Cooperation Strategies for Pursuit Games: from a Greedy to an Evolutive Approach

Juan Reverte, Francisco Gallego, Rosana Satorre, Faraón Llorens
{jreverte, fgallego, rosana, faraon}@dccia.ua.es

Department of Computer Science and Artificial Intelligence
University of Alicante

Abstract. Developing coodination among groups of agents is a big challenge in multi-agent systems. An appropriate enviroment to test new solutions is the prey-predator pursuit problem. As it is stated many times in literature, algorithms and conclusions obtained in this environment can be extended and applied to many particular problems. The first solutions for this problem proposed greedy algorithms that seemed to do the job. However, when concurrency is added to the environment it is clear that inter-agent communication and coordination is essential to achieve good results.
This paper proposes two new ways to achieve agent coodination. It starts extending a well-known greedy strategy to get the best of a greedy approach. Next, a simple coodination protocol for prey-sight notice is developed. Finally, under the need of better coordination, a Neuroevolution approach is used to improve the solution. With these solutions developed, experiments are carried out and performance measures are compared. Results show that each new step represents an improvement with respect to the previous one. In conclusion, we consider this approach to be a very promising one, with still room for discussion and more improvements.

Keywords: Multi-agent systems, Communication, Coordination, Neuroevolution

1 Introduction

The Predator-prey problem (or pursuit domain) is an appropriate testbed for multi-agents systems [1]. It consists of world where a group of agents (called predators) aims to chase and surround another agent (called prey) that tries to evade them [2]. The goal of predator agents is to surround (capture) prey without touching it (i.e. occupying the adjacent cells), while the goal of the prey, as expected, is not to be captured.

This problem has been adressed many times in literature. Initially, Korf [3] proposed a greedy algorithm without inter-agent communication. His approach was to use a fitness function that combined 2 forces: each predator was "attracted" by the prey and "repelled" from the closest other predator. This solution kept predators away from other predators while they got closer to the prey.

The idea was to chase the prey arranging predators in an stretching circle. Korf concluded that the pursuit domain was easily solved with local greedy heuristics.

A great number of alternatives have emerged since Korf's. Haynes [4] used genetic programming to evolve coordinated predators. Haynes compared differences between communicating and non-communicating predators with respect to their success in capturing the prey. He also co-evolved predators and the prey and found that a prey following a straight, diagonal line in an infinite world was never captured unless it was slower than its pursuers. This demonstrated that for certain instantiations of the domain, Korf's heuristic was not enough. Tan [5] used Reinforcement Learning to improve cooperation in three ways: (1) sharing instantaneous information (sensation, action, rewards obtained), (2) sharing episodes of instantaneous information, and (3) sharing learnt policies. Tan showed that agents learn faster when they learn cooperatively than when they learn individually. Later, Jim and Giles [6] proposed a genetic algorithm and multi-agent communication through a blackboard. A really interesting alternative was proposed by Katayama et al. [7]. They integrated Analytic Hierarchy Process (AHP) into a Profit-Sharing algorithm. They gave primary knowledge to agents when they started their learning processes and proposed a way to progressively stop providing "hints" to agents as they grow.

Despite the great number of proposed solutions, there is still room for improvements in different instantiations of the pursuit domain. As Tan stated in his work [5], coordination algorithms or protocols tested under the pursuit domain may easily be ported to other autonomous agents domains in general. This paper presents a new proposal for improving cooperation between predators in the pursuit domain. The idea presented here is to mix the efficiency of greedy approaches with two coordination proposals: a simple sight notice protocol and an evolutionary coordination system based on Neuroevolution [8]. Results show that this is a promising approach that develops a very efficient coordination mechanism, with still room for more improvements.

This paper is organized as follows. Section 2 explains the characteristics of the pursuit domain. Section 3 presents the extension to Korf's greedy approach and the two new inter-agent coodination approaches. In section 4, results of the experiments carried out are analyzed. Finally, conclusions and future lines to follow are given in section 5.

2 The Pursuit Domain

Stone and Veloso [1] considered the pursuit domain as an interesting start point because it is easy to understand and difficult to master. Moreover, it is still popular because it is possible to create many different instances with different sorts of handicaps. The most classical environment consisted of a finite, discrete, grid world where 4 predators tried to capture 1 prey. In this environment, agents moved sequentially and two agents were not allowed to be on the same cell.

