Advanced Robotics, Vol. 15, No. 8, pp. 815–832 (2001)

Full paper

Cooperative Q-learning: the knowledge sharing issue

MAJID NILI AHMADABADI[1,2,*], MASOUD ASADPOUR[1,2] and EIJI NAKANO[3]

[1] *Robotics and AI Laboratory, Department of Electronic and Computer Engineering, Faculty of Engineering, University of Tehran, Tehran, Iran*
[2] *School of Intelligent Systems, Institute for Studies on Theoretical Physics and Mathematics, Tehran, Iran*
[3] *Advanced Robotics Lab., GSIS, Tohoku University, Japan*

Received 29 January 2001; accepted 11 April 2001

Abstract—A group of cooperative and homogeneous Q-learning agents can cooperate to learn faster and gain more knowledge. In order to do so, each learner agent must be able to evaluate the expertness and the intelligence level of the other agents, and to assess the knowledge and the information it gets from them. In addition, the learner needs a suitable method to properly combine its own knowledge and what it gains from the other agents according to their relative expertness. In this paper, some expertness measuring criteria are introduced. Also, a new cooperative learning method called weighted strategy sharing (WSS) is introduced. In WSS, based on the amount of its teammate expertness, each agent assigns a weight to their knowledge and utilizes it accordingly. WSS and the expertness criteria are tested on two simulated hunter–prey and object-pushing systems.

Keywords: Learning; cooperation; expertness; knowledge sharing.

1. INTRODUCTION

Parallelism, scalability, simpler construction and cost effectiveness are among the main characteristics of multi-agent systems [1, 2]. Having these attributes, multi-agent systems are used to solve complicated problems, search in large domains, execute sophisticated tasks, and make more fault-tolerant and reliable systems.

In most of the existing systems, agents' behavior and coordination schemes are desinged and fixed by the designer. But, an agent with limited and fixed knowledge and behavior cannot be sufficiently effective in a dynamic, complex or changing environment. Therefore, to have all benefits of implementing a multi-agent system, each agent and the team must learn to cope with the new, unseen and changing situations.

*To whom correspondence should be addressed. E-mail: mnili@ut.ac.ir

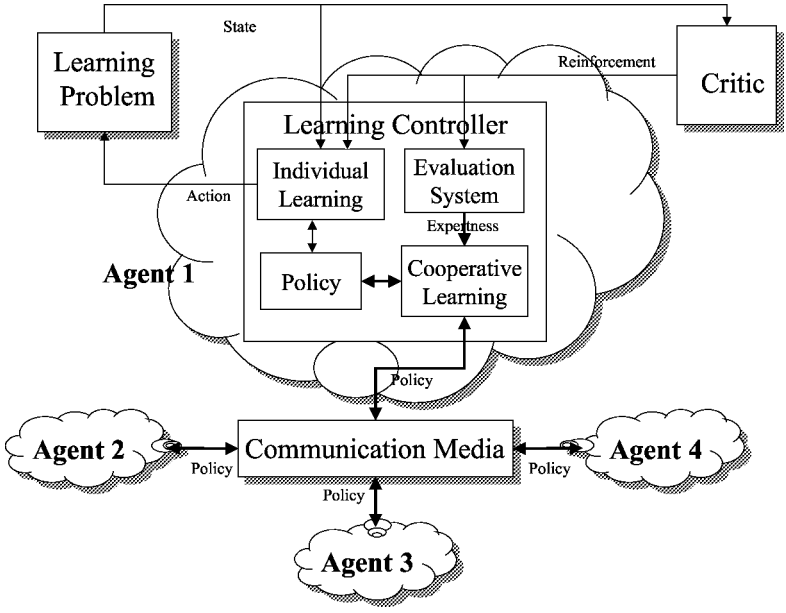

Figure 1. A cooperative learning system.

In approximately all of the present multi-agent teams, agents learn individually. However, the main favorable attributes of multi-agents systems can also be used in the learning process. In other words, agents are not required to learn all things from their own experiences.

Each agent can observe the others, and learn from their situation and behavior. Moreover, agents can consult with more expert agents or get advice from them. Agents can also share their information and learn from this data, i.e. the agent can cooperate in learning. Figure 1 shows an example of such a system.

Cooperative learning can be observed in human and some animal societies. As a result of having more knowledge and information acquisition resources, cooperation in learning in multi-agent systems may result in a higher efficiency compared to individual learning [2]. Researchers have shown an improvement in learning even when simple cooperative learning methods are used [3].

Cooperative learning has not been intensively investigated. In almost all of multi-agent learning papers, cooperation is uni-directional between a fixed trainer agent and a learner. We believe all agents can learn something from each other provided that some proper measures and methods are implemented.

One of the most important issues for the learner agent in cooperative learning is evaluation of the information and the knowledge it gets from the others. Also, assessment of the behavior and the intelligence level of the other agents is another important problem to be addressed. In other words, the learner agent must evaluate the level of intelligence or the expertness of the other agents. In addition, a proper

method must be devised for each agent to combine its own knowledge and that of the others according to their relative expertness.

These three issues are very complicated in general. Therefore, in this paper, we pay attention to finding solutions for homogeneous, independent and cooperative Q-learning agents.

Related research is reviewed in the next section. Then, a new cooperative learning strategy, called weighted strategy-sharing (WSS), is introduced. Later, individual learning is explained and some expertness measures are devised. The WSS method and the effects of implementing the expertness measures are studied on the hunter–prey and object-pushing problems in Sections 4 and 5. A conclusion of this paper is given in the last section.

