Mixing Greedy and Evolutive Approaches to Improve Pursuit Strategies

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. The prey-predator pursuit problem is a generic multi-agent testbed referenced many times in literature. Algorithms and conclusions obtained in this domain can be extended and applied to many particular problems. In first place, greedy algorithms seem to do the job. But when concurrence problems arise, agent communication and coordination is needed to get a reasonable solution. It is quite popular to face these issues directly with non-supervised learning algorithms to train prey and predators. However, results got by most of these approaches still leave a great margin of improvement which should be exploited.
In this paper we propose to start from a greedy strategy and extend and improve it by adding communication and machine learning. In this proposal, predator agents get a previous movement decision by using a greedy approach. Then, they focus on learning how to coordinate their own pre-decisions with the ones taken by other surrounding agents. Finally, they get a final decission trying to optimize their chase of the prey without colliding between them. For the learning step, a neuroevolution approach is used. The final results show improvements and leave room for open discussion.
Keywords: Multi-agent systems, communication, coordination, neuroevolution

1 Introduction

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

This problem has been adressed many times in literature. Initially, Korf [7] proposed a greedy 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 keeped 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 and Sen [3] 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. Next, Chainbi et al. [2] used petri nets to coordinate predators while solving concurrency problems between them. Tan [10] 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 [4] proposed a genetic algorithm and multi-agent communication through a blackboard. A really interesting alternative was proposed by Katayama et al. [5]. They developed a way to integrate Analytic Hierarchy Process (AHP) into a Profit-Sharing algorithm. They gave primary knowledge to agents when they start their learning process. As they say, it does not seem reasonable to continue giving "hints" to grown agents that have developed their own knowledge, so Katayama et al. proposed a way to progresively stop providing "hints" to agents.

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 [10], 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.

2 The Pursuit Domain

Stone&Veloso [9] considered the pursuit domain to be a toy problem with respect to multi-agent systems. However, it is an interesting start point because it is easy to understand, easy to play around with and difficult to master. Moreover, it is still popular because it is possible to create many different instances with different types of handycaps. The most classical environment consisted of a finite, discrete, grid world where 4 predators tried to capture 1 prey and agents were only allowed to move to orthogonally adjacent cells (i.e. north, south, east or west). In this environment, agents moved sequentially and two agents were not allowed to be on the same cell.

As Stone&Veloso stated [9], that classical environment could be varied by changing the size and shape of the world, the legal moves, the concurrency on

agent movements, the size of agent's field of vision (FOV) in cells, the presence of objects, the definition of capture, the behaviour of the prey/s and the way predators communicate or not. Among this characteristics, we find three of them to be key for the environment to be challenging enough: using a toroidal world, restricting perception of agents to their FOV and making predators move concurrently.

Proposed characteristics could be simulated inside Kok&Vlassis' Pursuit Domain Package (PDP)[6]. PDP is a software package that simulates a pursuit domain environment. It lets modifying the parameters previously mentioned to instantiate different experimental scenarios. Concretelly, PDP was tuned for the purposes of our research to reflect some exact characteristics that are described as follows: (1) Toroidal world with a discrete, orthogonal grid of squared cells, (2) Availability for the agents to move to every adjacent cell each turn (9 possible options), (3) Concurrency in the execution and movement of predators, (4) Limited FOV for agents in the world affecting all agent sensors, (5) Agent's capability to communicate with other agents inside FOV, (6) Programability of prey behahiour, (7) Selection of the capture method; in our case, 4 predators occuping the 4 orthogonally adjacent cells (i.e. north, sourth, east and west).

Defined this way, PDP has some challenges to face. As we stated before, the three most remarkables ones are: (1) Concurrency lets predator 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 [7] considered a solution quite simple yet effective. The approach was to consider an "attractive" force which 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 most 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 [6] 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 take 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, as it is shown in the left graph of figure 2.

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 atractive and repulsive forces between agents. In his proposal, predators were atracted 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 atractive one, a scale constant k is added. We will call Extended Korf, or ExtKorf for short, to the algorithm which works like Korf's but with the fitness function shown by equation 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 shown and explained in section 4 (see figure 2).

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 capture the prey. When predators have a reduced FOV, most of the times happens that when some predators have 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 the prey is to others when they had found it.

Consider that an agent located at x, y and having a FOV of n cells means that the agent is only able to perceive what happens in the cells $\{(x', y')/x - n <= x' <= x + n, y - n <= y' <= y + n\}$. 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 other agents are with respect to them.

