Collaborating and Learning Predators on a Pursuit Scenario

Nugroho Fredivianus, Urban Richter, and Hartmut Schmeck

Institute of Applied Informatics and Formal Description Methods (AIFB)
Karlsruhe Institute of Technology (KIT)
76128 Karlsruhe, Germany
{Nugroho.Fredivianus,Urban.Richter,Hartmut.Schmeck}@kit.edu
http://www.aifb.kit.edu

Abstract. A generic predator/prey pursuit scenario is used to validate a common learning approach using Wilson's eXtended Learning Classifier System (XCS). The predators, having only local information, should independently learn and act while at the same time they are urged to collaborate and to capture the prey. Since learning from scratch is often a time consuming process, the common learning approach, as investigated here, is compared to an individual learning approach of selfish learning agents. A special focus is set on the performance of how quickly the team goal is achieved in both learning scenarios. This paper provides new insights of how agents with local information could learn collaboratively in a dynamically changing multi-agent environment. Furthermore, the concept of a common rule base based on Wilson's XCS is investigated. The results based on the common rule base approach show a significant speed up in the learning performance but may be significantly inferior on the long run, in particular in situations with a moving prey.
Keywords. Multi-agent learning, predator/prey pursuit scenario, emergent behavior, collaboration, and XCS.

1 Motivation

Due to the increasing scale and complexity of strongly interconnected application systems there is a need for intelligent distributed information processing and control. The design of multi-agent systems (MASs) has addressed this need, using concepts from machine learning and distributed artificial intelligence [1]. MASs have been utilized successfully in a range of application scenarios: Guiding automated machines in collaborative industry scenarios [2], trading energy on market platforms [3], seeking smallest distance routes for delivery services [4], or managing air conditioners in buildings [5], are some examples of problems which are solved (completely or partially) using MAS approaches.

A MAS consists of a collection of agents acting autonomously within their common environment in order to meet their objectives. They take sensory inputs from the environment, match them on actions, and then perform some actions, which again affect their environment [6]. Here, an agent does not always represent

a physical entity. It could be a virtual one, defined by a piece of software, or even some lines of a program. The predator/prey scenario [7] has been shown to provide a generic scenario as a basis for fundamental research on MASs, capturing essential aspects of many potential fields of application (cf. [8]).

In this paper, we investigate different aspects of learning in predator/prey scenarios. Each predator collects experiences while trying to capture the prey and learns from others. We compare a *common (centralized) knowledge approach* where every predator contributes to a centralized rule base to an *individual knowledge approach* where every agent learns on its own and the experiences are stored locally in decentralized rule bases.

The paper is structured as follows: Section 2 summarizes some related work concerning XCS in multi-agent environments. Section 3 explains in more detail the investigated scenario. Section 4 concentrates on collaboration methods, while Sect. 5 discusses the methodology. Section 6 presents the results and comparisons, followed by a conclusion and an outlook in Sect. 7.

2 Learning Classifier Systems in MASs

The field of learning classifier systems (LCSs), introduced in the 1970ies [9], is one of the most active and best-developed forms of genetic-based machine learning. LCSs are rule-based on-line learning systems that combine nature-inspired optimization heuristics and reinforcement learning techniques to learn appropriate actions for any input they get.

A variety of different LCS implementations has been proposed, many are based on Wilson's eXtended Learning Classifier System (XCS) [10], as sketched in Fig. 1. A learning agent senses its environment and sends its detector values to an XCS. The input is compared to all rules (called *classifiers*) in the rule base (*population* P). Matching classifiers enter the *match set* M and are grouped by their actions using the *prediction array* PA, which consists of each action's average of the predicted values. Then, the action with the highest prediction value is chosen, and the related group of classifiers enters the *action set* A. The chosen action is executed and a reward value is received based on the quality of this action with respect to the resulting state of the environment. The reward is used to update the prediction values of the classifiers in the action set and a learning cycle starts again.