In order to realize our experiments we used Kok and Vlassis' Pursuit Domain Package (PDP)[9]. PDP is a software package that simulates a pursuit domain

environment and allows to modify its parameters to instantiate different experimental scenarios. Concretelly, PDP was tuned for the purposes of our research to show off these characteristics: (1) toroidal world with a discrete, orthogonal grid of squared cells, (2) availability for the agents to move to every adjacent cell (9 possible options), (3) concurrency in the execution and movement of predators, (4) limited field of vision (FOV) for agents, (5) agent's capability to communicate with others inside FOV, (6) capture method: 4 predators occupying the 4 orthogonally adjacent cells (i.e. north, south, east and west).

Defined this way, PDP has some challenges to face. The three most remarkables ones are: (1) Concurrency lets predators move to the same cell in the same timestep (i.e. they collide). If this occurs, colliding predators are penalized by replacing them randomly. (2) FOV makes exploration necessary, and (3) the toroidal world removes the possibility of cornering the prey.

3 Methodology

Initially, Korf [3] considered a solution quite simple yet effective. The approach was to consider an "attractive" force that pushed predators towards the prey. The method was to calculate this force as fitness function for each of the possible cells to go next, and finally select the more attractive one. This solution had the problem that predators piled up and disturbed themselves; then, it turned difficult to achieve the final surrounding capture position. Korf overcomed this problem considering a "repulsive" force which pushed each predator away from the nearest other predator. With this new force, predators attacked the prey more jointly, not piling themselves up.

The reduced number of cycles that predators took to capture the prey with Korf's method seemed good enough not to consider the necessity of improving it. However, the differences between the environment used by Korf and new environments like PDP [9] lead to reconsider it. For instance, Korf reported that his algorithm captured the prey in 119 cycles in average. The experiments we have run in the most similar conditions possible to Korf's inside PDP took 366 cycles in average. In this case, which is the best one for Korf, the toroidal world and the collisions between agents multiply time to capture the prey by 3. When conditions get worse, namely when the FOV of predators is reduced, the performance of Korf's approach deteriorates exponentially.

This means that it is necessary to extend Korf's algorithm to deal with the new issues of the environment. One possible way to extend it is to reconsider the way Korf treated attractive and repulsive forces between agents. In his proposal, predators where attracted by the prey and repelled by the nearest other predator. This leads to situations where one predator may be repelled directly against other predator, resulting in a collision. Then, the first approach to take is to make predators repel from all other predators. Equation (1) shows the fitness function used to do this. This function depends on the (x, y) coordinates of the cell and calculates distances from that cell to prey location (X_p, Y_p) and to other n predators locations (X_i, Y_i) using Manhattan Distance $d(x, y, x', y')$. To

balance the relative amount of repulsive forces against the attractive one, a scale constant $k \in [0, 1]$ is added. We will call Extended Korf (ExtKorf, for short) to the algorithm which works like Korf's but using the fitness function from Eq.(1).

$$f(x, y) = d(x, y, X_p, Y_p) - k \sum_{i=1}^{n} d(x, y, X_i, Y_i) \tag{1}$$

The Extended Korf algorithm dramatically outperforms results of the Korf algorithm. The main reason for this is that it reduces collisions between predators by an order of magnitude, thus avoiding penalties. Results supporting this are explained in Sect. 4.

3.1 Cascading Sight Notice (CSN)

In the environment where Korf did his experiments, communication between agents were not necessary, as he demonstrated. The main reason was that his agents were able to see the whole world at once. But, the more we limit the FOV of the predators the more they need to get more information to efficiently locate the prey. When predators have a reduced FOV, most of the time it happens that when some predators have already found the prey, others are still wandering around. This delay in founding the prey could be avoided if the predators were able to effectively tell where was the prey to others when they had found it.

Consider that an agent is located at (x, y) and having a FOV of f cells means that the agent is only able to perceive what happens in the set of cells $C = \{(x', y') \mid x - f \leq x' \leq x + f, y - f \leq y' \leq y + f\}$. Take into account that this refers to sensing in general, and not seeing in particular. Therefore, an agent is only able to communicate with other agents being inside its FOV. Moreover, agents never know their global location, nor global coordinates of other agents. They are only aware of the relative location of other agents with respect to them.

