants of the game have different applications and specific challenges to be coped with, and "many multi-agent systems (MAS) issues arise" in instances of the pursuit domain [1].

The pursuit domain is originally proposed by Benda et al. [2] and studied by successive work [3], [4] as a testbed for the optimal cooperation by correlating the organization structure of agents with the intelligence of the system. One of their conclusions is that the organization structure among agents is a crucial hyperparameter, the selection of which influences the selections of others, such as the coordination strategies. Afterwards, most work have used the pursuit domain as the testbed for the multi-agent coordination strategies design, in addition to the organization structure itself. One class of such work tries to implement the coordination with a group of independent agents that only considers two things: the agent itself and its own task, such as [5]–[7]. Although such implicit cooperation is simpler for the rules generation since the partnership is not incorporated in the design, i.e., no perception of other partners, nor communications, agents may fail to cope with many conflict scenarios and thus the final cooperation performances are not satisfactory.

Another class of work focuses on the strategies design for a group of cooperative agents. Tan [8] specially compared the performances of cooperative agents with that of independent agents, which had 2 main results that provide a direct motivation for the coordination strategies development of cooperative agents rather than independent agents. One is that for both normal tasks that can be accomplished by a single agent and joint tasks that need more than one agents, perception cooperation can help improve the efficiency in accomplishing tasks compared with a group of independent agents. The other is that sharing learned policies or experiences speeds up learning but does not influence the final task accomplishing performance. More successive work tried to answer the questions like what cooperative methods to use or how to be cooperative. For example, the game theoretic techniques used by Levy et al. [9], the case learning [10], the path planning methods [11], the reinforcement learning (RL) approaches [12]–[15], and the evolutionary computation (EC) [7], [16].

Corresponding author: Yuhui Shi.

The work is supported by the Science and Technology Innovation Committee Foundation of Shenzhen under Grant No.: ZDSYS201703031748284 and JCYJ20200109141235597, National Science Foundation of China under Grant No.: 61761136008, Australian Research Council (ARC) under discovery grant DP180100670.

Besides, another similar game to the pursuit domain is the pursuit-evasion game where agents have different names under different backgrounds. For example, a predator may be called a lion, a cop, a hunter, or a searcher, while a prey may be called a man, a robber, a rabbit, or a target, etc. As for their origins, the lion-and-man game was invented by R. Rado [17] where a lion wants to capture an evading man in a bounded circular arena. Then Parsons [18] studied the pursuit-evasion on graphs based on the application that searchers need to find a lost spelunker in a cave whose behavior is unpredictable, and claimed that the problem was raised by Richard Breisch. As for the pioneer work of the cops-and-robbers game, according to [19], it was investigated by Quilliot [20] and independently investigated by Nowakowski et al. [21]. A detailed survey on the pursuit-evasion can be seen in [22].

However, the pursuit domain primarily occurs in the grid world where the attention of researchers are mainly paid on the development of coordination strategies and results are empirical; while the pursuit-evasion game primarily occurs on a graph, or in an environment with some analyzable geometric properties where researchers concern more on the theoretical analyses and seek performance guarantees. In particular, there are many NP-hard problems in such theoretical analyses. For example, as shown in [19], [23], the determination of the minimum number of cops needed to win the game is NP-hard.

In this paper, we specially investigate the coordinated multiple-preys pursuit (MPP) problem, which is more challenging than the general studied single-prey pursuit (SPP) domain, with real-world applications such as sources of gas contamination clearance [22]. The game occurs in a finite grid world with 5 possible actions (moving one grid in the north, west, south, east, and keeping still) of agents. The capture of a prey is defined as that the prey is surrounded by predators such that it cannot move any more. So, generally, 4 predators are needed in capturing a prey, as shown in Fig. 1. The contributions of this paper are as follows. The MPP is modeled as a dynamic optimization problem and each its time step is solved by the proposed two-stage approach. Firstly, the task allocation problem of assigning each prey to 4 predators is modeled as the biquadratic assignment problem (BiQAP), and a multiple-preys pursing fitness function is proposed based on the single-prey pursuing function proposed in [16] to evaluate the multi-tasks coordination of agents in the BiQAP task allocation. In this way, the application of BiQAP is extended to the multiple robots systems (MRS) or multi-agent systems (MAS), and the MPP is transformed to several SPPs. Secondly, for each SPP, we modify the single-prey pursuing algorithm CCPSO-R (cooperative coevolutionary particle swarm optimization) [16] to allow the parallel observation, decision-making, and move of agents to further facilitate the MAS efficiency.

