Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 964871, 13 pages
http://dx.doi.org/10.1155/2015/964871

Research Article

Multiagent Cooperative Learning Strategies for Pursuit-Evasion Games

Jong Yih Kuo,[1] Hsiang-Fu Yu,[2] Kevin Fong-Rey Liu,[3] and Fang-Wen Lee[1]

[1] *Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei, Taiwan*
[2] *Department of Computer Science, National Taipei University of Education, Taipei, Taiwan*
[3] *Department of Safety, Health and Environmental Engineering, Ming Chi University of Technology, New Taipei City, Taiwan*

Correspondence should be addressed to Jong Yih Kuo; jykuo@ntut.edu.tw

Received 20 June 2014; Revised 17 September 2014; Accepted 12 October 2014

Academic Editor: Saeed Balochian

This study examines the pursuit-evasion problem for coordinating multiple robotic pursuers to locate and track a nonadversarial mobile evader in a dynamic environment. Two kinds of pursuit strategies are proposed, one for agents that cooperate with each other and the other for agents that operate independently. This work further employs the probabilistic theory to analyze the uncertain state information about the pursuers and the evaders and uses case-based reasoning to equip agents with memories and learning abilities. According to the concepts of assimilation and accommodation, both positive-angle and bevel-angle strategies are developed to assist agents in adapting to their environment effectively. The case study analysis uses the Recursive Porous Agent Simulation Toolkit (REPAST) to implement a multiagent system and demonstrates superior performance of the proposed approaches to the pursuit-evasion game.

1. Introduction

A multiagent system (MAS) comprises a set of agents that interact with each other. These agents may either share a common goal or have contradictory objectives [1, 2]. This work deals with cooperative agents trying to achieve a common goal. When coordinating multiple agents as a team for the same task, the agents must have the ability to handle unknown and uncertain situations and take the success of the whole team into account.

A multiagent pursuit-evasion game involves guiding one group of agents (pursuers) to cooperate with each other to catch another group of agents (evaders). However, this game varies with the type of environment in which it is played (e.g., plane, grid, and graph), the knowledge of the players (e.g., the evaders' position and strategy), the controllability of their motions (is there a limit on their speed? and can they make turns whenever they want?), and the meaning of a capture (are the evaders to be intercepted, seen, or surrounded?). Being complex and dynamic, pursuit-evasion problems are difficult to solve [3]. To address such problem, the robotics community proposed several models [4, 5]. In these models, the motion of the evader is usually modeled by a stochastic process. There has been growing interest in modeling the game, in which the evader is intelligent and has certain sensing capabilities [6]. Solutions to the problem proposed in other studies [7–9] included competitive coevolution, multiagent strategies, and multiagent communication algorithms.

The hybrid learning approach to RoboCup [10] includes a coach agent and multiple moving agents. Using case-based reasoning and genetic algorithms (CBR-GA), the coach agent decides on a strategy goal and assigns tasks to the moving agents. Every moving agent then executes its respective task to achieve the strategy goal. The proposed method also includes two kinds of agents (pursuer and evader). Unlike the hybrid learning approach to RoboCup, the proposed method does not use the coach agent to guide pursuers in catching evaders. The pursuers must search for the evaders themselves. If other evaders/pursuers are not in the sensing area of a pursuer, the pursuer does not know their positions. The pursuer thus uses the case-based reasoning with assimilation and accommodation to catch evaders.

Owing to the uncertain environment and the dynamic information of agents at each moment, previous research has mainly examined how to improve the efficiency of communication and accuracy in the learning process. In a different manner, this study focuses on agents with a mental state and the ability to plan evolution by using strategic modules and combining learning methods. Armed with such capability, the agents can adapt to the dynamic environment.

The rest of the paper is organized as follows. Section 2 reviews the background and related works of pursuit-evasion problems. Section 3 introduces the agent's adaptive learning process. Section 4 describes the case study of the pursuit-evasion game. The conclusions and directions for future work are presented in the last section.

2. Background and Related Work

Pursuit-evasion problems have long been studied from the perspective of differential game theories. Nishimura and Ikegami observed random swarming and other collective predator motions in a prey-predator game model [11]. Jim and Giles [12] proposed a simple effective predator strategy, which could enable predators to move to the closest capture position. This strategy does not prove to be very successful because the predators may block each other when they try to move to the same capture position. Haynes and Sen [13] used genetic programming to evolve strategies for both predators and preys. Hládek et al. [14] developed a multiagent control system using fuzzy inference for a group of two-wheeled mobile robots to execute a common task. They defined a pursuit-evasion task using fuzzy sets to establish a framework for inserting and updating expert knowledge in the form of rules by an inference system. Antoniades et al. [3] proposed several pursuit algorithms for solving complex multiplayer pursuit-evasion games. Their work revealed that reducing sensing overlap between pursuers and avoiding overassignment of pursuers to target locations could improve the capture time. According to the Contract Net Protocol (CNP) [15], Zhou et al. [16] proposed a multiagent cooperative pursuit algorithm and extended the CNP by improving alliance decision and forming dynamic alliance. They used the support decision matrix that includes credits of agents, degrees of hookup, and degrees of self-confidence to help agents make decisions during the negotiation process.

Rao and Georgeff [17] proposed the belief-desire-intention (BDI) model for the agents in a distributed artificial intelligence environment. As its name implies, the BDI agent model involves belief, desire, and intention. In addition, the assimilation and accommodation approaches to the pursuit-evasion game were proposed by Piaget [18]. These two approaches equip an agent with a mental state and the ability to plan an evolution process. Using different learning methods, the agent can also adapt to the dynamic environment effectively.

2.1. BDI Agent Model. On the foundation of BDI, Kuo et al. [19, 20] proposed an agent, which can be completely specified to fulfill its intentions by events it perceives, actions it performs, beliefs it holds, goals it adopts, and plans it has. A goal module describes the goals that an agent may adopt, as well as the events to which it can respond. A belief module includes the information about the internal state that an agent of a certain class holds, the strategies it may perform, and the environment it is in. A plan module generates the plans that an agent may employ to achieve its goals. A plan is a sequence of actions or strategies derived through a reasoning mechanism.

In the proposed approach, *belief* denotes the agent's knowledge of the environment, including the orientations and locations of evaders; *desire* represents the agent's wish to catch all evaders, and *intention* stands for the agent's plan of actions.

2.2. Assimilation and Accommodation. According to Piaget's cognitive theory [18], each recognition system aims at equilibration of the inconsistent information about the world. Thus, an organism seeks not only adaptation (harmony of organism and world) but also organization (harmony within itself). Assimilation and accommodation represent forms of the maintenance and modification of these cognitive schemata [21]. Takamuku and Arkin [22] applied assimilation on domestic robots for social learning, enabling the robots to perform well under various situations. This study proposes both bevel-angle and positive-angle accommodation strategies for modifying the agent's cognitive structure.

2.3. Pursuit-Evasion Game. In a pursuit-evasion game, pursuers try to capture evaders by besieging them from all directions in a grid world. The game focuses mainly on the effectiveness of structures, with varying degrees of pursuers' cooperation and control, to entrap evaders efficiently [13]. The first mathematical formulation of graph searching was proposed by Parsons [23] in 1978. The formulation was inspired by an earlier research by Breisch [24] who put forward an approach to finding an explorer lost in a complicated system of dark caves.