In general, *multi-agent learning approaches* using LCSs are based on the idea of several independent LCSs which work in parallel on the same learning problem. Agents administrate individual populations learned locally on the one hand, and contribute to global shared rule sets on the other hand. In multi-agent scenarios, this may be useful, when agents have to cooperate with each other and their local behaviors contribute to a global goal with respect to different roles. In dynamic environments, agents have to cope with changes, which often require different behaviors. This corresponds to different roles an agent can take.

In [11], an XCS approach is investigated by modeling social problems. The *El Farol bar problem* is used as a benchmark for ten up to hundred agents learning

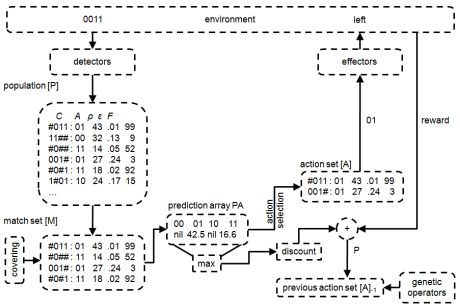

Fig. 1. Schematic overview of the on-line learning in XCS [10].

a cooperative behavior in parallel. XCS is also used in [12], where some agents in forms of various five-square tiles (called *pentominos*) collaboratively manage themselves to cover the smallest possible area by lying side by side. These papers indicate the feasibility of using XCS in multi-agent scenarios with collaborative agents, which is also in the focus of this paper.

3 Predator/Prey Pursuit Scenarios

In the literature, various types of predator/prey pursuit scenarios exist. Typically, some predators follow the goal of capturing a prey in a two-dimensional grid world [13]. Since such a two-dimensional grid world with a team of collaborating agents (i. e., predators) offers many design possibilities, this approach has been adopted for the investigations of this paper. Our special scenario is described as follows.

3.1 Grid World

In this paper, the grid world consists of a borderless two-dimensional array (also known as a torus), some predators, and one or more preys. For example, Fig. 2(a) shows a 10×10 grid world with four predators working as a team to capture one prey (a capturing situation is shown in Fig. 2(b)).

At each simulation tick, predators and prey move to one of the neighboring cells in the von Neumann neighborhood. If the cell at the desired direction is occupied, the agent stays where it is. Also, when more than one agent intends to move on a free cell, only one of them (chosen arbitrarily) will move into the cell while the other ones do not move.

The prey is captured when it has no possibility to move as all four directly neighboring cells are occupied by the predators. Therefore, the quality of the predators' moves is evaluated with respect to their ability to minimize their distance to the prey (measured as the *Manhattan distance*, i. e., the sum of horizontal and vertical distances).

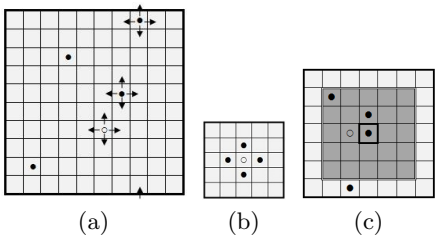

Fig. 2. (a) A borderless grid world with four predators (black dots) and a prey (white dot); (b) The goal is achieved as the prey is captured by the predators. (c) A predator's local observation range (using the Chebyshev distance of two).

3.2 Prey

Since the investigated scenario focuses on collaboratively learning predators, we start with a simple prey which ignores any sensory information except for the status of its four directly neighboring cells. In every tick it moves to one of its von Neumann neighboring cells in an arbitrarily chosen direction, unless stated differently for experimental purposes. If the prey is captured (i. e., it cannot move any more), it is eliminated and another one will appear at a random location within the grid world – to ensure that the simulation is continuously running and predators can learn in several cycles.

3.3 Predators

Every predator is designed to obtain sensory information within a limited observation range determined by a *Chebyshev distance* of two (which refers to the maximum of the horizontal and vertical distances), as depicted in Fig.2(c). There, the predator in the middle of the grid can sense itself, two of its teammates, and the prey. Here, sensing is interpreted as recognizing and knowing the grid coordinates (x, y) of all the currently sensed objects.

Moreover, the location of the prey is broadcasted to all the predators as soon as one of them has sensed it locally. This is intentionally implemented to allow for a local evaluation of the quality of the moves. In other words, without knowing the prey's location, predators cannot learn anything (since possible movements could not be rewarded in a goal-oriented way).