In strict sense, the probability of a predator indefinitely not finding the prey in these conditions is not 0, and that is definitely a problem to overcome. But communication is not as simple as telling others directly where is the prey; there is no way to do that. In order to communicate where the prey is, we propose a simple protocol called Cascading Sight Notice (CSN). The idea is that a predator P^0 seeing the prey Y has to communicate the relative location of Y that P^0 is perceiving to each other predator P^i that P^0 can see (i.e. P^i is inside the FOV of P^0). Then, each P^i not seeing Y could locate it by listening to P^0. P^i will then be aware of the relative location of P^0 with respect to P^i and also aware of the relative location of Y with respect to P^0. So, P^i will be able to calculate the relative location of Y with respect to P^i by adding the vectors of the two relative locations it will know. Then, when P^i will have located the prey, P^i will resend this new relative location to other predators in its FOV. The cycle will go on until no predator will be hearing or hearing predators will have already known where prey is (see Fig. 1).

This simple protocol lets predators with reduced FOV find the prey earlier than predators without communication do, and this turns into an improvement in

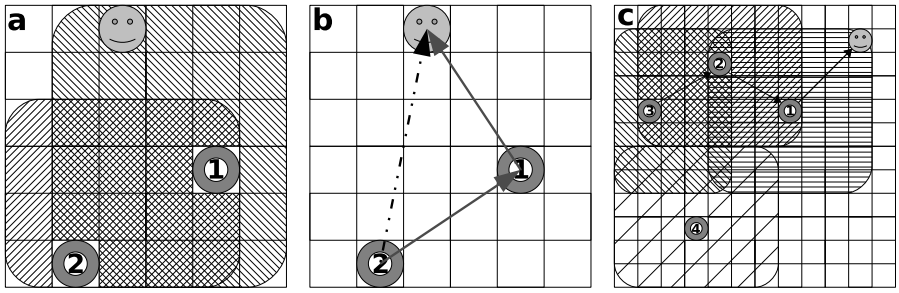

Fig. 1. a) Two predators with FOV 3 seeing each other, predator 1 seeing the prey. b) Predator 2 can figure out prey location from message of predator 1. c) Predators 1, 2 and 3 can figure out prey location, predator 4 cannot

the average number of cycles needed to capture the prey. The results supporting this are shown and explained in Sect. 4 (see Fig.3).

3.2 NEAT Coordination Protocol (NECool)

Inside PDP, collisions occur when two or more predators move to the same cell on the same timestep. As long as predators decide where to move in a concurrent fashion, they have no opportunity of avoiding collisions unless they establish an appropriate coordination protocol. To create a fast and nearly-optimal coordination protocol we propose to evolve a neural network that decides the next movement to do for each predator. The neural network of predator P^i will receive as input the location of each predator P^j inside the FOV of P^i (including P^i), and a prediction of the next possible movements of each P^j. With this information, the neural network will be able to figure out the next movement that P^i should do to optimize the capture of the prey not colliding with any P^j.

The most relevant information each neural network will receive will be that regarding movement predictions of other predators. Each predator P^i has an associated 3×3 fitness matrix $F^i_{3\times3}$ where each element $F^i_{jk} \in [0,1]$ represents the fitness associated to P^i moving to that cell, assuming the predator is currently located in F^i_{22}. The values of F^i are calculated using Extended Korf's algorithm, having previously used CSN to locate the prey.

These movement predictions represent 9 values for each predator, 36 for the 4 predators. Trying to pass these values directly to a neural network would require 36 input neurons, which means a great search space. Moreover, this approach is not scalable, because increasing the number of predators requires a linear increase in the number of input neurons. As long as a predator P^i is not interested in the exact F^j_{kl} of other predators P^j, but in the total other relation $F^j_{kl} < F^j_{mn}$, an approach to reduce dimensionality can be used (see Fig. 2). A consecutive number from 1 to 9 is assigned to each cell of F^j. Then a vector with

the 9 cells ordered by F^j_{kl} is formed. From the numbers assigned to the ordered cells in the vector, a 9-digit number $\lambda \in [123456789, 987654321]$ is formed. Then α is calculated as λ scaled into $[-1, 1]$. α is known as the Compressed Fitness Value, and it is sent to other predators as movement prediction. Then, each P^i feeds its neural network with the αs received from each P^j inside the FOV of P^i.