Furthermore, the pursuit-evasion game varies with different approaches to searching evaders as described in the following.

2.3.1. Node Search and Mixed Search. Ferrari [25] proposed a new family for path planning algorithms. One algorithm is for searching the goal, and the other is for searching the obstacles. These algorithms can be utilized to parameterize easily the potential scale length and strength, thus providing better control over the moving object path.

To address some unique demands from the game domain, Walsh and Banerjee [26] presented a new algorithm, called "VRA*" algorithm, for path finding on game maps, and proposed also the extension of a postsmoothing technique. Wong et al. presented a Bee Colony Optimization (BCO) algorithm for the symmetrical Traveling Salesman Problem (TSP) [27]. The BCO model is constructed algorithmically according to the collective intelligence observed from the foraging behavior of bees. The algorithm is integrated with

a fixed-radius near-neighbor 2-opt (FRNN 2-opt) heuristic to further improve prior solutions generated by the BCO model.

Raboin et al. [28] considered multiagent pursuit scenarios in which there is a team of tracker agents and a moving target agent. The trackers observe continuously the target until the target is at least one tracker's observation range at the end of the game. Their study described formalism and algorithms for game-tree search in partially observable Euclidean space.

Gerkey et al. [29] introduced a new class of searchers, the φ-searchers, which can be readily instantiated as a physical mobile robot, and proposed the first complete search algorithm for a single φ-searcher. They also showed how this algorithm can be extended to handle multiple searchers and gave various examples of computed trajectories. Their work aimed at coordinating teams of robots to execute tasks in application domains, such as clearing a building, for reasons of security or safety.

2.3.2. Game Theoretic. Game theoretic approaches to patrolling have increasingly become an interesting topic in recent years. Basilico et al. [30] presented a game theoretic scheme to determine the optimal patrolling strategy for a mobile robot that operates in environments with arbitrary topologies.

Prior research [31] proposed a game theory-based approach, which uses a multirobot system to perform multitarget search in a dynamic environment. A dynamic programming equation is employed to estimate the utility function, which considered the a priori probability map, travel costs, and current decisions of other robots. According to this utility function, a utility matrix can be calculated for an N-robot nonzero-sum game. Thus, pure Nash equilibrium and mixed-strategy equilibrium can be utilized to guide the robots in making their decisions.

3. Adaptive Agent Learning

This section introduces the adaptive agent model, the refinement of previously developed agent models [19, 20]. This adaptive agent model contains a hybrid approach for a multiagent learning method. In addition, this model enables agents to learn and accumulate their experience and knowledge from other agents or the environment.

3.1. Adaptive Agent Model. The agent model, shown in Figure 1, is a cooperative learning model responsible for controlling the learning process. A goal module returns the goals that an agent may possibly adopt and the events to which it can respond. A belief module describes the information about the environment, the internal state that a certain class of agents may hold, and the strategies or tactics that the agent may perform. A plan module returns plans that are possibly employed to achieve the goals of the agent. A plan is a sequence of actions or strategies derived through reasoning. There are two types of ontology that provide the domain-specific and issue-specific knowledge, respectively.

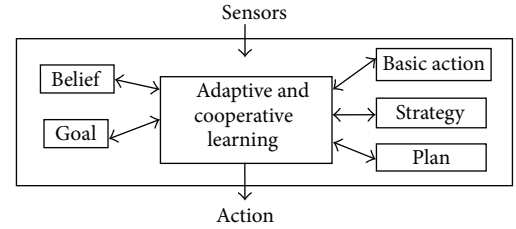

Figure 1: Agent model.

The mental state of an agent is represented by first-order language as follows:

$$FOTerm ::= \langle Const\rangle \mid \langle FOV\rangle \mid functor\left\langle FuncSym \times seq\ FOTerms\right\rangle, \quad (1)$$

$$Atom ::= \left\langle Predicate \times seq\ FOTerms\right\rangle.$$

A first-order term (*FOTerm*) is defined by a set of constants ($\langle Const\rangle$), first-order variables ($\langle FOV\rangle$), and functions ($functor\left\langle FuncSym \times seq\ FOTerms\right\rangle$). An atom consists of a predicate and a set of *FOTerms*.

3.1.1. Belief. The beliefs of an agent describe the situations where the agent is. The beliefs are specified by a belief base, which contains the information the agent believes about the world, as well as the information that is internal to the agent. The agent is assumed to have beliefs about its task and its environment. A belief is represented by a set of well-formed formulas such as

$$Belief ::= \langle Atom\rangle \mid \neg\langle Atom\rangle \mid \langle Atom\rangle\left[\wedge\langle Atom\rangle\right]. \quad (2)$$

For instance, a belief that there is an obstacle (*obstacle_id*) at a certain location $(X1, Y1)$ can be represented as $(obstacle\_id, X1, Y1)$.

3.1.2. Goal. A goal is an agent's desire and describes the states of affairs that an agent would like to realize. For instance, CatchAllEvader(True) stands for a pursuer's goal to catch all evaders. A goal is represented by a set of well-formed formulas such as

$$Goal ::= achieved\ \langle Atom\rangle. \quad (3)$$

3.1.3. Basic Actions. Basic actions are generally used by an agent to manipulate its environment. Before performing the basic actions, certain beliefs should be held. The beliefs of the agent must be updated after the execution of actions. Basic actions are the only entities that can change the beliefs of an agent. Let α be an action with parameters x, and let $\varphi, \psi \in belief$. The programming constructs for basic actions are then expressed as

$$\{\varphi\}\,\alpha(x)\,\{\psi\}, \quad (4)$$

where φ and ψ are preconditional and postconditional, respectively. The precondition indicates that an action cannot

be performed only if certain beliefs are held. For example, a pursuer is assumed to catch an evader from location $(X0, Y0)$ to location $(X1, Y1)$ with a basic action $MoveTo(X1, Y1)$. Before this action is executed, the pursuer must be at position $(X0, Y0)$, denoted by $pursuer(self, X0, Y0)$. The condition that the pursuer senses the evader at position $(X1, Y1)$ is denoted by $evaderAt(X1, Y1)$. This action is then defined in terms of the following beliefs:

$$\{pursuer\,(self, X0, Y0)\,, evaderAt\,(X1, Y1)\}\,. \quad (5)$$

After executing an action, an agent must update its beliefs to make its postcondition true. According to the above example, after performing the catch action, the agent updates its beliefs and obtains

$$\{pursuer\,(self, X1, Y1)\,, evaderAt\,(X2, Y2)\}\,, \quad (6)$$

where $(X2, Y2)$ is an evader's new location.