2. RELATED WORK

Samuel [4] used the Competitive Learning algorithm to train a Checker game player. In his method, the cooperator agent has the role of the enemy or the evaluator and tries to find the weak strategies of the learner. In the Ant Colony System [5], some ants learn to solve the travelling salesman problem by non-verbal communication through pheromones on the edges of a graph.

Imitation [6] is one of the cooperative learning methods. In imitation, the learners watch the actions of a teacher, learn them and repeat the actions in similar situations. This method does not affect the performance of the teacher [7]. For example, in [6], a robot perceives a human doing a simple assembly task and learns to reproduce it again in different environments. Hayes and Demiris [8] have built a robotic system in which a learner robot imitates a trainer that moves in a maze. The robot learns to escape from the maze later.

Yamaguchi *et al.* [9] have developed a robotic imitation system to train ball-pusher robots. In this system, agents learn individually based on Reinforcement Learning and they have the ability to imitate each other. In their research, three methods for imitation are used: Simple Mimetism, Conditional Mimetism and Adaptive Mimetism.

In Simple Mimetism all agents imitate each other with an imitation rate when they are neighbors. In this method, the Inter-Mimetism Problem occurs when two neighbors wait to imitate each other and do nothing. This problem is solved by Conditional Mimetism in which only the low-performance agent (performance is measured based on sum of the rewards and the punishments received in n previous actions) imitates the other one. Adaptive Mimetism is similar to Conditional Mimetism and the imitation rate is adjusted based on the difference between the performances of two neighbor robots.

Cooperation in learning can also be done when agents share their sensory data and play the role of a scout for each other [3]. Episode Sharing can be used to communicate the (state, action, reward) triples between the Reinforcement

Learners [3]. Tan showed that sharing episodes with an expert agent could significantly improve the group learning [3].

In the Collective Memory method, the learners put the learnt strategy or experienced episodes on a shared memory [10] or have a unique memory and update it cooperatively [3].

A Cooperative Ensemble Learning System [11] has been developed as a new method in neural network ensembles. In that study, a linear combination of outputs of concurrent learning neural networks is used as a feedback to add a new penalty term to the error function of each network.

Provost and Hennessy [12] developed a Cooperative Distributed Learning System which is used when the training set is huge. First, the training set is divided into k smaller training subsets and k rule-learning agents learn local rules from these subsets. Then, the rules are transmitted to other agents for evaluation. A rule will be accepted as a global rule if it satisfies the evaluation criterion.

Maclin and Shavlic [13] used an Advice Taking scheme to help a Connectionist Reinforcement learner. The learner accepts advice in the form of a simple computer program, compiles it, represents the advice in a neural network form and adds it to its current network.

In most of the reviewed research, cooperation is uni-directional between a predefined trainer and a pre-specified learner agent. However, in the real world, all of the agents may learn something from each other; even from the non-experts.

In the strategy-sharing method [3], members of a multi-agent system learn from all of the agents. The agents learn individually by the Q-learning version of Reinforcement Learning. At defined times, the agents gather the Q-tables of the other agents and adopt the average of the tables as their new strategy.

In this system, all of the agents' knowledge is equally used and the agents do not have the ability to find good teachers. Moreover, it seems that simple averaging of the Q-tables is non-optimal when the agents have different skills and experiences. Additionally, the Q-tables of the agents become equal after each cooperation step. This decreases the agents' adaptability to environment changes [9].

To overcome these drawbacks, we have developed a new strategy-sharing method based on expertness detection where the agents assign some weights to the other agents' Q-tables.

3. WSS METHOD

In the WSS method (Algorithm 1) it is assumed that members of a group of n homogeneous agents are learning in some distinct environments, so their actions do not change the others learning environment (and do not produce the Hidden State Problem [14]).

Algorithm 1. WSS algorithm for agent a_i.

(1) *Initialize*
(2) while not *EndOfLearning* do
(3) begin
  (4) If *InIndividualLearningMode* then
  (5) begin {*Individual Learning*}
    (6) $x_i := FindCurrentState()$
    (7) $a_i := SelectAction()$
    (8) $DoAction(a_i)$
    (9) $r_i := GetReward()$
    (10) $y_i := GoToNextState()$
    (11) $v(y_i) := \mathrm{Max}_{b \in \mathrm{actions}} Q(y_i, b)$
    (12) $Q_i^{\mathrm{new}}(x_i, a_i) := (1 - \beta_i) Q_i^{\mathrm{old}}(x_i, a_i) + \beta_i (r_i + \gamma_i V(y_i))$
    (13) $e_i := UpdateExpertness(r_i)$
  (14) end
  (15) else {*Cooperative Learning*}
  (16) begin
    (17) for $j := 1$ to n do
      (18) $e_j := GetExpertness(A_j)$
    (19) $Q_i^{\mathrm{new}} := 0$
    (20) for $j := 1$ to n do
    (21) begin
      (22) $W_{ij} := ComputeWeights(i, j, e_1 \ldots e_n)$
      (23) $Q_j^{\mathrm{old}} := GetQ(A_j)$
      (24) $Q_i^{\mathrm{new}} := Q_i^{\mathrm{new}} + W_{ij} * Q_j^{\mathrm{old}}$
    (25) end
  (26) end
(27) end

The agents are learning in two modes: Individual Learning Mode and Cooperative Learning Mode (lines 4 and 15 of Algorithm 1). At first, all of the agent are in the Individual Learning Mode. Agent i executes t_i learning trials, based on the one-step Q-learning version of reinforcement learning (different t_is causes different experiences). Each learning trial starts from a random state and ends when the agent reaches the goal. At the time when a specified number of individual trials

is performed (which is called the cooperation time) all agents stop the Individual Learning Mode and switch to Cooperative Learning Mode.