The rest of the paper is organized as follows. First, the BiQAP problem and the SPP strategy CCPSO-R are introduced in Section II. Then the proposed multiple-preys pursuing solution is described in Section III. Experimental results and discussions are presented in Section IV. Finally, conclusions, limitations, and future researches are given in Section V.

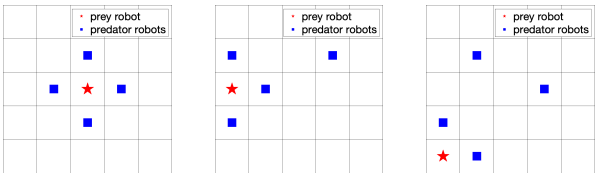


Fig. 1. Illustration of the surrounding based capture [16].

II. Related work

A. Biquadratic assignment problem (BiQAP)

The biquadratic assignment problem (BiQAP) [24]–[26] of size n is a combinatorial optimization problem, whose solution is typically represented as the permutation φ, which is the the bijective mapping between two sets $A = \{1, ..., n\}$ and $B = \{1, ..., n\}$, with the objective being

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

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 an 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. [24] 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 [24], [25], heuristic methods are pursued and reported in literature. In [27], 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 [28], 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 [26]–[28] on benchmarks generated by the method in [24], 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 MPP BiQAP task allocation, GRASP may be not the first-class choice, and the simpler the solver is the better, as will shown by the experimental results of Section IV.

B. Cooperative coevolutionary particle swarm optimization for robots (CCPSO-R)

In a single-prey pursuit, the fitness function for the j-th predator of the i-th subpopulation is designed in [16] as

$$f^{ij} = f^{ij}_{repel} \cdot (f^{ij}_{closure} + f^{ij}_{expanse} + f^{ij}_{uniformity}) \quad (2)$$

where

$$f^{ij}_{repel} = \begin{cases} e^{-c\cdot(NND^{ij}-D_{min})}, & \text{if } NND^{ij} < D_{min} \\ 1, & \text{else} \end{cases} \quad (3)$$

penalizes the fitness if the predator's nearest neighbor distance NND^{ij} is smaller than the specified minimum secure distance D_{min} with c being a constant;

$$f^{ij}_{closure} = inconv(\quad p_{prey}, \\ conv(p^{11}_{robots}, ..., p^{ij}_{robots}, ..., p^{N1}_{robots})) \quad (4)$$

evaluates whether the prey locates in the convex hull $conv(\cdot)$ shaped by the predators group: 0 if inside, 0.5 if on the edge, and 1 if outside (details see [16]);

$$f^{ij}_{expanse} = \frac{1}{N}(\sum_{k=1,k\neq i}^{N} |p^{k1}_{robots} - p_{prey}| + |p^{ij}_{robots} - p_{prey}|) \quad (5)$$

is the group expanse of predators in terms of the prey; and

$$f^{ij}_{uniformity} = std\left(\begin{bmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{bmatrix} \right), \quad (6)$$

or

$$f^{ij}_{uniformity} = \quad std([N_{12}, N_{21}, N_{23}, N_{32}]) \\ +std([N_{11}, N_{13}, N_{31}, N_{33}]). \quad (7)$$

evaluates the uniformity of predators' space distribution with respect to the prey by standard deviation $std(\cdot)$ where N_{ij} in the set $\{N_{11}, N_{12}, N_{21}, N_{22}\}$ or $\{N_{11}, N_{12}, N_{13}, ..., N_{33}\}$ is the number of predators in the (i, j)-th space bin divided by different schemes (details see [16]).