3.1.4. Strategies. The strategy of a player refers to one of the options that can be chosen in a setting. The choice depends not only on the agent's own actions but also on other agents' actions. A prior study [28] proposed a pure strategy in terms of a function. Given an agent's current information set, the function returns the agent's next move (i.e., its change in location between time t_j and t_{j+1}). For example, if an evader has a pure strategy σ_0, then $\theta_0(t_{j+1}) = \theta_0(t_j) + \sigma_0(I_0(t_j))$, where $\theta_0(t_j)$ is the location of tracker agent 0 at time t_j and $I_0(t_j)$ is the tracker's information set at time t_j. Suppose that a pursuer has a pure strategy set $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_k)$ and the locations of the tracker agents $\theta(t) = \{\theta_1(t), \ldots, \theta_k(t)\}$. Then, $\theta(t_{j+1}) = \theta(t_j) + \sigma(I_0(t_j))$ for each time t_j. The strategy is further refined by adding an algorithm for selecting a plan according to different conditions. The algorithm can be represented by a function that generates a plan. The condition includes the beliefs of agents or the current situation of the environment. The proposed strategy is denoted by

$$Strategy ::= \langle conditions\rangle\ function. \quad (7)$$

3.1.5. Plans. A plan is a method for an agent to approach its goal and is generated according to the agent's strategy. A plan consists of beliefs, actions, and rewards, as expressed in the following equation:

$$Plan ::= \langle belief\rangle\ \langle basic\ action\rangle\ [\langle basic\ action\rangle]\ \langle reward\rangle\,. \quad (8)$$

A plan can be one or a set of actions for an agent, and the value of the plan is determined by the reward.

3.2. Case-Based Reasoning. According to previous studies [32, 33], a general case-based reasoning (CBR) cycle may include retrieving the most similar cases for a problem, reusing the information and knowledge in those cases to solve the problem, revising the proposed solution, and retaining the parts of this experience likely to be useful for future problem solving.

In order to solve a new problem, the cases which contain the most useful knowledge have to be identified first. Since the utility of a case cannot be evaluated directly a priori, the similarity between problem descriptions is used in heuristics to estimate the expected utility of the cases. Therefore, the quality of this measure is crucial for the success of CBR applications. There are numerous similarity measures in use today. They are used not only in CBR but also in other fields, including data mining, pattern recognition, genetic algorithm, and machine learning. Euclidean distance is a typical similarity measure for objects in a Euclidean space [34]. According to Euclidean distance, this work devises a simple distance function to evaluate the similarity between two cases.

3.2.1. Case Description. A case stands for an agent's mental state and the result of its past output. Each case consists of a goal, beliefs, an action, a plan, and a reward. The goal is what the agent wants to achieve or realize. The beliefs describe the situation the agent is in. The action is what the agent does under that situation. The plan is the approach the agent takes to achieve the goal. The reward serves to evaluate the result of the plan.

3.2.2. Similarity Relation. Consider the following:

$$\mathrm{Sim}_{ab} = \sum_{i=1}^{n} w_i \times \left(\left|a_i - b_i\right|\right). \quad (9)$$

This study proposes an evaluation function (Sim), shown as (9), for measuring the similarity between cases a and b. In this equation, a and b represent a new problem and a case in a case base, respectively. The variable n is the number of features in case a. The weight variable w_i stands for the importance of the ith feature. The weight vector is freely defined by a user. a_i and b_i are the ith features of cases a and b, respectively. In the case-retrieving stage of the CBR approach, the case with the smallest (Sim) value is always retrieved for reuse.

3.3. Probabilistic Framework. Suppose that a finite two-dimensional environment EX with n_c square cells contains an unknown number of fixed obstacles. We also assume that the environment has n_p pursuers and n_e evaders. Let x_{pk} and x_{ei} be the cells occupied by pursuer k and evader i, where $1 \le i \le n_p$ and $1 \le j \le n_e$. The pursuers and evaders are restricted to move to cells not occupied by obstacles. Each pursuer collects information about EX at discrete time instances $t \in T = \{1, 2, \ldots, t_{\mathrm{end}}\}$.

Define $x_{pk}(t) \subset \mathrm{EX}$ as the cell of pursuer k at time t and $x_{ei}(t) \subset \mathrm{EX}$ as the cell of evader i at time t. $o(t)$ is a set of cells where obstacles are detected. Let $U(x_{e_i}(t))$ denote the one-step reachable set for the evader when the evader is in cell x

at time t. Then the probability of the evader being in cell x at time $t + 1$ is given by the following (10):

$$p\left(x, t+1 \mid x_{e_i}(t)\right) = \begin{cases} \dfrac{1}{\left|U\left(x_{e_i}(t)\right) - o(t)\right|} & x \in U\left(x_{e_i}(t)\right) \\ 0 & x \in o(t), \end{cases} \quad (10)$$

where $p(x, t + 1 \mid x_{e_i}(t))$ represents the probability of the evader being in cell x at time $t + 1$ according to the location where the pursuer detects the evader at time t.

3.4. Strategies for Plan Generation. When the distance between the retrieved case and the new case is smaller than a threshold, the above approach cannot generate a useful plan for the new case. To overcome this problem, this study proposes two plan-generation strategies, local-max strategy and local-cooperative strategy.

3.4.1. Local-Max Strategy. This strategy is used only when there is a single pursuer for evaders. Let $S_k(y)$ be the set of all cells that lie within the sensing area of a pursuer k at cell y. The total evasion probability of the evaders at time t associated with the pursuer is then obtained by the following (11):

$$P(y, t) = \sum_{z \in S_k(y)} \left[p\left(z, t+1 \mid x_{e_i}(t)\right)\right]. \quad (11)$$

By computing the evasion possibilities of all sensing cells, pursuer k moves to cell $x_{pk}(t + 1)$ that has the highest total evasion probability expressed as follows:

$$x_{p_k}(t+1) = \arg\max_{y \in U(x_{p_k}(t))} \left[P(y, t)\right]. \quad (12)$$

3.4.2. Local-Cooperative Strategy. This strategy is for pursuers that cooperate with each other to catch the same evader. Initially, pursuers need to decide whether they can cooperate with each other or not. Let $d(x, y)$ be the Manhattan distance from cell x to cell y; that is, if cells x and y lie in a two-dimensional space with coordinates (x_1, x_2) and (y_1, y_2), the Manhattan distance is calculated using the following equation:

$$d(x, y) = \max\left(\left|x_1 - y_1\right|, \left|x_2 - y_2\right|\right). \quad (13)$$

Let S_k be the farthest sensing cell of a pursuer k. For example, if the sensing area of a pursuer k is a $5 * 5$ square and the pursuer stands at the center of the area, the farthest distance that the pursuer can detect is two squares. If there exists two or more pursuers and one evader corresponding with $d(x_{p_k}(t), x_{e_i}(t)) \leq s_k$ and $d(x_{p_k}(t), x_{e_i}(t)) \leq s_k$, the pursuers can cooperate with each other.

Suppose that two or more pursuers execute the local-cooperative strategy. Let Overlap(x, y) be the number of overlap cells within the one-step reachable set (a $3 * 3$ region) around cells x and y. The overlap function is illustrated in Figure 2. The yellow grid represents the one-step reachable set for a pursuer $P1$, the dashed grid depicts the one-step reachable set for a pursuer $P2$, and the green grid represents the position of an evader E. The number of overlap cells thus equals four (i.e., the return value of Overlap(x, y)).

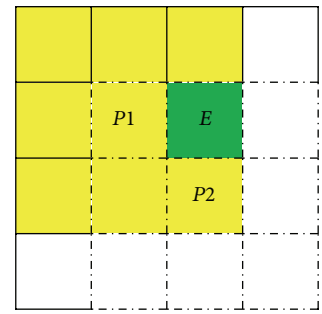

Figure 2: Number of overlap cells.