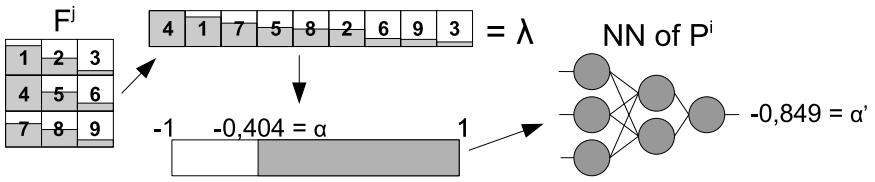

Fig. 2. Using Compressed Fitness Values to feed the neural networks

To make this algorithm even more scalable, the neural network of a predator P^i has been designed to be recurrent with a fixed number of input neurons. Therefore, each neural network has only 3 input neurons (X^j, Y^j, F^j): X^j and Y^j represent the coordinates and F^j the Compressed Fitness Value for predator P^j. The neural network is expected to sequentially receive a triplet of inputs from each P^j inside the FOV of P^i, and to activate all its neurons once. Finally, the neural network receives the 3 inputs related to P^i, activating and flushing the net to get a final output value α'. α' is treated as a Compressed Fitness Value and an $\lambda' \in [123456789, 987654321]$ is obtained rescaling α'. The first digit of λ' is identified with the cell where the predator should move next. All the steps described up to now belong to the algorithm 1.

In the algorithm 1 the function $compressAllFitnesses(F^i)$ takes the matrix F^i with the 9 fitnesses of the adjacent cells where P^0 could move next, and transforms it into $\alpha \in [-1, 1]$. Then, α is sent to other predators which use it as input to their neural networks. Each other predator P^i uses its neural netwotk to calculate α', which is passed to the inverse function of $compressAllFitnesses$, that is $getNextCellToMove$. $getNextCellToMove(\alpha')$ "decompresses" α', obtaining λ' and returns the first digit of λ', which identifies the best fitted movement to do next.

As long as the neural networks of predators need to be recurrent, an algorithm capable of training this kind of networks is needed. For that, we propose to use Neuroevolution of Augmenting Topologies (NEAT [8]). In our work we have set up populations of predators and tested their performance against a random prey and an evading prey. Predators were rewarded for early catching the prey and punished for each collision between them. Finally, we got the best neural network from the last population and used it within our predators to test if it reduced the collisions and improved performance. Results of this experiment are shown in Sect. 4.

Algorithm 1 Coordinate decisions of predator P^0 with each P^i decision

Require: P^0 = predator with a 3-to-1 recurrent neural network
Require: P = a set of predators inside FOV of P^0
Let F^0 be a $3x3$ real matrix
Let $\overrightarrow{C}$ be a vector of real numbers
Let α, α' be real numbers
for all $((x, y) \mid 1 \leq x \leq 3, 1 \leq y \leq 3)$ **do**
    $F^0_{xy} \Leftarrow calculateExtendedKorfFitness(x, y)$
end for
$\alpha \Leftarrow compressAllFitnesses(F^0)$
$sendCompressedFitnessToOtherPredators(\alpha, P)$
$\overrightarrow{C} \Leftarrow receiveCompressedFitnessesOfPredators(P)$
for all $(P^i \in \{P \setminus P^0\})$ **do**
    $\alpha \Leftarrow getCompressedFitnessOf(P^i, \overrightarrow{C})$
    $(x, y) \Leftarrow getLocationOfPredator(P^i)$
    $activateNeuralNetworkWithValues(x, y, \alpha)$
end for
$\alpha \Leftarrow getCompressedFitnessOf(P^0, \overrightarrow{C})$
$(x, y) \Leftarrow getLocationOfPredator(P^0)$
$\alpha' \Leftarrow flushNeuralNetworkWithValues(x, y, \alpha)$
$c \Leftarrow getNextCellToMoveTo(\alpha')$
$movePredatorTo(P^0, c)$

4 Results

To validate our approach we compared the results of the 3 methods (ExtKorf, ExtKorf+CSN and ExtKorf+CSN+NECool) between them and against the original of Korf in the same environment conditions, but variating the FOV. We measured predators against two different preys: a random moving prey, and a evading prey. The second prey moved to the adjacent cell that is more distant from the closest predator. These tests let us show the magnitude of the improvements and their relative relevance. For running the simulations we used Kok and Vlassis' Pursuit Domain Package (PDP) [9]. Concretely, we used a 30×30 cells field, allowing agents to move diagonal, with the prey starting on the center and predators starting randomly placed. We lauched 4 predators and 1 prey, and to capture the prey predators needed to be surrounding the prey orthogonally, without touching it. In case of collision, predators colliding were penalized being replaced randomly. Simulations were always ran for 500 consecutive episodes.