Algorithm 1: CCPSO-R [16]

```
Initialization.
while the termination conditions are not satisfied do
    for each subpopulation do
        Re-evaluate the subpopulation due to the
          environmental changes.
        Update each virtual predator robot.
        Update the real predator robot.
        Real robot deadlock detection and processing.
    Predator swarm deadlock detection and processing.
```

Based on the fitness function Eq (2), real predators pursue the prey without fixed behavior rules yet under the immediate guidance of the fitness function in CCPSO-R (Algorithm 1) [16]. That is, for one single prey pursuit with N predators, there are N subpopulations. In each subpopulation, the first individual is one distinct real predator, while all the others are virtual robots distributed in the vicinity of the real one. Under the scheme of cooperative coevolution (CC), robots evaluate the positions using Eq (2) in terms of all the other real predators of the other subpopulations such that the global benefit of the swarm is considered. Under the guidance of particle swarm optimization (PSO), virtual robots sample the vicinity and update their positions. Finally, the corresponding real robot chooses its locally optimal action based on the results of virtual robots that involve the global benefit of the whole swarm. The game continues until the maximum time limit is reached or the prey is captured.

III. The Proposed Method

A. The proposed solution for the multiple-preys pursuit

The multiple-preys pursuit (MPP) 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 a prey is defined as that the prey cannot move any more due to the surrounding of predators, as illustrated in Fig. 1. So, in general, 4 predators are needed for the capture of one prey. Once a prey is captured, it will not disappear. The MPP is said to be a success if all the preys are captured.

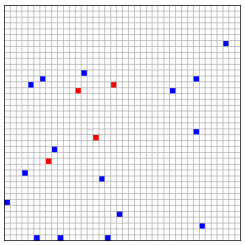


Fig. 2. An example multiple-preys pursuit scenario.