Once the pursuers can cooperate with each other, the pursuers randomly select one of them as a leader. Other pursuers can determine their locations at time $t + 1$ by the following (14):

$$x_{p_k}(t+1) = \arg\min_{y \in U(x_{p_k}(t))} \left[\text{Overlap}\left(x_{p_k}(t+1), y\right)\right], \quad (14)$$

where $x_{p_k}(t + 1)$ is the location of the leader. To avoid pursuers moving to the same location, each pursuer finds the minimum overlap area.

3.5. Reward Calculation. Each case has its own reward for evaluating its result. The higher the reward of a case is, the more likely the goal will be achieved. In this study, the reward is calculated using the following equation:

$$R = \left|S_k(y) \cap U\left(x_{e_i}(t)\right)\right|. \quad (15)$$

Reward R represents the number of one-step reachable sets for an evader in a pursuer's sensing area. The reward value is between 0 and 10. If an evader is caught, then the reward of the case will be 10.

3.6. Assimilation and Accommodation. Piaget's cognitive theory [18] reveals that assimilation refers to the tendency to interpret experience as much as possible through existing cognitive structure of knowing. When an agent faces a new situation, it generates a new plan with the current cognitive structure, also called its strategy, to direct itself what to do next. The entire process is called assimilation. In other words, an agent assimilates a new situation with its current cognitive structure. However, if the current cognitive structure cannot explain the environment (i.e., the current cognitive structure cannot get the equilibration of inconsistent information about the world), an agent has to use the strategy of accommodation. Accommodation refers to the realization that the current structure is insufficient for adequate understanding of the world and that the current structure must be changed until it can assimilate the new situation.

This study proposes two accommodation methods, bevel-angle and positive-angle strategies, for modifying a cognitive structure. After performing accommodation, an agent employs the modified strategy to assimilate the new situation. By adjusting constantly its cognitive structure, the agent can adapt more effectively to the environment.

3.6.1. Bevel-Angle Strategy. Suppose that an evader's location is in the one-step reachable set for a pursuer, and the evader is at a bevel angle of the pursuer. The pursuer randomly chooses one of the following strategies when the evader is at 45° ($\pi/4$), 135° ($3\pi/4$), 225° ($5\pi/4$), and 315° ($7\pi/4$) of the pursuer. Let $\deg(x_{p_k}(t), x_{e_i}(t))$ be the angle between the pursuer k and evader i, where $x_{e_i}(t)$ is the position of the evader and $x_{p_k}(t)$ is the position of the pursuer at time t.

Strategy 1. According to (16), the pursuer moves to the evader's position at time $t+1$:

$$x_{p_k}(t+1) = x_{e_i}(t), \quad \text{if } x_{e_i}(t) \in U\left(x_{p_k}(t)\right) \quad \deg\left(x_{p_k}(t), x_{e_i}(t)\right) \in \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}. \tag{16}$$

Strategy 2. According to (17), the pursuer moves to 45° of the evader's position at time $t+1$. Let x_j be an element of the one-step reachable set for the pursuer k when it is in cell x at time t:

$$x_{p_k}(t+1) = x_j, \quad \text{if } x_{e_i}(t) \in U\left(x_{p_k}(t)\right) \cap \deg\left(x_{p_k}(t), x_{e_i}(t)\right) \in \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4} \cap x_j \in U\left(x_{p_k}(t)\right) \cap \deg\left(x_{p_k}(t), x_j\right) = \deg\left(x_{p_k}(t), x_{e_i}(t)\right) + 45. \tag{17}$$

Strategy 3. According to (18), the pursuer moves to −45° of the evader's position at time $t+1$:

$$x_{p_k}(t+1) = x_j, \quad \text{if } x_{e_i}(t) \in U\left(x_{p_k}(t)\right) \cap \deg\left(x_{p_k}(t), x_{e_i}(t)\right) \in \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4} \cap x_j \in U\left(x_{p_k}(t)\right) \cap \deg\left(x_{p_k}(t), x_j\right) = \deg\left(x_{p_k}(t), x_{e_i}(t)\right) - 45. \tag{18}$$

Strategy 4. According to (19), the pursuer stays in the same place at time $t+1$:

$$x_{p_k}(t+1) = x_{p_k}(t), \quad \text{if } x_{e_i}(t) \in U\left(x_{p_k}(t)\right) \quad \deg\left(x_{p_k}(t), x_{e_i}(t)\right) \in \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}. \tag{19}$$

3.6.2. Positive-Angle Strategy. Suppose that an evader's location is in the one-step reachable set for a pursuer, and the evader is at a positive angle of the pursuer, where the angle includes 0°, 90° ($\pi/2$), 180° (π), and 270° ($3\pi/2$).

Strategy 1. According to (20), the pursuer moves to the evader's position at time $t+1$:

$$x_{p_k}(t+1) = x_{e_i}(t), \quad \text{if } x_{e_i}(t) \in U\left(x_{p_k}(t)\right) \quad \deg\left(x_{p_k}(t), x_{e_i}(t)\right) \in 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}. \tag{20}$$

Strategy 2. According to (21), the pursuer moves to 45° of the evader's position at time $t+1$:

$$x_{p_k}(t+1) = x_j, \quad \text{if } x_{e_i}(t) \in U\left(x_{p_k}(t)\right) \cap \deg\left(x_{p_k}(t), x_{e_i}(t)\right) \in 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2} \cap x_j \in U\left(x_{p_k}(t)\right) \cap \deg\left(x_{p_k}(t), x_j\right) = \deg\left(x_{p_k}(t), x_{e_i}(t)\right) + 45. \tag{21}$$

Strategy 3. According to (22), the pursuer moves to −45° of the evader's position at time $t+1$:

$$x_{p_k}(t+1) = x_j, \quad \text{if } x_{e_i}(t) \in U\left(x_{p_k}(t)\right) \cap \deg\left(x_{p_k}(t), x_{e_i}(t)\right) \in 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2} \cap x_j \in U\left(x_{p_k}(t)\right) \cap \deg\left(x_{p_k}(t), x_j\right) = \deg\left(x_{p_k}(t), x_{e_i}(t)\right) - 45. \tag{22}$$

Strategy 4. According to (23), the pursuer moves to 90° of the evader's position at time $t + 1$.

$$\begin{aligned} & x_{p_k}(t+1) = x_j, \\ & \quad \text{if } x_{e_i}(t) \in U\left(x_{p_k}(t)\right) \\ & \qquad \cap \deg\left(x_{p_k}(t), x_{e_i}(t)\right) \in 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2} \\ & \qquad \cap x_j \in U\left(x_{p_k}(t)\right) \\ & \qquad \cap \deg\left(x_{p_k}(t), x_j\right) = \deg\left(x_{p_k}(t), x_{e_i}(t)\right) + 90. \end{aligned} \tag{23}$$

Strategy 5. According to (24), the pursuer moves to −90° of the evader's position at time $t + 1$:

$$\begin{aligned} & x_{p_k}(t+1) = x_j, \\ & \quad \text{if } x_{e_i}(t) \in U\left(x_{p_k}(t)\right) \\ & \qquad \cap \deg\left(x_{p_k}(t), x_{e_i}(t)\right) \in 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2} \\ & \qquad \cap x_j \in U\left(x_{p_k}(t)\right) \\ & \qquad \cap \deg\left(x_{p_k}(t), x_j\right) = \deg\left(x_{p_k}(t), x_{e_i}(t)\right) - 90. \end{aligned} \tag{24}$$

Strategy 6. According to (25), the pursuer stays in the same place at time $t + 1$:

$$\begin{aligned} & x_{p_k}(t+1) = x_{p_k}(t), \\ & \quad \text{if } x_{e_i}(t) \in U\left(x_{p_k}(t)\right) \\ & \qquad \deg\left(x_{p_k}(t), x_{e_i}(t)\right) \in 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}. \end{aligned} \tag{25}$$

Combining both the bevel-angle and positive-angle strategies, the pursuer obtains 24 different plans to adapt to the environment effectively.