In the Cooperative Learning Mode, each learner assigns some weights to the other agents according to their expertness values (Section 3.2). Then, it takes a weighted average of the others' Q-tables and uses the resulted table as its new Q-table. (In Algorithm 1, multiplication ($*$) and Summation ($+$) operators must be specified according to the knowledge representation method. For example, in the one-step Q-learning method, $*$ is the scalar matrix multiplication operator and $+$ represents the matrix summation.)

3.1. Individual learning based on reinforcement learning

In this paper, one-step Q-learning is used for the Individual Learning Mode.

Reinforcement learning is one of the simplest and widely used learning method, and has many similarities to human experiences. In this method, the learner perceives something about the state of its environment and, based on a predefined criterion, chooses an action. The action changes the world's state and the agent receives a scalar 'reward' or 'reinforcement', indicating the goodness of its new state. After receiving the reward or the punishment, it updates the learnt strategy based on a learning rate and some other parameters.

In the one-step Q-learning algorithm [15, 16] the external world is modeled as a Markov Decision Process with discrete time finite states. Next to each action, the agent receives a scalar 'reward' or 'reinforcement'.

The state-action value table, the Q-table, which estimates the long-term discounted reward for each state/action pair, determines the learned policy of the agent. Given the current state and the available actions a_i, a Q-learning agent selects action 'a' with the probability (P) given by the Boltzmann distribution (line 7 of Algorithm 1):

$$P(a_i|x) = \frac{e^{Q(x,a_i)/\tau}}{\sum_{k\in\text{actions}} e^{Q(x,a_k)/\tau}}, \tag{1}$$

where τ is the temperature parameter and adjusts the randomness of the decision. The agent executes the action (line 8), receives an immediate reward r (line 9), moves to the next state y (line 10) and updates $Q(x, a)$ as (line 12):

$$Q(x,a) \leftarrow (1-\beta)Q(x,a) + \beta\big(r + \gamma V(y)\big), \tag{2}$$

where β is the learning rate, $\gamma (0 \leqslant \gamma \leqslant 1)$ is a discount parameter and $V(x)$ is given by (line 11):

$$V(y) = \max_{b\in\text{actions}} Q(y,b), \tag{3}$$

Q is improved gradually and the agent learns when it searches the state space.

3.2. Measuring the expertness

In the WSS Method, weights of each agent's knowledge must be properly specified so that the group learning efficiency is maximized.

The expertness measuring criterion can significantly affect the learning efficiency. This can be also observed in human societies — in those, a learner evaluates the others' knowledge with respect to their expertness. In other words, each learner tries to find the best evaluation method to find out how much the others' knowledge is reliable.

Different mechanisms for choosing the expert agents are used. Most researchers have a pre-specified expert agent(s). For example, in [3], the expert agent is predefined and it is not changed when learning. This expert agent, which is trained or pre-programmed, does not learn and only helps the learners.

In the strategy-sharing method [3], the expertness of the agents are assumed to be equal. Nicolas Meuleau [17] used user judgment for specifying the expert agent. This method requires continuous supervision by a human being.

In [18] different but fixed expertness values are assumed for the agents. Differences in the expertness values can be due to having initial knowledge, different experiences, different learning algorithms or different training sets. It is noteworthy that the difference in the expertness values may change during the learning process and cannot be assumed to be constant.

Yamaguchi *et al.* [9] specified the expert agents by means of their successes and failures during their current n moves. In their work, the expertness value of each agent is the algebraic sum of the reinforcement signals it received for its current n moves. This means that agents with more successes and fewer failures are considered to be more expert. This method is not optimal in some situations.

For example, an agent that has faced many failures has some useful knowledge to be learnt from it. In fact, this agent may not know the ways to reach the goal, but it is aware of those not leading to the target. Also, the expertness of an agent at the beginning of the learning process — that has not faced many failures — is less than those that have learned for a longer time and have naturally faced more failures.

3.2.1. Some experness evaluation criteria. Considering the above discussions, six methods for measuring the agents' expertness are introduced in this paper. These methods are:

(i) Normal (Nrm): Algebraic sum of the reinforcement signals.

(ii) Absolute (Abs): Sum of absolute value of the reinforcement signals.

(iii) Positive (P): Sum of the positive reinforcement signals.

(iv) Negative (N): Sum of absolute value of the negative reinforcement signals.

(v) Average Move (AM): Inverse of the number of moves each agent does to reach the goal.

(vi) Gradient (G): The change in the received reinforcement signals since the last cooperation time.

3.3. The weight-assigning mechanism

In this paper, to decrease the amount of communication required to exchange Q-tables, the learner uses only the Q-tables of more expert agents. Therefore, partial weights of the less expert agents are assumed to be zero.

Learner i assigns the weight to the knowledge of agent j as:

$$W_{ij} = \begin{cases} 1 - \alpha_i & \text{if } i = j, \\ \alpha_i \dfrac{e_j - e_i}{\sum_{k=1}^{n} (e_k - e_i)} & \text{if } e_j > e_i, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $0 \leqslant \alpha_i \leqslant 1$ is the impressibility factor and shows how much agent i relies on the others knowledge. e_i and e_j are the expertness value of agents i and j respectively, and n is the total number of the agents.