When the prey's location is unknown (for instance, at the beginning of the simulation), all predators move arbitrarily expecting to find it somewhere (Fig.3(a)). When at least one of them has located the prey (e. g., as depicted in Fig. 3(b)), the coordinates of the prey are broadcasted (i. e., $(5, 7)$ in Fig. 3(b), as $(0, 0)$ is the bottom-left corner cell) and retrieved by all teammates at the same tick. If the prey's location is known, the predator uses it for deciding about the next action. This decision is taken at each time step of a simulation run.

Now, conventional non-collaborative predators will individually decide to take their best movement regardless of any information about its teammates'

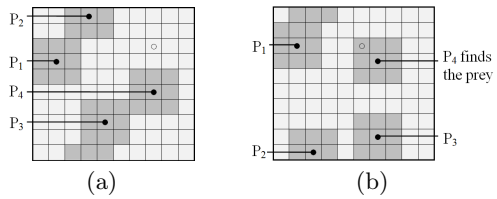

Fig. 3. Two examples of situations with predators having a viewing range of one: (a) No predator sees the prey; (b) The prey is located by predator P_4.

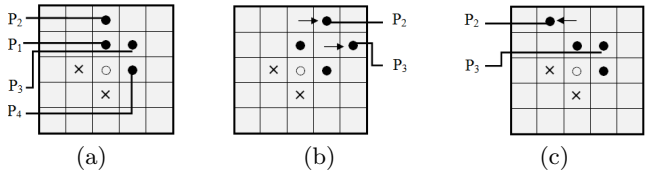

Fig. 4. (a) P_1 and P_4 are denoted as *blocking* predators, P_2 and P_3 as *blocked* predators. (b) The blocked predators cannot get closer the prey. (c) Similar to Fig. 4(b)

positions. However, it is possible that the set of a predator's movements which minimize the distance to the prey is limited, since it may be blocked by its teammates, as depicted in Fig. 4(a). Then, the desired goal of capturing the prey is not directly achievable.

As depicted in the example, the predators P_1 and P_4 perform a *selfish* behavior and block their teammates P_2 and P_3 as long as they all try to minimize their distance to the prey with each move. Consequently, if P_1 and P_4 remain at their capturing positions, P_2 and P_3 can only follow the option to move around P_1 and P_4 in order to reach the other capturing positions, as marked with crosses. Fig. 4(b) and Fig. 4(c) show the two *blocked predators* P_2 and P_3 could attempt to resolve this by moving away to the east or to the west. This provides possibilities for the blocked predators to have good moves afterwards, but the common behavior of all four predators does not relate to a desirable collaborative behavior. The following section describes ideas to pursue the team task collaboratively.

4 Collaboration Methods

In order to overcome the drawbacks of selfish behavior of non-collaborating predators, we investigate possibilities of learning collaborative behavior which is superior with respect to the global goal of capturing the prey. In a dynamically changing environment, learning is often challenged by the need to adjust the learning speed to the dynamics of the system (as mentioned in [14]). These aspects are focused on in the following.

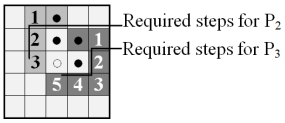

Fig. 5. Required steps to capture the prey starting from a blocking situation.

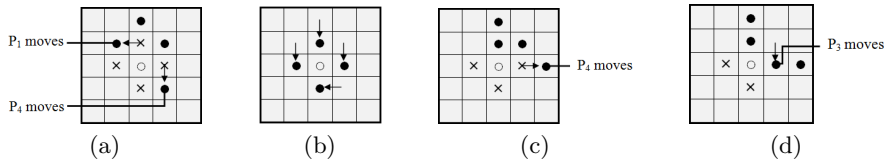

Fig. 6. (a) Fair moves by P_1 and P_4. (b) Goal achieved. (c) Not a fair move. (d) P_3 blocks P_4.

4.1 Fair Moves

In Fig. 4(a), blocking situations have been discussed which may arise in various ways. Two possible static solutions are explained here:

1. The blocked predators move step by step around the other predators to get closer to the prey eventually;
2. The predators collaborate and perform so-called *fair moves*.