In strict sense, the probability of a predator indefinitely not finding the prey in this conditions is not 0, and that is definitely a problem to overcome. But communication is not as simple as telling others directly where the prey is; 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 seeing the prey Y has to communicate the relative location of Y that P is perceiving to each other predator P^i that P can see (i.e. P^i is inside the FOV of P). Then, each P^i not seeing Y could locate it by listening to P. P^i will then be aware of the relative location of P with respect to P^i and also aware of the relative location of Y with respect to P. So, P^i is able to calculate the relative location of Y with respect to P^i by adding the vectors of the two relative locations it knows. Then, when P^i has located the prey, P^i resends this new relative location to other predators in its FOV. The cycle continues until no predator is hearing or hearing predators already know where prey is (see figure 1).

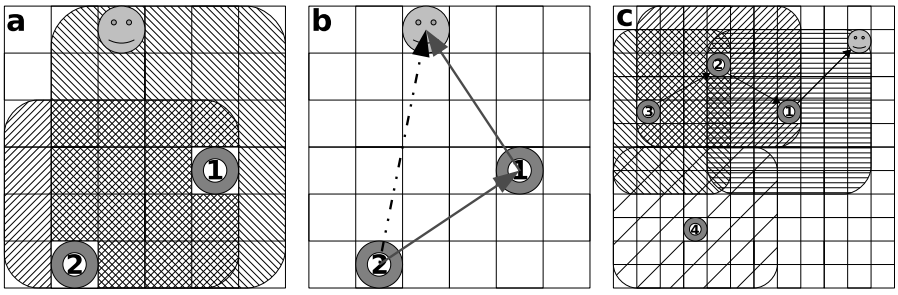

Fig. 1. a) Two predators with FOV 3 seeing each other, predator 1 seeing the prey. b) Predator 2 can figure out prey location from message of predator 1. c) Predators 1, 2 and 3 can figure out prey location, predator 4 cannot

This simple protocol lets predators with reduced FOV find the prey earlier than predators without communication do, and this turns into an improvement in the average number of cycles needed to capture the prey. The results supporting this are shown and explained in section 4 (see figure 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 appropiate coordination protocol. One possibility to look for an optimal coordination protocol is to evolve a neural network that decides the next movement to do, taking into account the fitness values of the 9 possible cells to go next. If the neural network of predator P receives as input the location of all the other predators P^i that are inside the FOV of P, then the neural network will be able to output the next move that P should do to optimize the capture of the prey without colliding with any P^i.

Following this idea, algorithm 1 lets predators learn how to optimally coordinate and improve their performance in capturing the prey. To make algorithm 1 understandable, it is necessary to clarify some insiders. First of all, the function $compressAllFitnesses(D)$ takes as argument a matrix with the 9 fitnesses associated to each one of the 9 adjacent cells where P^0 could move next. This is a biyective function that transforms the matrix D into a real value in $[-1, 1]$. This value is sent to other predators which use it as input to their neural networks. The neural networks, having locations and compressed fitnesses of predators as inputs, output a real value in $[-1, 1]$. This value is passed to the inverse function of $compressAllFitnesses$, that is $getNextCellToMove(c)$. This last function treats the output from the neural network as a new compressed fitness: it "decompresses" the value, reconstructing a 3x3 matrix of fitnesses, and then it returns the number of the cell with best fitness value.

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
Let P be a vector of predators
Let D be a $3x3$ real matrix
Let F be a vector of real numbers
Let f, d be real numbers
if $seesPrey(P^0)$ **then**
    $(X_p, Y_p) \Leftarrow getPreyRelativeLocationTo(P^0)$
else
    $(X_p, Y_p) \Leftarrow (0, 0)$
end if
$P \Leftarrow getAllPredatorsInsideFOVOf(P^0)$
for all $((x, y) \mid 1 \leq x \leq 3, 1 \leq y \leq 3)$ **do**
    $D_xy \Leftarrow calculateExtendedKorfFitness(x, y)$
end for
$f \Leftarrow compressAllFitnesses(D)$
$sendCompressedFitnessToOtherPredators(f, P)$
$F \Leftarrow receiveCompressedFitnessesOfPredators(P)$
for all $(P^i \in P)$ **do**
    $f \Leftarrow getCompressedFitnessOf(P^i, F)$
    $(x, y) \Leftarrow getLocationOfPredator(P^i)$
    $activateNeuralNetworkWithValues(x, y, f)$
end for
$f \Leftarrow getCompressedFitnessOf(P^0, F)$
$(x, y) \Leftarrow getLocationOfPredator(P^0)$
$d \Leftarrow flushNeuralNetworkWithValues(x, y, f)$
$c \Leftarrow getNextCellToMoveTo(d)$
$movePredatorTo(P^0, c)$

It is important to point some details. The functions $compressAllFitnesses$ and $getNextCellToMove$ do not directly compress the 9 fitness values into 1: there is no biyective function to do that. But that is not a problem, because

the most relevant information for predators in order to coordinate their actions is not exactly the fitness itself, but the priority to choose each cell as next movement. Therefore, $compressAllFitnesses$ forms a unique number λ by concatenating the numbers of the 9 cells ordered by their fitness. For example, $compressAllFitnesses$ could form the number $\lambda = 854713269$ meaning that the cell number 8 is the best fitted one, while the cell number 9 is the poorly fitted one. Then, as long as λ is a discrete integer value, it is now possible to associate an $\alpha \in [-1, 1]$ though a biyective function. $compressAllFitnesses$ finally returns α, while $getNextCellToMove$ reconstructs λ from an α and returns the first digit (8 in our previous example).