Algorithm 2: Proposed two-stage approach to MPP

```
Initialize the environment.
for each time step do
    for each prey do
        Observe, make decision, and take one action.
    The central virtual predator observes and makes the
      BiQAP task allocation decision by Algorithm 4.
    All the real predators 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 MPP, as shown in Fig. 2, firstly, tasks need to be allocated that each prey is assigned to each 4 predators and the MPP problem is transformed to several single-prey pursuit (SPP) problems. Due to the evaluation of such a task allocation involves the assignments between a prey and 4 predators, it can be naturally modeled as the biquadratic assignment problem (BiQAP). Secondly, all the predators will concurrently cooperate with their group members to capture

the assigned prey where a distributed coordinated single-prey pursuit strategy CCPSO-R [16] will be modified to its parallel version PCCPSO-R to improve the scalability. As the time goes, both preys and predators move, the environment changes, and thus both each single-prey pursuit fitness and the task allocation fitness change. So, for the MPP 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 preys are captured or time limit is reached. As a whole, the proposed two-stage approach to MPP is described in Algorithm 2, which will be introduced in detail in Section III-B and III-C.

B. BiQAP task allocation in the dynamic optimization

For the MPP, a central virtual predator is designed to allocate each 4 predators to the pursuit of each prey. Specially, if the set of predators is represented by $PREDATORS = \{1, 2, .., n\}$, and the set of preys is represented by $PREYS = \{1, 2, ..., m\}$ with $n = 4m$, then the mapping from $PREDATORS$ to $PREYS$ is a biquadratic semi-assignment problem (semi-BiQAP) [25], [29] since 4 predators will be assigned to the same prey.

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

$$\min_{\varphi \in S} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{o=1}^{n} c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)}. \tag{8}$$

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

$$c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)} \begin{cases} \neq 0, & \text{if } (i,j,k,o) \in K_1, \\ = 0, & \text{otherwise} \end{cases} \tag{9}$$

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 (8) can be simplified as

$$\min_{\varphi \in S} \sum_{(i,j,k,o) \in K_1} c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)}. \tag{10}$$

Further, since each prey is repeated 4 times in $\overline{PREYS}$, 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 (10) is equivalent to

$$\min_{\varphi \in S} \sum_{(i,j,k,o) \in K_2} c_{i,j,k,o,\varphi(i),\varphi(j),\varphi(k),\varphi(o)}, \tag{11}$$

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, we have $c = f^i$, where f^i is the fitness function for the i-th single-prey pursuit. Therefore, we propose the fitness function for the MPP as

$$f = \sum_{i \in \{1,2,...,n/4\}} f^i, \tag{12}$$

and the MPP BiQAP task allocation (11) is equivalent to

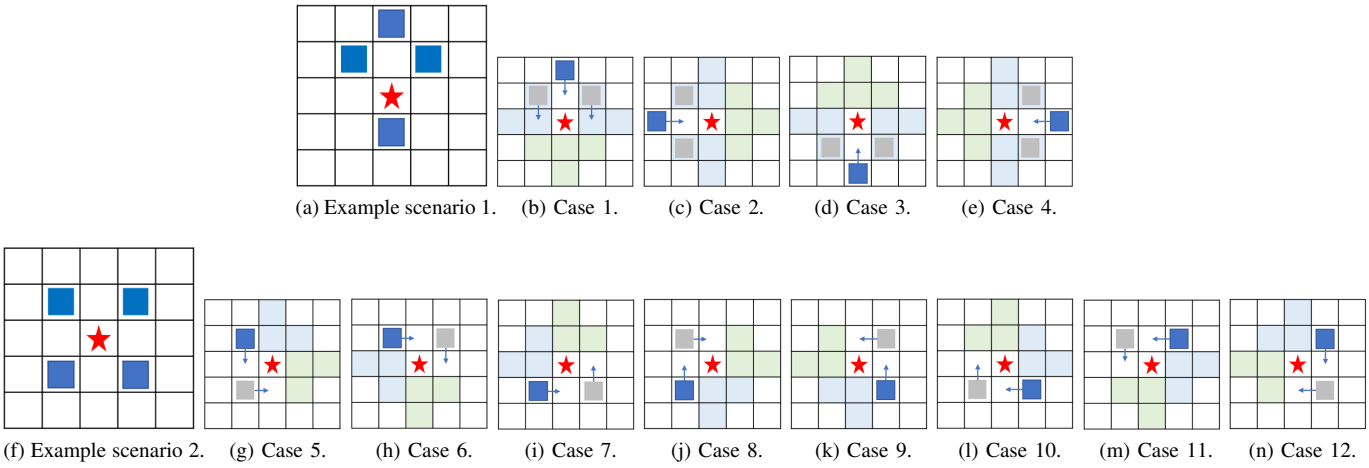
$$\min f. \tag{13}$$

In a summary, the one time step task allocation problem of assigning each prey to 4 predators is modeled as the BiQAP, which is an optimization problem with the objective (13). According to the convention in solving BiQAP, a solution is represented as a permutation of $\{1, 2, ..., n\}$. As for the MPP, it is a dynamic optimization problem where the fitness function Eq (12) changes every time step since the agents (predators and preys) are moving. To solve such dynamic optimization problem, based on the domain knowledge, we propose a scheme to greedily construct an initial good solution based on the space distributions of agents in Algorithm 3. However, note that, this procedure is conducted only once at the beginning of the game, while for successive time steps, since the MPP 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 3: Greedy inidividual construction scheme