As depicted in Fig. 5, the first strategy requires at least five steps for all predators to surround the prey. In contrast, using fair moves as shown in Fig. 6(a) and Fig. 6(b), only two steps are required which is a significant benefit.

The idea behind the fair moves is that blocking predators should move out of their current position to give their teammates a chance to get closer to the prey. This is called a fair move, only if the Chebyshev distance of the moving predator to the prey does not change.

Starting from the situation displayed in Fig. 4(a), an example of fair moves is shown in Fig. 6(a). The fair moves of P_1 and P_4 are allowing P_2 and P_3 to come closer to the prey, as shown in Fig. 6(b). On the other hand, Fig. 6(c) shows a move by P_4 which is not a fair one. This unsurprisingly leads to a situation where P_4 becomes a blocked predator, as depicted in Fig. 6(d). Due to the benefit of fair moves, the agents should get a special reward in the on-line learning cycles, whenever they perform a fair move.

4.2 Common Rule Base

As outlined in Sect. 2, multi-agent learning approaches may use individual rule bases or global shared rule sets. Generally, selfish agents would learn for themselves while collaborative ones do it for the team. In this paper, two different learning architectures (as depicted in Fig. 7) are compared. Figure 7(a) shows

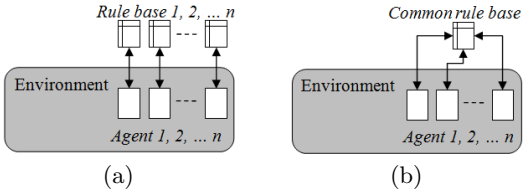

Fig. 7. Learning architectures: (a) Individual rule bases; (b) Common rule base

an architecture where every agent has its own rule base and the others do not get the benefit of learning from any of their teammates' experience. In contrast to this, the second architecture uses a common rule base for all the agents, as depicted in Fig. 7(b), i. e., every predator decides on its action by using the accumulated experience of the team. Different from centralized learning approaches with a centralized single-learning agent (e. g., in [5]); this approach is based on a rule base which cumulatively collects experience from all predators. In other words, all predators still make decisions autonomously, but store their knowledge in a centralized rule base – accessible to all teammates.

Obviously, a good move for a predator is always a good one for others being in the same situation. If predators act as a set of sensors (or experience collectors for a common rule base), they will presumably have shorter learning times than in scenarios where learning is limited to selfish agent behavior (i. e., a team of four predators can update a common rule base four times faster than a single predator can update its own rule base). Thus, in dynamically changing environments, a quicker converging process of how to behave well seems to be very desirable.

Although these architectures are independent of the specific method of updating the rule bases, in the following it is assumed that XCS is used as the learning method in both scenarios. Following, the predators' algorithm in applying the common rule base is described.

4.3 The Algorithm

At each tick of the simulation, every predator executes the algorithm given in Fig. 8, which does the following: A predator observes its environmental surrounding and takes a decision on an action that specifies the direction of the next movement. Having the prey in the local observation range, the predator broadcasts the location of the prey. Without having the prey in sight, a predator will examine whether any other teammate can sense the prey.

If the prey's location is available (locally or by broadcast), a learning mechanism is applied based on Wilson's XCS: An action is selected from the local or global XCS rule base. It is triggered and evaluated, and the reward is used to build up the predator's knowledge. Otherwise, the predator performs a random movement without referring to the rule base.

The next section describes how these behaviors are implemented, how the methods affect the results, and how useful they are in achieving the goal.

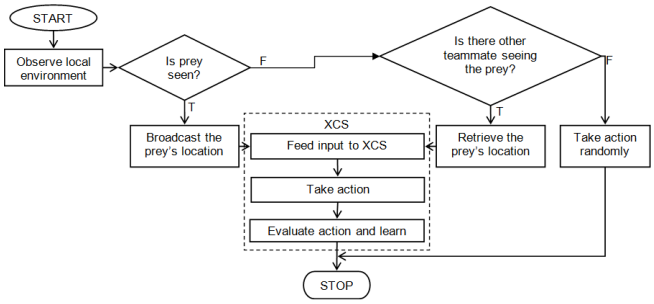

Fig. 8. A flow chart of the predator's algorithm performed at each simulation tick.