Our first experiment measured relative improvement of the Extended Korf algorithm against original Korf's. Results shown an improvement of an order of magnitude in most cases. The improvement was much greater when the FOV was unlimited. Namely, for the maximum FOV (15 cells), relative improvement of ExtKorf against Korf was of $\frac{50}{490} \rightarrow 90\%$, while for the minimum FOV (3 cells) it was $\frac{1049}{3007} \rightarrow 65\%$. It is normal, though, that the maximum improvement in cycles and collisions happened when agents could sense the whole world at once. In this case, predators did not lose cycles in trying to find the prey and

always knew where other predators were. As long as this was the most obvious improvement and for space reasons, we deliverately leave out the graph to center on the other improvements and analyze them more deeply.

As we stated in previous section, there are two major ways of improvement: (1) more efficiently finding the prey and (2) avoiding collisions. We have made two proposals to cover each of these two ways. The CSN protocol enables predators to locate the prey earlier by using the indications from other predators. To check the relative improvement of CSN, we have measured Extended Korf's performance with and without CSN. Figure 3 shows results of this comparison. As expected, results suggest that there are plenty of situations in which CSN saves cycles of exploration to the predators. On average, CSN reduces time spent by predators in finding the prey, and it is more significant when FOV is minimal.

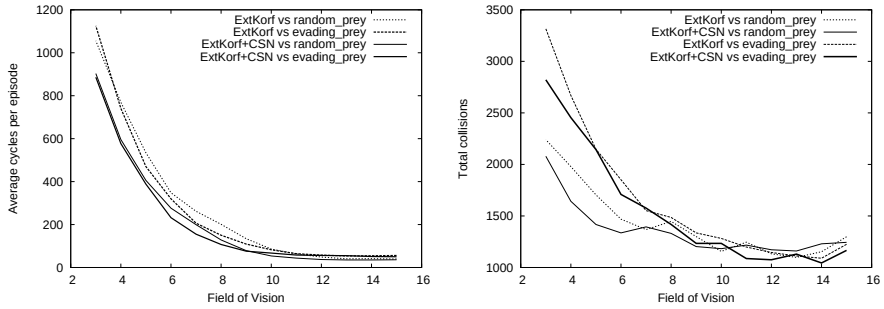

Fig. 3. Comparison between Extended Korf's model and Extended Korf's model with CSN with respect to average cycles per episode and total collisions in 500 episodes

However, CSN is not a definitive solution. CSN turns less effective when dimensions of the world increase due to the necessity of predators to be inside FOV of others to communicate with them. This limits the relative improvement that could be achieved with CSN to a factor depending on the relation of FOV with the size of the world. The less proportion of cells a predator is able to perceive, the more difficult to communicate with others and the more difficult to find the prey.

Altough these results suggest that CSN could be improved, it is not an easy task because FOV restricts communication between agents. Therefore, other way to globally improve performance is to reduce collissions between predators. NECool addressed this issue. To test NECool we set up a training session of 250 generations, with a population of 100 predators. Each predator was tested by 50 episodes against each type of prey, with 6000 cycles as maximum episode time to capture it. The fitness function used to train predator was $f(n, c) = \frac{6001}{n+10c+1}$, which depends on the average number of cycles to capture the prey (n) and the average number of collisions (c). All agents had 6 cells as FOV.

Once we had trained NECool predators, we ran for them the same 500 episodes test we had run earlier, but this time against ExtKorf+CSN predators. The result (see Fig. 4) was a dramatic reduction in the number of total collisions, and this reflected directly in an improvement in the average number of cycles to capture the prey, by around $25-35\%$. It is interesting to notice that predators were trained with a FOV of 6 cells but tested with different FOVs. To do this, the information passed to the neural network about (x, y) coordinates of each predator inside FOV of P^0 was always normalized and used coordinates of P^0 as origin. This way, predator coordinates were always independent and scalable to the real FOV in use. If agents had a FOV of f cells, and a predator P^i was located in coodinates (x, y) relative to P^0, the information passed to the neural network of P^0 about the coordinates of P^i was $(\frac{x}{f}, \frac{y}{f})$. Therefore, predators always used coordinates in range $[-1, 1]$, whichever the FOV was.