```
1 M ← pairwise distances between preys and predators.
2 for each prey do
3     Assign the nearest 4 predators to the prey.
4     Delete these 4 predators from M.
```

For solving the BiQAP task allocation, we modify 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: (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 MPP. That is, we do 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 predators assigned to the same prey in the permutation solution does not change the single-prey pursuit fitness f^i and thus the task allocation fitness f.



(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.

Fig. 3. The local capture patterns in the repair stage of PCCPSO-R.

Algorithm 4: HEIDER-Random [27]

Construct an initial solution by Algorithm 3 as the current solution and set the flag *converged* as False.
Generate the two-positions interchanging set $I = \{(i,j)|i,j \in \{1,...,n\}, i < j, mod(i,4) \neq mod(j,4)\}$ and randomize its elements order for the semi-neighborhood searching.
while *not converged* **do**
    Set the flag *neighborhood_exhausted* as False.
    while *not neighborhood_exhausted* **do**
        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.
        if *the neighbor's fitness is better than the current solution* **then**
            neighborhood_exhausted ← True.
            The current solution ← the neighbor.
        else if I *is traversed* **then**
            neighborhood_exhausted ← True.
            converged ← True.
Output the current solution and its fitness.

C. Parallel CCPSO-R (PCCPSO-R)

As shown in Algorithm 1 and mentioned in [16], the sequential scheme in the observation, decision making, and moving of real predators limits the scalability of CCPSO-R. Therefore, we propose 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 1 in the following ways.

First, the single-prey pursuit function Eq (2) is modified as

$$f^{ij} = f^{ij}_{repel-prey} \cdot f^{ij}_{repel-predator} \cdot (f^{ij}_{closure} + f^{ij}_{expanse} + f^{ij}_{uniformity}) \quad (14)$$

where $f^{ij}_{repel-prey}$ and $f^{ij}_{repel-predator}$ have the same form of Eq (3) yet with different meanings of NND^{ij} and different values of the Manhattan distance D_{min}. For $f^{ij}_{repel-prey}$, NND^{ij} is the nearest distance to prey agents and the secure distance D_{min} is set as 1, while for $f^{ij}_{repel-predator}$, NND^{ij} is the nearest distance to predator agents and D_{min} is 2. However, note that, the function Eq (14) is for a predator's fitness in capturing a prey, while to evaluate the capturing fitness f^i of the whole predator swarm, $f^{ij}_{repel-prey} = f^{ij}_{repel-predator} = 1$.

Second, a repair stage is added when the predator swarm and their assigned prey 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. 3a, 4 local capture patterns Fig. 3b to Fig. 3e are extracted, while for scenarios like that shown in Fig. 3f, 8 patterns Fig. 3g to Fig. 3n are extracted. In these pattern cases, the red pentagram is the prey, the blue square with an moving direction arrow is the current predator, the grey square is also a predator that will prevent the current predator from moving to the next capturing position according to the configurations in Eq (14), and the shallow blue and shallow green filled cells represent the potential positions that a one-step away predator may locate for each distinct capturing position. Therefore, once a predator detects a matched local capturing pattern, it will enter the repair stage and behave according to the pointed moving direction such that all predators will move according to the rules to capture the prey without collisions.

IV. Experiments

A. 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 predators, or 4 times of the number of preys. For each test instance, an initial good solution is constructed using the method in Algorithm 3 whose fitness value is evaluated by the MPP fitness function Eq (12) and

TABLE I
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	

TABLE II
NO. OF MPP FITNESS EVALUATIONS (12) 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	

listed in the "Initialization" column of Table I. For GRASP, $\Lambda = 1000, \alpha = 0.25$ and $\beta = 0.3$ are used as in [26], [28]. For SIMANN3, its initial temperature is 2000 for the problems of size 12 and 5000 for the problems of size 16 as in [27]. The semi-neighborhood searching in Algorithm 4 is used in all the BiQAP solvers.

The experimental results are presented in Table I and Table II, 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 MPP 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, we list the best results and their ratios in the corresponding parentheses; meanwhile, the average results and standard deviations are listed in the "avg." columns and the corresponding 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 preys and $t(f^i)$ is the time in evaluating f^i of Eq (12). As shown in Fig. 4, $t(f^i) \approx 0.0005$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 MPP fitness evaluations Eq (12) are allowed if only one solution is evaluated per iteration, as listed in the "MaxIter" column in Table II. 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. In contrast, the computational cost of GRASP and SIMANN3 are too high to converge within 0.5s even for the problems of size 12.

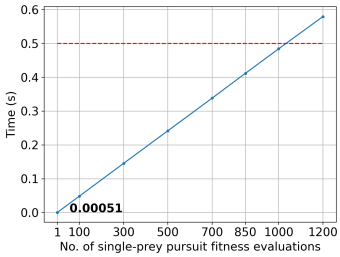


Fig. 4. The fitness evaluation time.