5 Methodology

In the experimental setting, the predator/prey scenario is performed in a two-dimensional 15×15 borderless grid world. Four predators having a *viewing range* limited to a Chebyshev distance of two have to learn to capture a prey. All rule bases are initialized to an empty population (i. e., no predefined knowledge). New rules are generated using the standard covering operator [15]. The maximum number of rules per rule base is set to 480. This means, whenever the number of classifiers is greater than 480, rule deletions will occur, as specified in [15].

Initially, all entities start at random coordinates. When the prey is captured, a new prey will appear and the old one disappears. The number of capture cycles is then used to compare the performances in different parameterized scenarios.

To adapt the XCS algorithm to the scenario, three things have to be defined: The input string to the XCS rule base, the action encoding, and the reward mechanism. As explained and known from the literature (e. g., in [14]), classifier systems have weaknesses in learning speed due to increasing search spaces. Therefore, an efficient way of learning favors an intelligent coding of input and output values and a proper reward mechanism.

Thus, two sorts of information are used as input to a learning predator: The current relative direction of the prey and the predator's von Neumann neighborhood of range one (denoted as *(direct) neighborhood* afterwards). The direction is used to decide where the predator should move to, while the direct neighborhood is useful for extracting the information whether a neighboring cell is occupied or not.

Figure 9(a) depicts an example derived from Fig. 2(c) and explains the encoding of the chosen XCS input. Firstly, the environment is simplified into eight directions coded into four bits sequentially representing *north*, *east*, *south*, and *west*, as shown in Fig. 9(b). For example, the direction *southeast* would be '0110'. Since our investigations are limited to a scenario with one prey, only one of the eight directions will be *true*.

The second part of the input is information concerning the possibility to move to the neighboring cells. Moving to a cell occupied by a teammate is unfavorable,

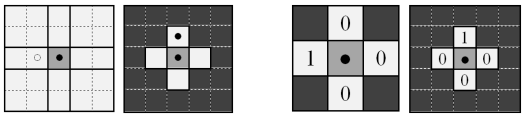

Fig. 9. (a) Simplification of the observed area; (b) Encoding of the XCS input.

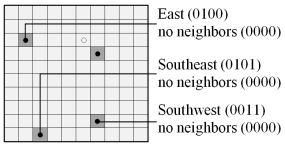

Fig. 10. XCS input of predators not seeing the prey

but in contrary, moving towards the prey is a good one. Therefore, no information about the prey's existence is given to this part of the input.

As seen in Fig. 9(b), the first part of the input is '0001' representing west. Then, a neighboring cell is coded to '1' if it is occupied by a teammate, or to '0' otherwise. Starting from the northern cell, going clockwise, and taking one bit for every direction, the neighboring cells are coded to '1000'. The XCS output will be one of four possible directions (*north*, *east*, *south*, or *west* are encoded as 0, 1, 2, or 3).

This simple form of input is also applicable to be implemented by the predators not seeing the prey. For instance in Fig. 3(a), P_1, P_2, and P_3 compose inputs using the prey's coordinate broadcasted by P_4, as shown in Fig. 10. Moreover, the encoding is able to represent fair moves. For example, if the prey is located at north ('1000') and a teammate is sensed on the southern neighboring cell ('0010'), then moving east (2) or west (4) will provide a fair move.

Based on the XCS output, each predator moves and gets a reward to the classifiers in the action set according to the mechanism shown in Table 1. The reward is based on the Manhattan distance between a predator and the prey. After moving, each predator checks its current distance to the prey (D_t) and compares it to the previous (D_{t-1}). The standard reward for an input-output-combination (a rule) is 50, and a reward is considered as low if it is less than that. A high reward will be given to a rule if it takes the predator closer to the prey. Otherwise, a low reward will be received. Stagnancy, where a predator fails to change its position, is rewarded very lowly.