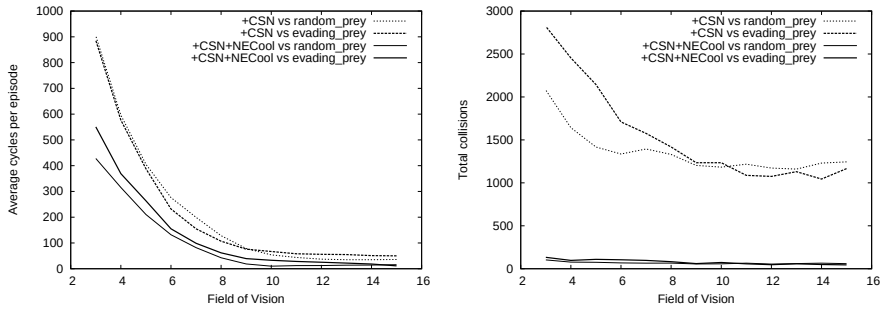

Fig. 4. Comparison between Extended Korf's model with CSN and with CSN+NECool with respect to average cycles per episode and total collisions in 500 episodes

5 Conclusions and Futher Work

Many authors claim that there are interesting challenges to face in the pursuit domain environment. This paper considers coordination problems that arise in some interesting instances of this environment. It proposes a way to mix the efficiency of greedy approaches with the power of evolutive algorithms to get the best of both. The proposal starts extending the approach of Korf (ExtKorf) and continues adding two cooperative strategies: Cascading Sight Notice (CSN) and NEAT Coordination Protocol (NECool).

In order to validate the proposal, the algorithms ExtKorf, ExtKorf + CSN and ExtKorf+CSN+NECool have been measured under the same environment conditions and compared pairways and against Korf's. The Pursuit Domain Package has been used to simulate the environment with the desired challenging conditions. A first experiment was ran to compare measured performance in cycles and collisions between Korf and Extended Korf algorithms. Results of this

experiment showed an improvement of an order of magnitude in most cases. A second experiment demonstrated that adding CSN to ExtKorf reduces the average number of cycles to capture the prey, but the reduction is only significant under some conditions. This was mainly due to the necessity of predators to be inside FOV of others to communicate with them. A third experiment added NECool and compared it with ExtKorf+CSN. Results showed a dramatic reduction of total collisions between predators, which mapped directly to a significant improvement (25 to 35%) in average number of cycles to capture the prey.

In consequence of what experiments have shown, we conclude that mixing greedy and evolutive approaches is a promising path to explore. Our final algorithm, ExtKorf+CSN+NECool achieved great results mainly due to its ability to make predators collaborate in an efficient way to lower down collisions with minimum impact in the greedy way to chase the prey. However, there remains room for improvements. For instance, it is still needed a way to early find the prey, what could be achieved if predators coordinate to explore the world, rather than exploring it randomly. Our future work will address this issue and it will also focus on lowering down collisions to 0, with the minimum impact on chasing efficiency.

References

1. Stone, P., Veloso, M.M.: Multiagent systems: A survey from a machine learning perspective. Autonomous Robots **8**(3) (2000) 345–383
2. Benda, M., Jagannathan, V., Dodhiawalla, R.: On optimal cooperation of knowledge sources. Technical Report Tech. Rep. BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue, WA (1986)
3. Korf, R.E.: A simple solution to pursuit games. In: Proceedings of the 11th International Workshop on Distributed Artificial Intelligence, Glen Arbor, MI. (1992) 183–194
4. Haynes, T., Sen, S.: Evolving behavioral strategies in predators and prey. In Sen, S., ed.: IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems, Montreal, Quebec, Canada, Morgan Kaufmann (1995) 32–37
5. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative learning. In Huhns, M.N., Singh, M.P., eds.: Readings in Agents. Morgan Kaufmann, San Francisco, CA, USA (1997) 487–494
6. Jim, K.C., Giles, C.L.: Talking helps: Evolving communicating agents for the predator-prey pursuit problem. Artificial Life **6**(3) (2000) 237–254
7. Katayama, K., Koshiishi, T., Narihisa, H.: Reinforcement learning agents with primary knowledge designed by analytic hierarchy process. In: SAC '05: Proceedings of the 2005 ACM symposium on Applied computing, New York, NY, USA, ACM (2005) 14–21
8. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10**(2) (2002) 99–127
9. Kok, J.R., Vlassis, N.: The pursuit domain package. Technical Report Technical Report IAS-UVA-03-03, Informatics Institute, University of Amsterdam, The Netherlands (2003)