3.7. Plan Evolution Process. Plan evolution plays an important role in an agent's life cycle. When a pursuer finds an evader in the sensing area, it generates a pursuit plan. Initially, the pursuer searches its case base for similar cases. If similar cases exist, the pursuer chooses one of them as the plan. Otherwise, the pursuer uses one of the abovementioned strategies to generate a plan. When the pursuer faces a new state, the pursuer will adjust the plan accordingly. Figure 3 shows the agent's plan evolution process.

(1) Problem Analysis. Before determining the location in the next step, a pursuer analyzes the environmental state and updates its goal with information from the environment. The pursuer chooses a plan according to its mental state.

(2) Case Retrieval. Comparing the similarity between the result of the analysis in *Step 1* and the cases in the case base, the pursuer retrieves similar cases by the (Sim) function and sorts them according to their similarity levels.

(3) Similarity Calculation. Similarity is classified into low and high levels. It is possible that the pursuer retrieves a case with high-level similarity but obtains unsatisfactory results, and thus a reward R is added to prevent such. If the case is with high-level similarity and the reward R exceeds a predefined constant ε, the pursuer executes *Step 4*. If the case is with low-level similarity or the reward R is smaller than the predefined constant ε, the pursuer executes *Step 5*.

(4) Case Reuse. If the retrieved case is with high-level similarity and its result is acceptable, the pursuer reuses the case with revisions and then goes to *Step 10*.

(5) Strategy Assimilation. If no similar case exists in the case base or the result of the similar case is unsatisfactory, the pursuer uses strategy assimilation; that is, the pursuer generates a suitable plan with the current strategy (*Steps 6–8*) according to the situation.

(6) Cooperation. Before using the strategy module, the pursuer decides whether to cooperate with the others or not. If the decision is to cooperate with the others in the situation, then the local-cooperative strategy (i.e., *Step 8*) is executed. Otherwise, the pursuer executes the local-max strategy (i.e., *Step 7*).

(7) Application of the Local-Max Strategy. If the pursuer decides not to cooperate with the others, it executes the local-max strategy and moves to the cell in the one-step reachable set with the highest probability of containing an evader. Then, go to *Step 9*.

(8) Application of the Local-Cooperative Strategy. If the cooperative mode is taken, the pursuer executes the local-cooperative strategy and moves to the cell which has the minimum overlap sensing area.

(9) Reward Calculation. The pursuer evaluates the plan and calculates a reward accordingly.

(10) Plan Transformation. The pursuer represents the plan by beliefs, actions, and rewards.

(11) Strategy Assessment. The pursuer evaluates whether the current strategy is suitable or not. The pursuer counts the number C of consecutive failures in catching an evader by using the strategy. If C is smaller than a constant σ, the strategy is effective. Then, go to *Step 13*. Otherwise, the strategy is accommodated by *Step 12*.

(12) Strategy Accommodation. The pursuer modifies the strategy to generate a new plan for the current environment.

(13) Case Revision. The actions and reward of the case are revised after the problem is solved.

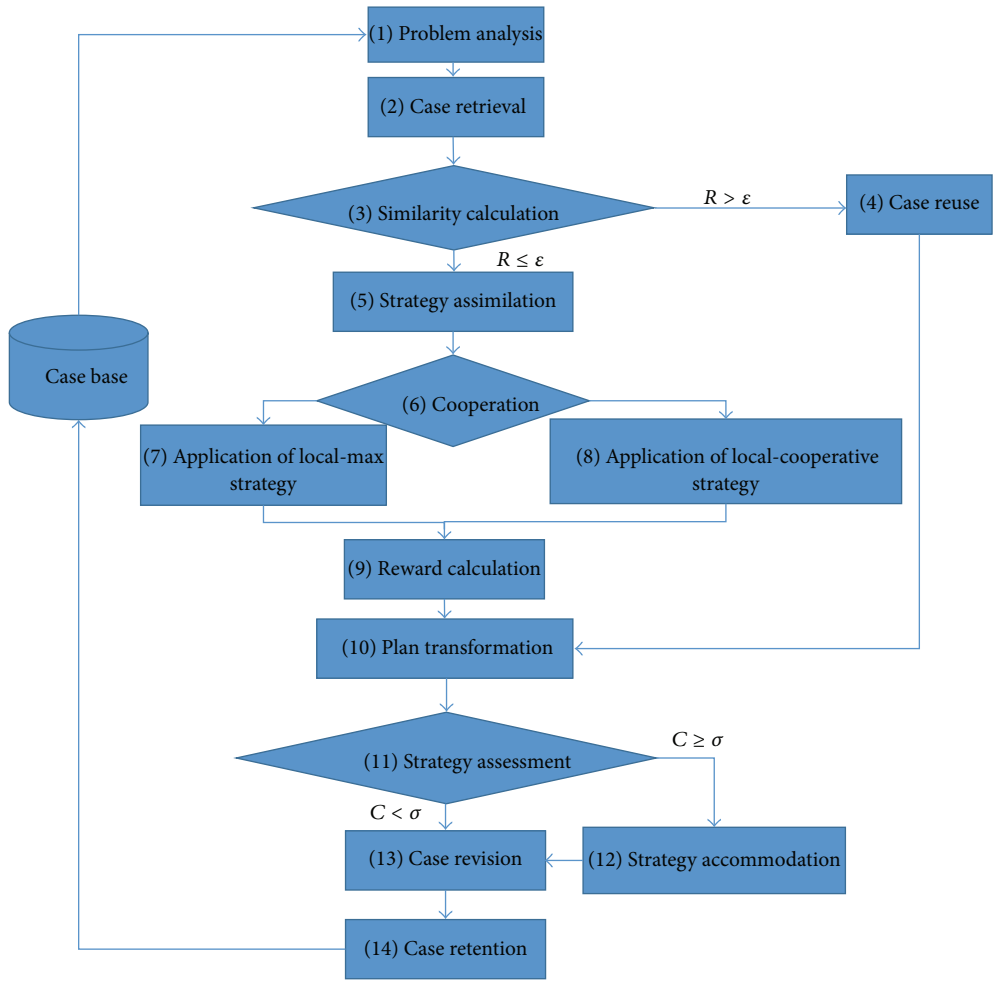

Figure 3: Plan evolution process.

(14) Case Retention. Parts of the experience that are likely to be useful for future problem solving are retained.