A *basic reward* is given to any actions, while an *additional reward* is only given to specific movements. Fulfilling more than one criterion, a move will be awarded by the sum of the basic and additional rewards. For instance, staying at any distance of x is a *stagnant* move deserving the low reward of 1. Additional reward would be given for $x = 1$ which is a *staying as a neighbor* move, deserving the total reward of 100. This value refers to *closer*, since in the rule base both of them are represented by the same classifier denoting *if the prey is in direction z*

Table 1. Allocation of rewards

no.	D_{t-1}	D_t	name	condition	basic reward	add. reward
1.	x	$x-1$	closer	any	100	0
2.	x	$x+1$	further	any	10	0
3.	x	x	stagnant	any	1	0
4.	x	$x+1$	fair move	was a blocking predator & unchanged Chebyshev distance	0	140
5.	1	1	staying as a neighbor	moving towards the prey	0	99

and you are not blocked by your teammate, then go to z. Finally, a *fair move* is rewarded very highly to encourage predators taking it, testing its effectiveness in achieving the team task.

6 Results and Comparisons

Experiments have been done using two types of prey, a static (not moving) prey and a moving one, having the same speed as the predators. The following figures show averaged experimental results how the agents behave in simulations over time. The horizontal axis denotes the simulation time in a logarithmic scale while the vertical axis depicts the average number of capture cycles from the beginning to the end of the simulations. Data are taken from 20 experiments where each simulation ends after one million ticks – one tick is one simulation step. Due to lack of space we did not include any information on the statistical significance of the results. But a simple statistical analysis indicates insignificant deviations.

Figures 11(a) and 11(b) present the experimental results of the comparison between simulations with and without rewarding fair move decisions in the case of both learning architectures (individual vs. common rule base). Moreover, both figures show some relatively significant increases for the number of average capture times by rewarding fair move decisions, either in capturing a static or a moving prey. Since rewarding fair moves seems advantageous, further comparisons only focus on the two different rule base architectures which are always rewarding fair decisions.

The learning speed, which has been pointed out as a weakness of XCS, is improved slightly by implementing a common rule base. As shown in Fig. 12, simulations using the common rule base approach are superior for some initial period although after some time the individual rule base approach provides better results, especially in capturing a moving prey.

Furthermore, the individual learning approach can benefit from storing individual knowledge for a longer period. Rarely used classifiers are possibly deleted in the common learning approach, since the maximal number of classifiers keeps the population as compact as possible (cf. the mechanism of deleting classifiers, as proposed by [15]).

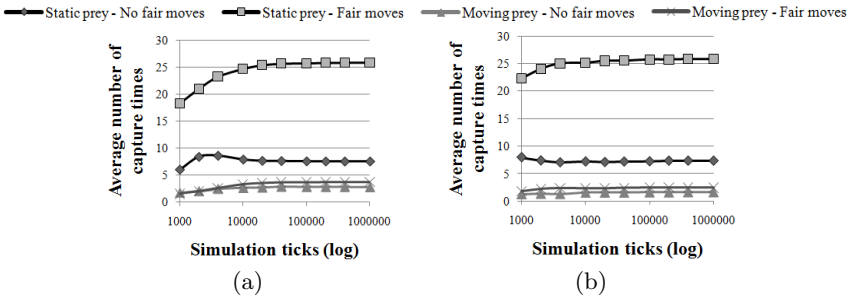

Fig. 11. Comparisons of learning with and without rewarding fair move decisions using: (a) the individual rule base approach; and (b) a common rule base

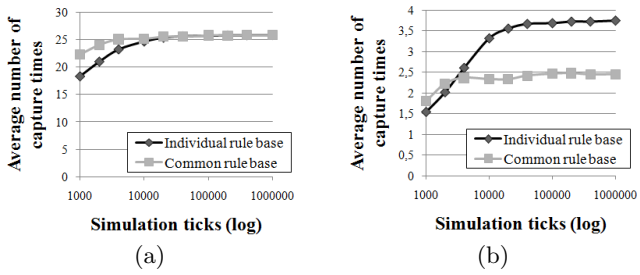

Fig. 12. Comparison of learning using individual and common rule bases in fair mode simulations using (a) a static prey; and (b) a moving prey

7 Conclusion and Outlook

This paper has focused on an instance of the generic pursuit scenario where predators should learn to contribute to a common goal - capturing a prey. The usability of Wilson's XCS has specially been investigated in two different approaches. Firstly, all predators individually learn and store their experience in local rule bases. Secondly, the predators share and store their experiences in a common rule base.