Substitution of this formula in weighted averaging formula (line 24 of Algorithm 1) results in the following:

$$Q_i^{\text{new}} \leftarrow (1 - \alpha_i) * Q_i^{\text{old}} + \alpha_i * \sum_{j \in \text{Exprt}(i)} \left(\frac{e_j - e_i}{\sum_{k=1}^{n} (e_k - e_i)} * Q_j^{\text{old}} \right) \tag{5}$$

$$\text{Exprt}(i) = \{j | e_j > e_i\}.$$

3.4. Special cells communication

Two mechanisms called Positive Only (PO) and Negative Only (NO) are introduced to eliminate the communication of some cells. In PO, agents send only positive-value cells of their Q-tables to others and the expertness of the agents is measured by positive criterion. In NO, agents communicate only negative-value cells of their Q-tables to others and the expertness is measured by negative criterion.

4. SIMULATION RESULTS ON THE HUNTER–PREY PROBLEM

The 'Hunter and prey' problem [3] is one of the classical problems to study the learning process and is a suitable testbed for comparing different learning methods. In this paper, all experiments consist of three hunters, that search independently in a 10×10 environment to capture a prey agent. Hunters can move with a speed between 0 and 1 and the prey can move with a speed between 0 and 0.5. The

prey is captured when its distance from the hunter is less than 0.5 (which is called the reward field). Upon capturing the prey, the hunter receives +R reward and −P punishment otherwise.

Each agent has a visual field in that it can locate the other agents and the environment. In the studied experiments, the visual field of the hunter is 2 and that of the prey is set as 2. A greater visual field helps the prey to escape before the hunter sees it.

The states of the hunter are specified with respect to the prey (x, y) coordinates in its local coordination frame. If the hunter is not in its visual field, a default state is considered. Actions of the hunter consist of rotations and changing its velocity: $a = (V, \theta)$. The distance, the velocity difference and the direction difference are divided into sections of 1 distance unit, 0.5 velocity units, and 45°, respectively.

In order to complicate the learning problem and to clearly show the differences in efficiency of the learning algorithms and the expertness measures, simulations are performed on a simple and a complex version of the hunter−prey problem. In the simple version, similar to other research, the moving pattern of the prey is irregular and random. In the complex version, the prey moves based on the potential field model and escapes from the hunter. We call this agent intelligent.

In the potential field model, there are four walls around the environment, and the prey and the hunter are assumed to be electropositive materials. Therefore, the prey selects the path with minimum potential. The repulsive force of the hunter is considered 1.5 times that of the walls. The hunter and each wall are modeled as a spot and a linear load, respectively. An example of computing the resultant force on the prey is showed in Fig. 2.

In an environment with intelligent prey, each hunter's movements may affect the prey. So, if several hunters learn together in an environment, the effects of each individual hunter on the others' learning cannot be easily calculated. So, only one hunter is in the environment in each trial. Also, to create agents with different expertness, the agents have different learning times (t_is). The first hunter learns six trials, then the second one is permitted to do three trials and, finally, the last hunter does one learning trial. In the other cases the hunters have equal t_is. The total number of individual learning trials is 1000 and cooperation occurs after each 50 individual learning trials. The reward and punishment signals are one of six pairs: (10, −0.01), (10, −0.1), (10, −1), (5, −0.01), (5, −0.1) and (5, −1).

In the simulations, each learning trial ends when the hunter captures the prey. At the beginning of each individual learning trial, agents are in a random situation. The one-step Q-learning parameters are set to $\beta = 0.01$, $\gamma = 0.9$ and $T = 0.4$. Q-table values are initialized to zero and all agents have $\alpha_i = 0.7$. Also, for trial n, the average number of hunter actions to capture the prey over past n trials is calculated.

4.1. Equal experiences

The average number of moves using six mentioned reinforcement values and the introduced expertness measuring criteria are shown in Figs 3 and 4 for individual

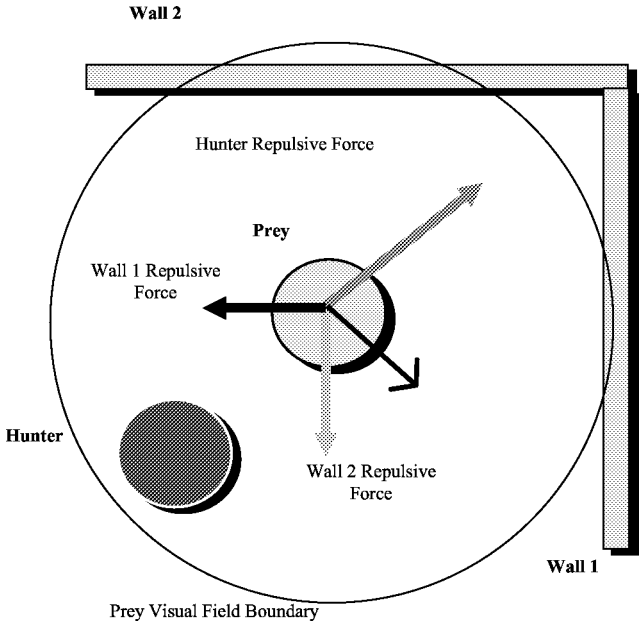

Figure 2. An example of computing the resultant force.