Regarding the neural networks, they are trained and evolved using Neuroevolution of Augmenting Topologies (NEAT [8]). Populations of predators are created each epoch, and each predator has its own neural network. Each neural network has 3 inputs and 1 output. The 3 inputs are designed to receive the (x, y) location of a predator P^i, scaled to $[-1, 1]$ depending on the FOV, and the compressed fitness value P^i. The neural network is expected to sequentially receive these 3 inputs from each of the predators inside the FOV o P^0, and to activate all its neurons once for each triplet of inputs. Finally, the neural network receives the 3 inputs related to P^0, activating and flushing the net to get its final output value.

4 Results

To validate our approach we compared the results of the 3 methods (ExtKorf, ExtKorf+CSN and ExtKorf+CSN+NECool) with the original of Korf in the same environment conditions, but variating the FOV. We have measured predators against two different preys: a random moving prey, and an evading prey. The second prey moves 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&Vlassis' Pursuit Domain Package (PDP). Concretely, we used a $30x30$ cells field, allowing agents to move diagonal, with the prey starting on the center and predators starting randomly placed. We lauched 4 predators to capture 1 prey, following capture method 2 (4 predators orthogonally surrounding the prey, without touching it). Finally, in case of collision, only predators colliding were penalized. Simulation was always ran for 500 consecutive episodes, and we got the average results.

Our first comparative experiment was to measure the improvement in cycles and collisions of the Extended Korf algorithm against original Korf's. For this experiment the results show an improvement of an order of magnitude in most cases. In figure 2 we see that the improvement is much greater when the FOV is more limited (3 is the minimum FOV considered). It is normal, though, that the minimum improvement in cycles and collisions happens when agents have 15 cells of FOV (i.e. they can sense the whole world at once). In this case, agents do not lose cycles in trying to find the prey, and they go straight to capture it.

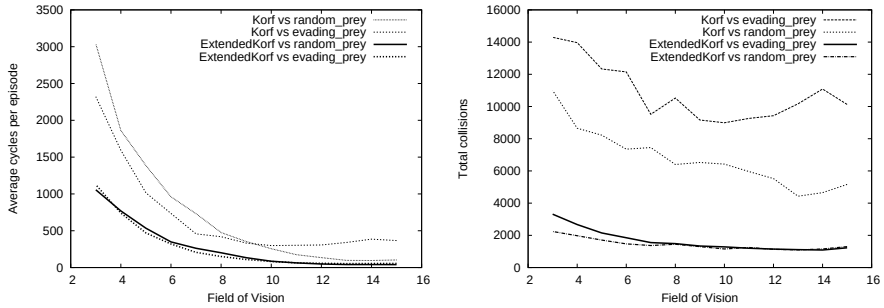

Fig. 2. Comparison between original and Extended Korf's model with respect to average cycles per episode and total collisions in 500 episodes

As we stated in previous section, figure 2 clearly shows that there are two major ways of improvement: more efficiently finding the prey and avoiding collisions. We have made two proposals, each one to cover each of these two ways. Our first proposal was about the CSN protocol, enabling predators to locate the prey by using the indications got from other predators. In order to check the relative improvement of using this protocol, we have compared Extended Korf's model against itself with and without CSN. Figure 3 shows the results of this comparison. As expected, results suggest that there are plenty of situations in which CSN saves cycles of exploration to the predators. So, CSN represents an improvement which leads predators to earlier find the prey on average, and CSN is more significant when FOV is minimal.

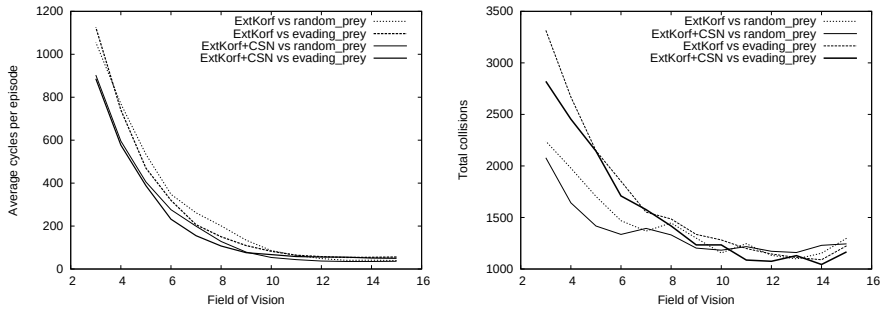

Fig. 3. Comparison between Extended Korf's model and Extended Korf's model with CSN with respect to average cycles per episode and total collisions in 500 episodes

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 hear 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 find the prey and the more difficult to communicate with others.