4. Case Study

This study uses a pursuit-evasion game to demonstrate the proposed approach and implements a multiagent system on the Recursive Porous Agent Simulation Toolkit (REPAST) [35]. Figure 4 shows the GUI of the system, where the blue, red, and black circles represent pursuers, evaders, and obstacles, respectively. The yellow region depicts the sensing area of the pursuers.

4.1. Problem Formulation. The system contains multiple pursuing robots, which search for multiple evading robots in a two-dimensional environment. The physical world is abstracted as a grid world, and the game is played according to the following settings and rules.

(i) The number of pursuers is n. The number of evaders is unknown to the pursuers.

(ii) Each pursuer has a sensing area comprising $5 * 5$ cells, and the pursuer is at the center of the area, as shown in Figure 4.

(iii) During each moving step, the agents are allowed to either move one cell away from their current cells or stay in their current cells.

(iv) Each pursuer has no information about the environment other than that of its own 5-by-5 sensing area.

(v) All pursuers use the same case base.

(vi) An evader is captured when it occupies the same cell as a pursuer, and it will be removed from the environment.

(vii) The game ends when all evaders have been captured.

4.2. Evasion Policy. When an evader is caught, the system first checks whether the sum of the caught and uncaught evaders is smaller than the total number of the predefined evaders. If so, the system generates a new evader and chooses its type of movement. Otherwise, the system does nothing. Clearly, when the sum of the caught and uncaught evaders

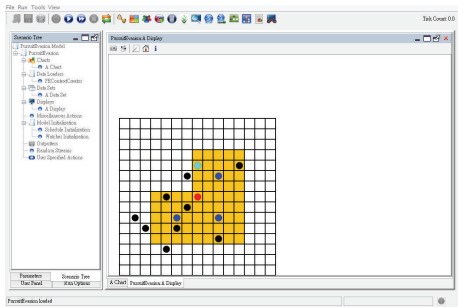

Figure 4: Implementation of multiagent system.

equals the number of the predefined evaders, the system stops generating new evaders and the uncaught evaders do not increase. The uncaught evaders even decrease with the successful evader captures. The game thus ends finally. In addition, when the system selects a movement type for a new evader, the choice varies with different experiments, which in turn affects the outcome of the experience. This study investigates four types of movement, namely, random, clockwise, counterclockwise, and smart movements.

Random Movement. An evader randomly selects a location that can be reached in one step or just stays in the same location. The details of the algorithm are shown in Algorithm 1.

Clockwise Movement. An evader moves clockwise in a square path. The evader initially selects the direction of its first step and sets up a distance randomly. If the evader runs into an obstacle on the way, it changes its direction automatically. Algorithm 2 shows the algorithm.

Counterclockwise Movement. This movement type is similar to the clockwise movement except that an evader moves counterclockwise. The algorithm is shown in Algorithm 3.

Smart Moving. Evaders try actively to avoid pursuers. Each evader has a square sensing area. Each side of the square is three cells long. If the evader finds any pursuer in the area, it attempts to move to the location that has fewer pursuers in the one-step reachable set of the area. Algorithm 4 presents the algorithm.

4.3. Results and Discussion. This study designs several experiments to compare the performance of the proposed approach, Hysteretic Q-Learning [36], and Reinforcement-Learning fusing (RL-Fusing) [37]. All these approaches are the learning techniques for multiagent systems. Each experiment has three pursuers and one evader. In addition, the experiments run in "time step" unit. One round is considered finished when the evader has been caught. The performance results are the average values obtained by running 30 rounds with the current learning policy after 100 training episodes for each algorithm.

4.3.1. Comparison of Approaches. Matignon et al. presented a decentralized reinforcement learning algorithm for independent learning robots called Hysteretic Q-Learning [36]. This algorithm computes a better policy in a cooperative multiagent system without additional information or communication between agents. Hysteretic Q-Learning updates the equation for an agent i executing the action a_i from states ending up in state s' as follows:

$$\delta \longleftarrow r + \gamma \max_{a'} Q_i\left(s', a'\right) - Q_i\left(s, a_i\right),$$
$$Q_i\left(s, a_i\right) \longleftarrow \begin{cases} Q_i\left(s, a_i\right) + \text{inc}\delta & \text{if } \delta > 0 \\ Q_i\left(s, a_i\right) + \text{dec}\delta & \text{else,} \end{cases} \tag{26}$$

where $Q_i(s, a_i)$ is the action value function of state s and action a_i, r is the reward it receives, $\gamma \in [0; 1]$ is the discount factor, and inc and dec are the rates of increase and decrease in Q-values, respectively. Partalas et al. proposed the RL-Fusing approach that uses coordinated actions and a fusing process to guide the agents [37]. The fusion process combines the decisions of all agents and then outputs a global decision, which the agents must follow as a team. Algorithm 5 shows the RL-Fusing algorithm. The set of strategies means the plans.

4.3.2. Obstacles. The first experiment compares the performance of the proposed approach, Hysteretic Q-Learning, and RL-Fusing in environments with/without obstacles. The evader is assumed to move randomly in every round. In the environment with obstacles, 10 obstacles are randomly set up initially, and no evader is trapped in the obstacles.

Table 1 shows the performance results. As can be seen, all three methods require more capture steps in the environment with obstacles, because pursuers need extra time to detect and avoid obstacles. Hysteretic Q-Learning takes the greatest number of capture steps in both environments because it does not support pursuer cooperation. Furthermore, RL-Fusing also takes more capture steps than the proposed scheme. Although RL-Fusing involves cooperation between pursuers, the generated strategies sometimes cause a pursuer to miss evaders in its sensing area. The pursuer thus needs to search again and requires more capture steps. In contrast, the proposed approach utilizes not only the cooperative strategy but also assimilation and accommodation which enable a pursuer to learn from either success or failure experiences. Thus, the pursuer can keep in step with an evader and catch it in a smaller number of capture steps.

4.3.3. Movement Types. The second experiment measures the average number of capture steps required by the proposed approach, Hysteretic Q-Learning, and RL-Fusing under two types of evader movement. One is clockwise movement, and the other is random clockwise and counterclockwise movement.

The average number of capture steps required by the three approaches for different movement types is shown in Table 2. Similar to the results obtained in the first experiment, the proposed scheme outperforms Hysteretic Q-Learning and RL-Fusing. Among the three approaches, Hysteretic Q-Learning requires the highest number of capture steps for

```
(1) while true do
(2)     next location ← generates randomly a location nearby the evader
(3)         if next location has an obstacle then
(4)             next location ← generates randomly a new location nearby the evader
(5)         end if
(6) end while
```

Algorithm 1: Random movement algorithm.

```
(1) direction ← generates randomly a direction
(2) distance ← generates randomly a distance shorter than the environment
(3) move ← 0
(4) while true do
(5)     get the next location according to the direction
(6)     move add 1
(7)         if move equals to distance or next location has an obstacle then
(8)             evader turns right
(9)             changes direction to evader
(10)            move ← 0
(11)        end if
(12) end while
```

Algorithm 2: Clockwise movement algorithm.

```
(1) direction ← generates randomly a direction
(2) distance ← generates randomly a distance shorter than environment
(3) move ← 0
(4) while true do
(7)         if move equals to distance or next location has obstacle then
(8)             evader turns left
(9)             change direction to evader
(10)            move ← 0
(11)        end if
(12) end while
```