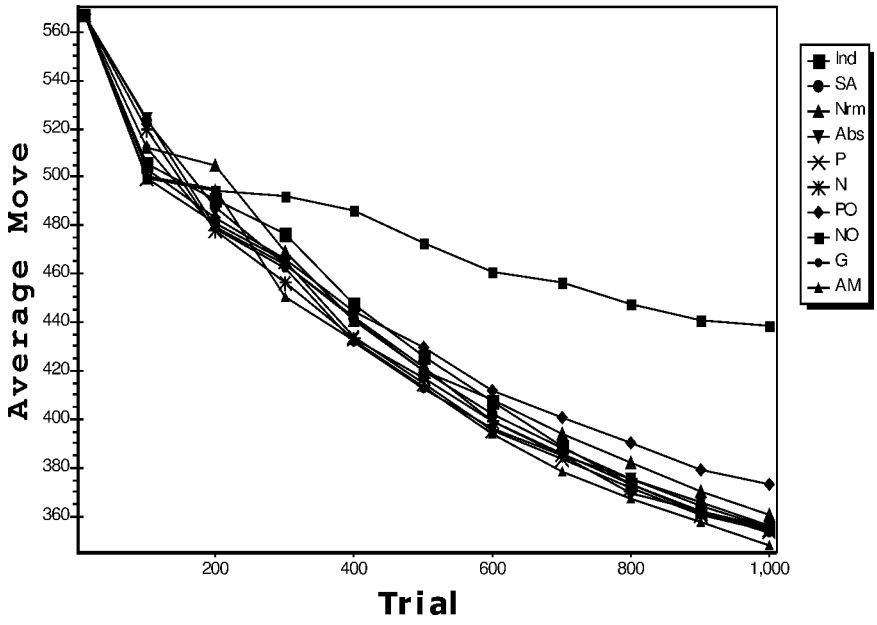

Figure 3. Average number of moves in the random prey and equal experience case.

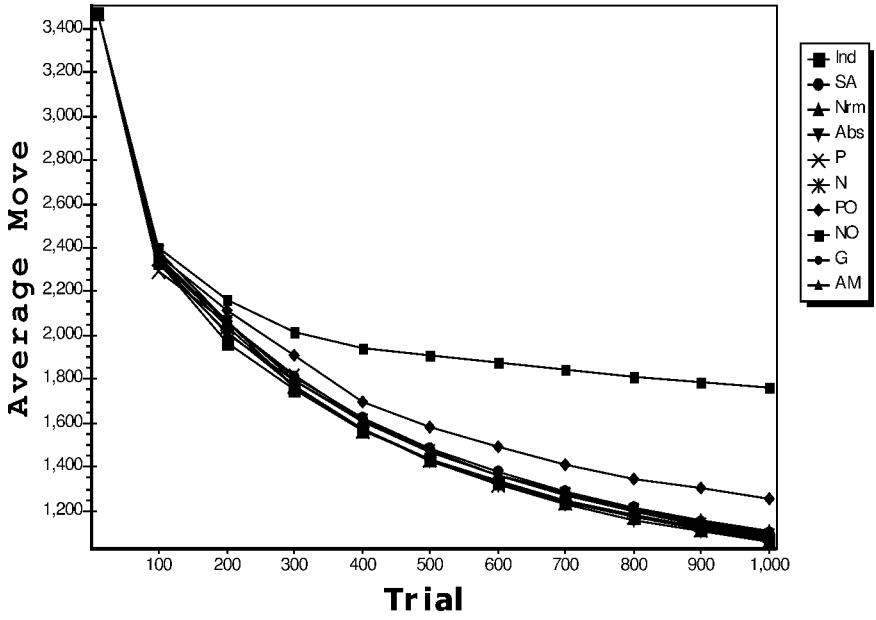

Figure 4. Average number of moves in the intelligent prey and equal experience case.

Table 1.
Improvement (%) in the equal experience case

Prey	SA	Nrm	Abs	P	N	PO	NO	G	AM
Random	−0.047	−1.36	−0.141	0.375	0.235	−4.972	−23.358	0.704	2.111
Intelligent	−3.136	−3.339	0.936	−1.95	−0.156	−17.663	−65.096	−1.264	−0.452

and cooperative learning methods. Also, improvements gained in learning over the independent learning method are given in Table 1.

All of the cooperative learning methods (except PO and NO) have approximately the same results as independent learning. PO and NO are the worst methods, because these methods change the probability distribution of actions in Boltzmann selection function and make the selection probability of positive-value and negative-value actions closer, thus the probability of selecting an inefficient action rises. NO is worse than PO, because even a significant difference of two negative-value actions makes little change in the selection probability of the actions. However, a little difference of two positive-value actions can raise the probability of choosing the better action significantly.

4.2. Different experiences

The average numbers of moves for the six described reinforcement values are shown for each cooperation method in Figs 5 and 6. The improvement percent of each method is given in Table 2.