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. 5. 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 3. 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.

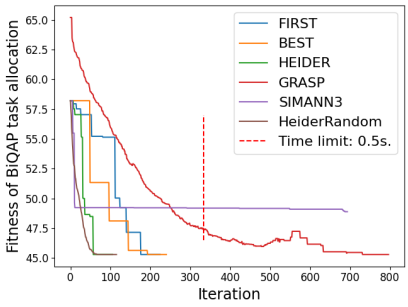


Fig. 5. The averaged fitness values of different algorithms during the optimization process over 50 runs.

B. PCCPSO-R vs. CCPSO-R

In this section, we compare the effectiveness of PCCPSO-R over CCPSO-R on the multiple random walking preys pursuit problems. The environmental instances in Section IV-A 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 preys. The experimental results are shown in Table III 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 III-C. 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 III
COMPARISON OF THE MULTIPLE-PREYS PURSUIT EFFICIENCY OF PCCPSO-R AND CCPSO-R

Problems	3-preys 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-preys 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)

V. CONCLUSIONS

In this paper, the MPP 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 prey to 4 predators by modeling such task allocation problem as the BiQAP, the solutions of which are evaluated by

the proposed multiple-preys pursuing fitness function. In this way, the MPP is transformed to several SPPs. In the second stage, each SPP is simultaneously solved by the assigned predators through PCCPSO-R, which parallelizes CCPSO-R to further improve the scalability of the predator's strategy to cooperatively capture the assigned prey. Since the MPP 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, we have 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 MPP. 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 predators by introducing two minimal secure distances in Eq (14), local capture patterns, and effective rules, these patterns are not complete to involve all possible local capturing scenarios. Therefore, the authors are currently working on solutions towards coping with these limitations, such as building fuzzy neural networks (FNNs) to automatically learn the local capturing patterns.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, Jun 2000.

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

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

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

[5] T. Haynes, R. L. Wainwright, and S. Sen, "Evolving cooperation strategies." in *ICMAS*, 1995, p. 450.

[6] 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.

[7] 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: Springer Berlin Heidelberg, 1996, pp. 113–126.

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

[9] R. Levy and J. S. Rosenschein, "A game theoretic approach to distributed artificial intelligence and the pursuit problem," *ACM SIGOIS Bulletin*, vol. 13, no. 3, p. 11, 1992.

[10] T. HAYNES and S. SEN, "Learning cases to resolve conflicts and improve group behavior," *International Journal of Human-Computer Studies*, vol. 48, no. 1, pp. 31 – 49, 1998.

[11] C. Undeger and F. Polat, "Multi-agent real-time pursuit," *Autonomous Agents and Multi-Agent Systems*, vol. 21, no. 1, pp. 69–107, Jul 2010.

[12] 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*, vol. 43, no. 4, pp. 245 – 256, 2003.

[13] S. Barrett, A. Rosenfeld, S. Kraus, and P. Stone, "Making friends on the fly: Cooperating with new teammates," *Artificial Intelligence*, vol. 242, pp. 132 – 171, 2017.

[14] M. Egorov, "Multi-agent deep reinforcement learning," *CS231n: Convolutional Neural Networks for Visual Recognition*, 2016.

[15] 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: Springer International Publishing, 2017, pp. 66–83.

[16] L. Sun, C. Lyu, and Y. Shi, "Cooperative coevolution of real predator robots and virtual robots in the pursuit domain," *Applied Soft Computing*, vol. 89, p. 106098, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494620300387

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

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

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

[20] A. Quilliot, "Jeux et pointes fixes sur les graphes," Ph.D. dissertation, Université de Paris VI, 1978.

[21] R. Nowakowski and P. Winkler, "Vertex-to-vertex pursuit in a graph," *Discrete Mathematics*, vol. 43, no. 2, pp. 235 – 239, 1983.

[22] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous robots*, vol. 31, no. 4, p. 299, 2011.

[23] 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: Springer US, 2008, pp. 171–185.

[24] 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.

[25] P. M. Pardalos and L. S. Pitsoulis, *Nonlinear assignment problems: algorithms and applications*. Springer Science & Business Media, 2013, vol. 7.

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

[27] R. E. Burkard and E. Çela, "Heuristics for biquadratic assignment problems and their computational comparison," *European Journal of Operational Research*, vol. 83, no. 2, pp. 283 – 300, 1995, eURO Summer Institute Combinatorial Optimization. [Online]. Available: http://www.sciencedirect.com/science/article/pii/037722179500007D

[28] T. Mavridou, P. Pardalos, L. Pitsoulis, and M. G. Resende, "A grasp for the biquadratic assignment problem," *European Journal of Operational Research*, vol. 105, no. 3, pp. 613 – 621, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221797000830

[29] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Society for Industrial and Applied Mathematics, 2012. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611972238