Algorithm 3: Counterclockwise movement algorithm.

```
(1) while true do
(2)         if evader's sensor area has pursuer then
(3)             next location ← generate a location with least pursuers detected
(4)         else
(5)             next location ← randomly generate a location nearby the evader
(6)         end if
(7) end while
```

Algorithm 4: Smart movement algorithm.

the same reason that it does not support agent cooperation. According to the RL-Fusing algorithm presented in Algorithm 5, when a state is not the coordinated state, the action of an agent chooses a random or predefined policy and a single pursuer thus has no learning ability. Therefore, the possibility of catching an evader is low, and RL-Fusing also takes more capture steps. In contrast, when the pursuer of the proposed scheme faces an evader, the pursuer observes the evader continuously. If the evader changes its moving type, the pursuer uses accommodation to change its catching

```
Require: A set of strategies, an initial state s_0
(1) s ← s_0
(2) while true do
(3)     if s is coordinated state then
(4)         ρ ← RandomReal(0, 1)
(5)         if ρ < ε then
(6)             select a strategy randomly    //the same for all agent
(7)         else
(8)             rank strategies
(9)             communicate ranks
(10)            receive ranks from other agents
(11)            average ranks
(12)            select corresponding strategy
(13)        end if
(14)        execute selected strategy
(15)        receive reward
(16)        transit to a new state s'
(17)        update Q-value
(18)    else
(19)        act with a predefined or random policy
(20)    end if
(21) end while
```

Algorithm 5: RL-Fusing algorithm.

Table 1: Average number of capture steps in different environments.

	Grid size	Without obstacles (steps)	With obstacles (steps)
Hysteretic Q-Learning	12 ∗ 12	68	73
RL-Fusing	12 ∗ 12	61	65
The proposed approach	12 ∗ 12	24	27

Table 2: Average number of capture steps for different movement types.

	Grid size	Clockwise movement (steps)	Random clockwise and counterclockwise movement (steps)
Hysteretic Q-Learning	12 ∗ 12	56	69
RL-Fusing	12 ∗ 12	48	56
The proposed approach	12 ∗ 12	26	32

strategy. If the moving type remains fixed, the pursuer keeps the same strategy by assimilation. As seen in Table 2, all three methods require more steps to capture evaders with random clockwise and counterclockwise movement because pursuers need extra time to detect the moving directions of the evaders.

Table 3: Average number of capture steps for evaders with smart movement in different grid sizes.

	Grid size: 12 ∗ 12 (steps)	Grid size: 24 ∗ 24 (steps)
Hysteretic Q-Learning	74	212
RL-Fusing	63	300
The proposed approach	63	119

Table 4: Average number of capture steps with/without assimilation and accommodation.

	Evader movement type Clockwise	Evader movement type Smart
With assimilation and accommodation	18.6	27.8
Without assimilation and accommodation	24.2	36.4

4.3.4. Smart Movement and Size of Grid. The third experiment involves evaders with smart movement. The evaders try actively to avoid pursuers when they sense any pursuer. The environment is of two grid sizes, 12 ∗ 12 and 24 ∗ 24.

Table 3 shows that the proposed scheme requires smaller capture steps in different grid sizes, when compared with Hysteretic Q-Learning and RL-Fusing. When the grid size is smaller, RL-Fusing outperforms Hysteretic Q-Learning. However, as the grid size becomes larger, RL-Fusing requires more capture steps than Hysteretic Q-Learning. This is

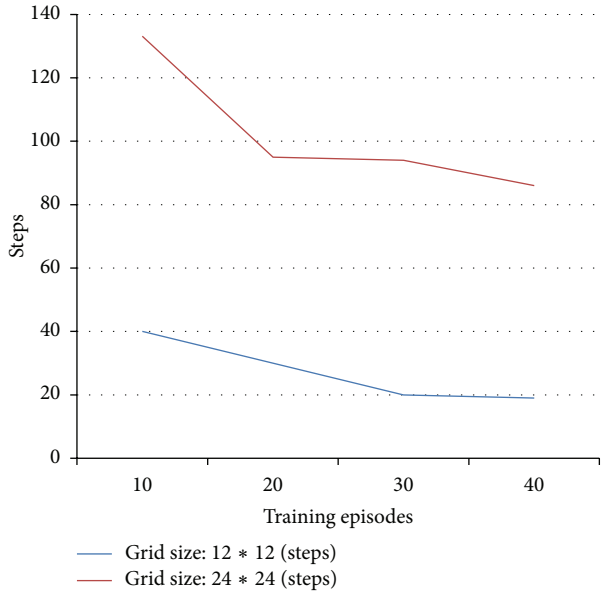

Figure 5: Average number of capture steps under different training episodes and grid sizes.

because RL-Fusing depends mainly on its cooperative strategies to catch evaders, but not all strategies can guide pursuers to move close to the evaders. For example, strategy 3 in RL-Fusing [37] is that once all the pursuers go at the distance of 3 from an evader. Because the sensing area of a pursuer is a 5 ∗ 5 square, the pursuers easily lose the evader in a larger environment. In contrast, the proposed approach assists the pursuers to move as close to the evader as possible by the avoidance of sensing overlapping. Thus, the approach has the highest possibility of catching the evader.

4.3.5. Smart Movement, Training Episodes, and Size of Grid. The fourth experiment investigates the relationship between training episodes and grid sizes when using the proposed approach.

Figure 5 shows that regardless of grid sizes (12 ∗ 12 or 24 ∗ 24), the greater the number of training episodes is, the smaller the average number of capture steps is required. This is because more training enriches the case base, thus enabling pursuers to make better strategies and decisions.

4.3.6. Assimilation and Accommodation. The last experiment evaluates the proposed approach with/without assimilation and accommodation. There are two types of evader movement, clockwise and smart. The results are listed in Table 4. Whether the approach is with or without assimilation and accommodation, the average number of capture steps for smart movement is greater than that for clockwise movement. This is because smart movement can enable an evader to dynamically change its movement and position. In contrast, movement of an evader with clockwise movement can be easily predicted; hence, the pursuers can take fewer steps for capture. Furthermore, the average number of capture steps required by the approach with assimilation and accommodation is smaller than that required by the approach without assimilation and accommodation for both movement types. The approach without assimilation and accommodation generates a fixed plan, which cannot be changed. In contrast, the approach with assimilation and accommodation enables the first pursuer to use the local-max strategy to find a better position, and the remaining pursuers can determine the cooperative positions according to the first pursuer's position. These pursuers can adjust their positions according to different catching situations and strategies. Accordingly, the approach with assimilation and accommodation requires fewer capture steps, when compared with the approach without assimilation and accommodation.

5. Conclusion

This work proposes an effective strategy learning approach to the pursuit-evasion game in a dynamic environment. This method can facilitate a pursuer in learning and implementing effective plans from either success or failure experiences under the condition of no knowledge of environment available. The evolved plans fare extremely well when compared with some of the best constructed strategies. This study also demonstrates the superior performance of the proposed approach for the pursuit-evasion game by implementing a multiagent system on REPAST.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by the Ministry of Science and Technology under Grant no. 102-2218-E-027-007.