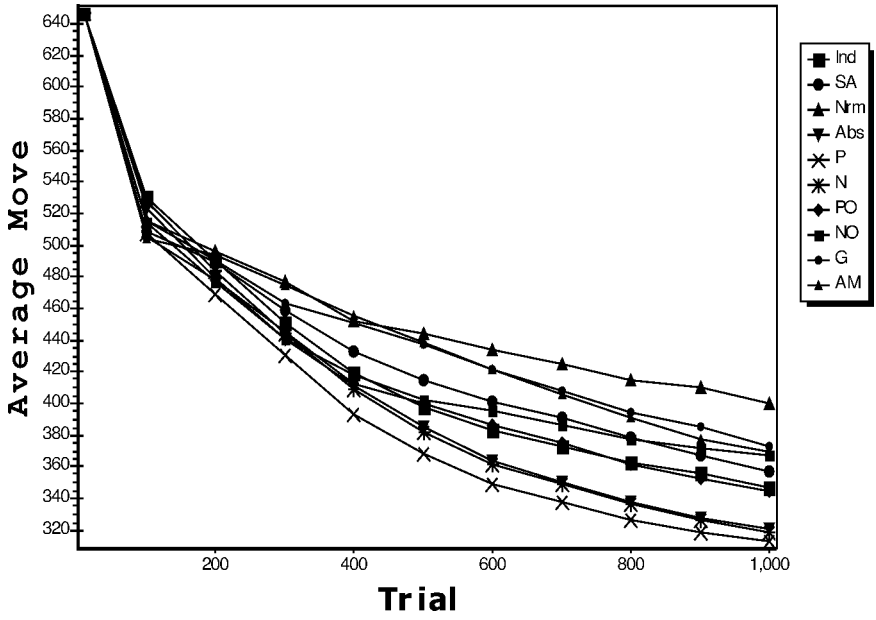

Figure 5. Average number of moves in the random prey and different experience case.

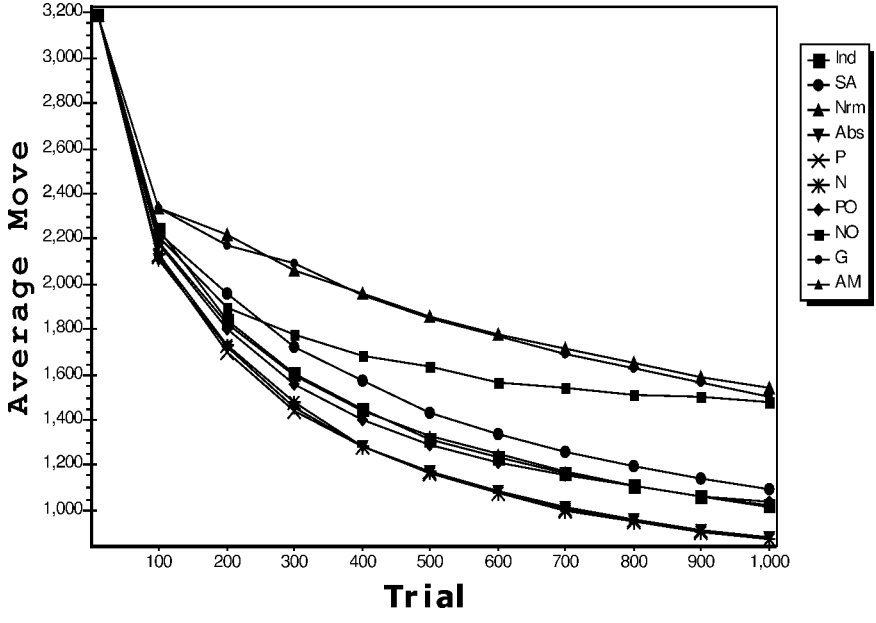

Figure 6. Average number of moves in the random prey and different experience case.

Table 2 shows that Abs, P and N result in improvement in all cases. In the random-prey case, where the number of punishments is less than in the intelligent-

Table 2.
Improvement (%) in the different experience case

Prey	SA	Nrm	Abs	P	N	PO	NO	G	AM
Random	−2.735	−15.307	7.582	9.933	8.301	0.845	−5.777	−7.486	−6.286
Intelligent	−7.572	−51.153	13.9	14.244	14.44	−1.676	−44.841	−47.49	−0.654

prey simulator, P criterion has the best results; however, in intelligent-prey case, N criterion works better.

N and G have the worst results. These criteria make the mistake in assigning expertness to the agents. It is due to the fact that the experienced agents have performed more actions and, consequently, have received more punishments. Therefore, their expertness becomes negative and less weight is assigned to their knowledge.

The Average Move criterion has negative impact in the random-prey case and has approximately no effect in the intelligent-prey system. In this criterion, the difference in the agents' expertness is small, even when they have considerably different numbers of moves.

NO has a negative effect on learning in both cases and PO causes no improvement. Simple Averaging (SA) has a negative impact in both systems, because it assigns equal weights to the agents with different expertness.

Four samples of hunting the intelligent prey are shown in Fig. 7. Circles show agents; agent 1 is the intelligent prey, and agents 2 and 3 are the hunters using the P expertness measure.

5. LEARNING TO PUSH AN OBJECT COOPERATIVELY

In the object-pushing problem, simulated in this paper, two robots learn to push an object toward a target area cooperatively (Fig. 8). Pushing forces (F_1 and F_2) are applied to the object horizontally at points A and B. The friction coefficient is fixed and the friction force is uniformly distributed over the object surface. The pushing velocity is slow and the inertia forces are neglected. Therefore, considering Fig. 8, linear (a) and angular (α) accelerations can be computed as:

$$a = \frac{F_1 \cdot \cos(\theta_1 - 45') + F_2 \cdot \cos(\theta_2 - 45') - F_s}{M}, \tag{6}$$

$$\alpha = \frac{F_1 \cdot r \cdot \sin(\theta_1) - F_2 \cdot r \cdot \sin(\theta_2)}{I}, \tag{7}$$

where M is mass of the object and I is its moment of inertia. Mechanical parameters are set to $M = 1$ (kg), $I = 1$, $r = 1$ (m), $\mu_s = 0.5$ and $g = 9.8$ (N/kg).

To avoid Structural Credit Assignment [19], a common Q-table for two robots is used. In other words, each action (a_p) is a joint action of two robots: $a_p = (a_{p1}, a_{p2})$.