Predators have been designed with a local view where they can sense their local environment. If the prey is found, its coordinates will be broadcasted to all other predators. Then, a simple input encoding has been defined, consisting of the direction where the prey has been located and information about the neighboring cells - are the cells occupied with the teammates or not. Finally, a proper reward function for fair moves has been proposed, which enforces collaborative group behavior. These fair moves are based on the idea that moving away from a desirable position and thus giving a chance for a teammate to come closer could be beneficial in some cases.

Experimental results have been achieved using different parameter combinations. The results provide a clear view: The learning approach using a common rule base provides a quicker improvement in the learning behavior but may be

significantly inferior on the long run, in particular in situations with a *moving prey*. Nevertheless, the presented idea of collaborative learning by storing the knowledge in a common rule base provides a wide area for further research on multi-agent learning: The complexities of heterogeneous predators, an intelligently acting prey, or more complex goals than capturing a prey give rise to new challenges for learning, which will be tackled by future work.

References

1. Sen, S., Weiß, G.: Learning in multi-agent systems. In: Multi-agent systems: A modern approach to distributed articial intelligence. MIT Press (1999) 259–298
2. Parunak, H.V.D.: Industrial and practical applications of distributed artificial intelligence. In: Multi-agent systems: A modern approach to distributed articial intelligence. MIT Press (1999) 377–421
3. Eßer, A., Franke, M., Kamper, A., Möst, D.: Impacts of consumer response and dynamic retail prices on electricity markets. Future Power Markets **5** (2007) 335–341
4. Dorigo, M., Gambardella, L.M.: Ant colonies for the traveling salesman problem. BioSystems **41** (1996) 73–81
5. Huberman, B.A., Clearwater, S.: A multi-agent system for controlling building environments. In: Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS 1995), MIT Press (1995) 171–176
6. Wooldridge, M.J.: An introduction to multi-agent systems. John Wiley & Sons (2002)
7. Benda, M., Jagannathan, V., Dodhiawala, R.: An optimal cooperation of knowledge sources: An empirical investigation. Technical Report BCS-G2010–28, Boeing Advanced Technology Center, Boeing Computing Services, Seattle, United States of America (1986)
8. Lenzitti, B., Tegolo, D., Valenti, C.: Prey-predator strategies in a multi-agent system. In: Proceedings of the 7th International Workshop on Computer Architecture for Machine Perception (CAMP 2005), IEEE Computer Society (2005)
9. Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press (1975)
10. Wilson, S.W.: Generalization in the XCS classifier system. Genetic Programming 1998: Proceedings of the Third Annual Conference (1998) 665–674
11. Hercog, L.M., Fogarty, T.C.: Social simulation using a multi-agent model based on classifier systems: The emergence of vacillating behaviour in "el farol" bar problem. In: Proceedings of the 4th International Workshop on Learning Classifier Systems (IWLCS 2001). Volume 2321 of LNAI., Springer (2002) 88–111
12. Takadama, K., Terano, T., Shimohara, K.: Learning classifier systems meet multi-agent environments. In: Advances in Learning Classifier Systems. Volume 1996 of LNAI. Springer (2001) 192–212
13. Stone, P., Veloso, M.: Multi-agent systems: A survey from a machine learning perspective. Autonomous Robots **8**(3) (2000) 345–383
14. Richter, U., Prothmann, H., Schmeck, H.: Improving XCS performance by distribution. In: Proceedings of the 7th International Conference on Simulated Evolution And Learning (SEAL 2008). Volume 5361 of LNCS., Springer (2008) 111–120
15. Butz, M.V.: XCSJava 1.0: An implementation of the XCS classifier system in Java. Technical Report 2000027, Illinois Genetic Algorithms Laboratory, Urbana, United States of America (2000)