Although 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 $f = \frac{6001}{n+10c+1}$, 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 run for them the same 500 episodes test we run earlier, but this time against ExtKorf+CSN predators. The result (see figure 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.

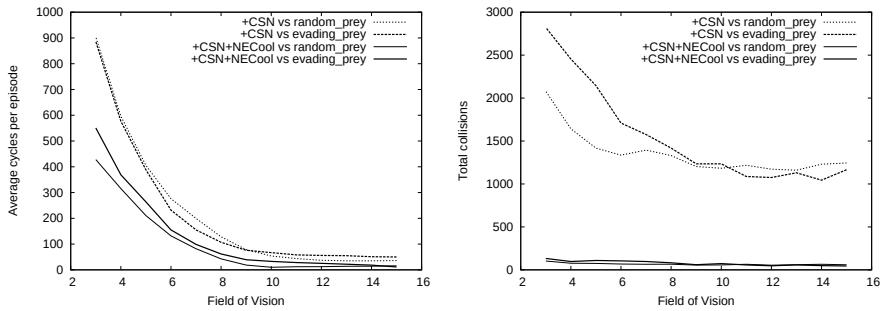

Fig. 4. Comparison between Extended Korf's model with CSN and with CSN+NECool with respect to average cycles per episode and total collisions in 500 episodes

5 Conclusions and Further Work

This paper describes a new proposal for improving cooperation between predators in the pursuit domain. This new proposal mixes the efficiency of greedy approaches with Machine Learning techniques to get the best of both. The proposal suggests to extend the greedy approach proposed by Korf (ExtKorf) and to add two cooperatives strategies: Cascading Sight Notice (CSN) and NEAT Coordination Protocol (NECool).

To validate this approach we compared the results of ExtKorf, ExtKorf+CSN and ExtKorf+CSN+NECool between them pairwise and with Korf's using Pursuit Domain Package (PDP) in more challenging environment conditions. Our

first experiment measured performance in cycles and collisions of Extended Korf's algorithm against Korf's. Results shown an improvement of an order of magnitude in most cases. Our second experiment demonstrated that CSN improves the average cycles to capture the prey, but only significantly in few cases. This was mainly due to the necessity of predators to be inside FOV of others to communicate with them. Our third experiment added NECool and compared it with ExtKorf+CSN. Results shown a dramatic reduction of total collisions between predators, which maps directly to a significant improvement (25 to 35%) in average number of cycles to capture the prey.

Therefore, we conclude that mixing greedy and evolutive approaches is a promising path to explore, as our experiments have shown. 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. M. Benda, V. Jagannathan, and R. Dodhiawalla. On optimal cooperation of knowledge sources. Technical Report Tech. Rep. BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue, WA, 1986.
2. W. Chainbi, C. Hanachi, and C. Sibertin-Blanc. The Multi-agent Prey-Predator problem : A Petri net solution. In *Proceedings of the IMACS-IEEE-SMC conference on Computational Engineering in Systems Application (CESA'96)*, pages 692–697, Lille, France, 1996.
3. Thomas Haynes and Sandip Sen. Evolving behavioral strategies in predators and prey. In Sandip Sen, editor, *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, pages 32–37, Montreal, Quebec, Canada, 20-25 1995. Morgan Kaufmann.
4. Kam-Chueun Jim and C. Lee Giles. Talking helps: Evolving communicating agents for the predator-prey pursuit problem. *Artificial Life*, 6(3):237–254, 2000.
5. Kengo Katayama, Takahiro Koshiishi, and Hiroyuki Narihisa. Reinforcement learning agents with primary knowledge designed by analytic hierarchy process. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 14–21, New York, NY, USA, 2005. ACM.
6. Jelle R. Kok and Nikos Vlassis. The pursuit domain package. Technical Report Technical Report IAS-UVA-03-03, Informatics Institute, University of Amsterdam, The Netherlands, August 2003.
7. Richard E. Korf. A simple solution to pursuit games. In *Proceedings of the 11th International Workshop on Distributed Artificial Intelligence*, pages 183–194, Glen Arbor, MI., February 1992.
8. Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.
9. Peter Stone and Manuela M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

10. Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative learning. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 487–494. Morgan Kaufmann, San Francisco, CA, USA, 1997.