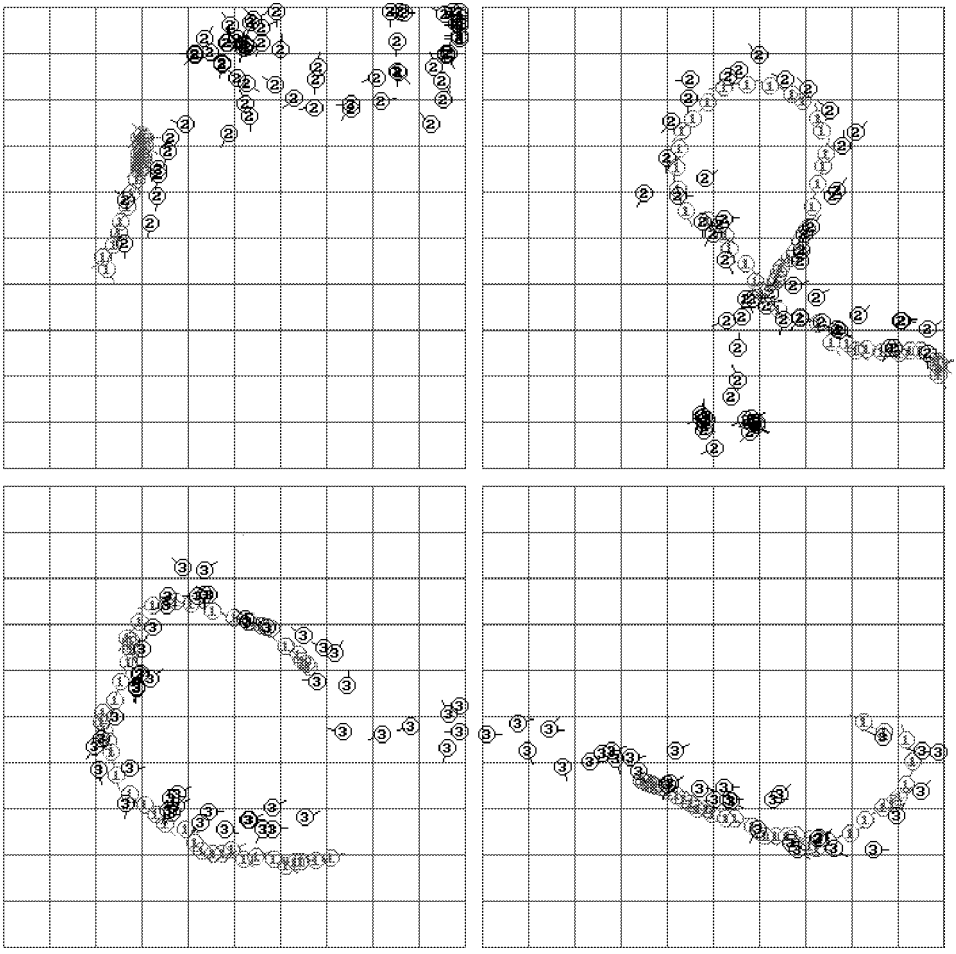

Figure 7. Four samples of hunting the intelligent prey. Circles show agents; agent 1 is the intelligent prey, and agents 2 and 3 are the hunters using the P expertness measure.

States of the group are specified in the object coordinate system and relative to the target: $s = (x, y, \theta)$. Ranges of x, y and θ are divided into segments of 1 (m), 1 (m) and 45 ($\circ$), respectively. The environment is 10×10, and since x and y can be between 10 and -10, the robots have $8 \times 20 \times 20 = 3200$ states. The actions of the robots are specified based on the amount and angle of the forces.

Each force is applied for 4 s and the robots follow the object until it stops. The target position is at (8, 8) and the initial position of the object is (2, 2). If the object crosses the walls, the robots are positioned at the initial point. Each learning trial starts from the initial position and ends when the robots take the object into the target area or their number of actions exceeds 2000.

To implement cooperative learning, three groups of such systems are used. Their learning trials are divided based on $t_1 = 6$, $t_2 = 3$ and $t_3 = 1$, and cooperation times

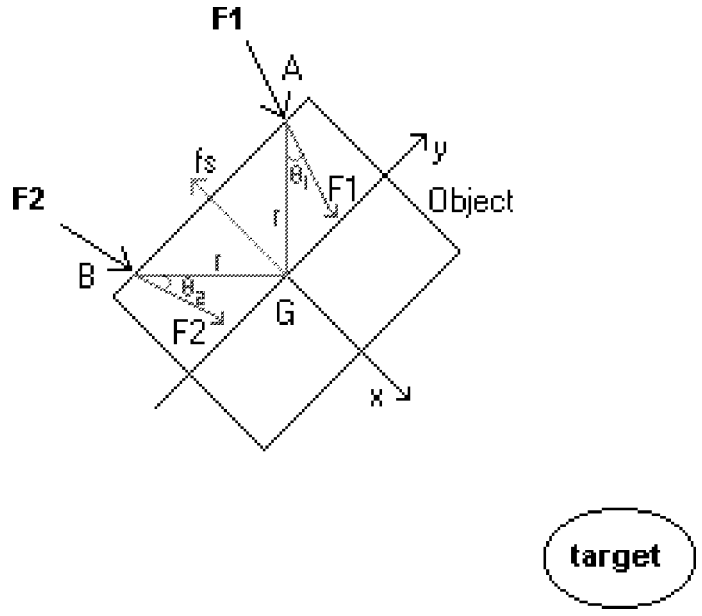

Figure 8. The object pushing model.

occur after each 100 learning trials. The maximum number of trials is 50 000, and other parameters are $\alpha = 0.7,\ \beta = 0.01,\ \gamma = 0.9$ and $T = 0.4$.