References

[1] K. P. Sycara, "Multiagent systems," *AI Magazine*, vol. 19, no. 2, pp. 79–92, 1998.

[2] N. Vlassis, *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool, 2007.

[3] A. Antoniades, H. J. Kim, and S. Sastry, "Pursuit-evasion strategies for teams of multiple agents with incomplete information," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, pp. 756–761, December 2003.

[4] L. J. Guibas, J.-C. Latombe, S. M. Lavalle, D. Lin, and R. Motwani, "A visibility-based pursuit-evasion problem," *International Journal of Computational Geometry & Applications*, vol. 9, no. 4-5, pp. 471–493, 1999.

[5] J. P. Hespanha, G. J. Pappas, and M. Prandini, "Greedy control for hybrid pur-suit-evasion games," in *Proceedings of the European Control Conference*, pp. 2621–2626, 2001.

[6] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: theory, implementation,

and experimental evaluation," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 662–669, 2002.

[7] R. E. Korf, "A simple solution to pursuit games," in *Proceedings of the 2nd International Workshop on Distributed Artificial Intelligence*, pp. 183–194, 1992.

[8] S. Nolfi and D. Floreano, "Coevolving predator and prey robots: do "arms races" arise in artificial evolution?" *Artificial Life*, vol. 4, no. 4, pp. 311–335, 1998.

[9] N. J. Savill and P. Hogeweg, "Evolutionary stagnation due to pattern-pattern interactions in a coevolutionary predator-prey model," *Artificial Life*, vol. 3, no. 2, pp. 81–100, 1997.

[10] J. Y. Kuo, F.-C. Huang, S.-P. Ma, and Y.-Y. Fanjiang, "Applying hybrid learning approach to RoboCup's strategy," *Journal of Systems and Software*, vol. 86, no. 7, pp. 1933–1944, 2013.

[11] S. I. Nishimura and T. Ikegami, "Emergence of collective strategies in a prey-predator game model," *Artificial Life*, vol. 3, no. 4, pp. 243–260, 1998.

[12] K.-C. Jim and C. L. Giles, "Talking helps: evolving communicating agents for the predator-prey pursuit problem," *Artificial Life*, vol. 6, no. 3, pp. 237–254, 2000.

[13] T. Haynes and S. Sen, "Evolving behavioral strategies in predators and prey," in *Adaption and Learning in Multi-Agent Systems*, vol. 1042 of *Lecture Notes in Computer Science*, pp. 113–126, Springer, Berlin, Germany, 1995.

[14] D. Hládek, J. Vaščák, and P. Sinčák, "Hierarchical fuzzy inference system for robotic pursuit evasion task," in *Proceedings of the 6th International Symposium on Applied Machine Intelligence and Informatics (SAMI '08)*, pp. 273–277, Herl'any, Slovakia, January 2008.

[15] R. G. Smith, "The contract net protocol: high-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. 19, no. 12, pp. 1104–1113, 1980.

[16] P.-C. Zhou, B.-R. Hong, Y.-H. Wang, and T. Zhou, "Multi-agent coopoerative pursuit based on extended contract net protocol," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, vol. 1, pp. 169–173, August 2004.

[17] A. Rao and M. Georgeff, "BDI agents: from theory to practice," in *Proceedings of the International Conference on Multi-Agent Systems*, pp. 312–319, 1995.

[18] J. Piaget, *The Equilibration of Cognitive Structures: The Central Problem of Intellectual Development*, University of Chicago Press, 1985.

[19] J. Y. Kuo, S. J. Lee, C. L. Wu, N. L. Hsueh, and J. Lee, "Evolutionary agents for intelligent transport systems," *International Journal of Fuzzy Systems*, vol. 7, no. 2, pp. 85–93, 2005.

[20] J. Y. Kuo and F. C. Huang, "A hybrid approach for multi-agent learning systems," *Intelligent Automation and Soft Computing*, vol. 17, no. 3, pp. 385–399, 2011.

[21] S. Gebhardt, P. Grant, R. von Georgi, and M. T. Huber, "Aspects of Piaget's cognitive developmental psychology and neurobiology of psychotic disorders—an integrative model," *Medical Hypotheses*, vol. 71, no. 3, pp. 426–433, 2008.

[22] S. Takamuku and R. C. Arkin, "Multi-method learning and assimilation," *Robotics and Autonomous Systems*, vol. 55, no. 8, pp. 618–627, 2007.

[23] T. Parsons, "Pursuit-evasion in a graph," in *Theory and Applications of Graphs*, vol. 642 of *Lecture Notes in Mathematics*, pp. 426–441, Springer, Heidelberg, Germany, 1978.

[24] R. Breisch, "An intuitive approach to speleotopology," *Southwestern Cavers*, vol. 6, no. 5, pp. 72–78, 1967.

[25] F. Ferrari, "A new parameterized potential family for path planning algorithms," *International Journal on Artificial Intelligence Tools*, vol. 18, no. 6, pp. 949–957, 2009.

[26] K. Walsh and B. Banerjee, "Fast A∗ with iterative resolution for navigation," *International Journal on Artificial Intelligence Tools*, vol. 19, no. 1, pp. 101–119, 2010.

[27] L.-P. Wong, M. Y. H. Low, and C. S. Chong, "Bee colony optimization with local search for traveling salesman problem," *International Journal on Artificial Intelligence Tools*, vol. 19, no. 3, pp. 305–334, 2010.

[28] E. Raboin, D. S. Nau, U. Kuter, S. K. Gupta, and P. Svec, "Strategy generation in multi-agent imperfect-information pursuit games," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 947–954, 2010.

[29] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of View," *International Journal of Robotics Research*, vol. 25, no. 4, pp. 299–315, 2006.

[30] N. Basilico, N. Gatti, and F. Amigoni, "Leader-follower strategies for robotic patrolling in environments with arbitrary topologies," in *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '09)*, pp. 57–64, May 2009.

[31] M. Yan, "Multi-robot searching using game-theory based approach," *International Journal of Advanced Robotic Systems*, vol. 5, no. 4, pp. 341–350, 2008.

[32] K.-D. Althoff, "Case-based reasoning," in *Handbook on Software Engineering and Knowledge Engineering*, pp. 549–587, 2001.

[33] E. K. Burke, B. L. MacCarthy, S. Petrovic, and R. Qu, "Multiple-retrieval case-based reasoning for course timetabling problems," *Journal of the Operational Research Society*, vol. 57, no. 2, pp. 148–162, 2006.

[34] J. Long, D. G. Schwartz, S. Stoecklin, and M. K. Patel, "Application of loop reduction to learning program behaviors for anomaly detection," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC '05)*, vol. 1, pp. 691–696, April 2005.

[35] The Repast Suite, The Repast Suite, 2014, http://repast.sourceforge.net/.

[36] L. Matignon, G. J. Laurent, and N. L. Fort-Piat, "Hysteretic Q-Learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '07)*, pp. 64–69, San Diego, Calif, USA, October-November 2007.

[37] I. Partalas, I. Feneris, and I. Vlahavas, "A hybrid multiagent reinforcement learning approach using strategies and fusion," *International Journal on Artificial Intelligence Tools*, vol. 17, no. 5, pp. 945–962, 2008.