To define suitable reinforcement signals, the average number of randomly selected moves of each group to arrive at the target (i.e. without learning) is computed. It was approximately 200 moves. Then the rewards are set to 10 and three cases of punishments are assigned such that sum of the received punishments are approximately lower than (-0.01×200), equal to (-0.05×200) or greater than (-0.1×200) the reward $(+10)$.

5.1. The simulation results

A sample of the learned path for pushing the object is shown in Fig. 9.

The average number of moves to push the object into the target area after 50 000 learning trials is shown in Table 3 for each reinforcement function. It is shown that SA has little positive effect on the learning, compared to independent learning. Also, in all cases, WSS based on N, Abs, P and N criteria have better results than SA. When the punishments are greater than the rewards at the beginning of learning, N is the best criterion and P is the worst one among the four above-mentioned expertness measures. However, in the case where the rewards are greater than the punishments, P is the best criterion and N is the worst one. Also when the punishments and the rewards are approximately equal, Abs has the best results. Since any little difference in the rewards and the punishments significantly affects the weights in this case, the N criterion has the worst results. Naturally, P and N have approximately the same results in this case.

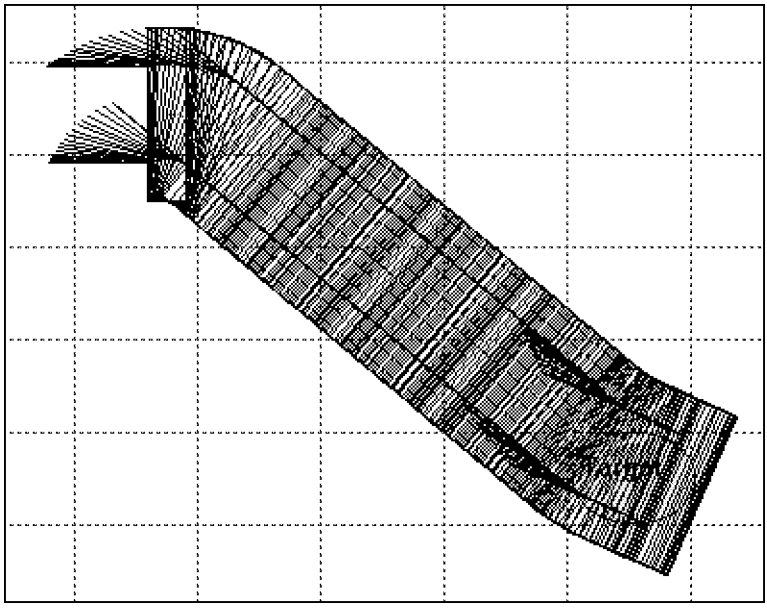

Figure 9. A sample of the learned object-pushing path.

Table 3.
Average number of moves to push the object to the target area

Reward, Punishment	Independent	SA	N	Abs	P	N
10, −0.01	25.0	24.8	20.6	21.4	20.6	21.5
10, −0.05	27.9	26.9	21.4	20.8	21.0	21.1
10, −0.1	27.3	26.4	20.7	21.8	22.5	20.5

6. CONCLUSION

In this paper, a cooperative learning method, named WSS is introduced. Also, some criteria to evaluate the expertness of the agents are given. In addition, based on the expertness measures, a suitable weight-assigning formula is devised for the learners who learn from the more expert agents. The method and the measures are tested on hunter–prey and object-pushing problems.

Results show that the strategy-sharing method has no effect (or little effect) on the learning of a multi-agent system when the agents have equal experiences. Agents with different levels of expertness learn better when they implement the WSS method.

SA, G, N and Average Move have a positive effect on the group learning when the learning problem is simple. In the other case (e.g. intelligent-prey case) these expertness measures have a negative impact. PO and NO criteria have completely negative effect, and are not useful for the tested cooperative learning problems.

The introduced criteria are sensitive to the reward, but the Abs criterion has the minimum sensitivity because it has the properties of both P and N measures.

When the sum of the received rewards is greater than the punishments at the beginning of learning, the P criterion is the best among Abs, N, P and N criteria, and N has the worst results. In contrast, when the sum of the received punishments is greater than the rewards, N is the best and P is the worst.

When the difference of the rewards and the punishments is small, the N measure has the worst results. P and N have approximately the same impact, and Abs is the best method.

REFERENCES

1. M. Nili Ahmadabadi and E. Nakano, A 'constrain and move' approach to distributed object manipulation, in: *IEEE Trans. Robotics and Automat.* **17** (2), 157–172 (2001).
2. P. Stone and M. M. Veloso, Multiagent systems: a survey from a machine learning perspective, *Auton. Robot.* **8** (3), 345–383 (2000).
3. M. Tan, Multi-agent reinforcement learning: independent vs. cooperative agents, in: *Machine Learning, Proc. 10th Int. Conf.*, Amherst, MA, pp. 330–337 (1993).
4. A. Samuel, Some studies in machine learning using the game of checkers, *Computer and Thought* (1963).
5. M. Dorigo and L. M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Trans. Evolutionary Comput.* **1**, 53–66 (1997).
6. Y. Kuniyoshi, M. Inaba and H. Inoue, Learning by watching: extracting reusable task knowledge from visual observation of human performance, *IEEE Trans. Robotics Automat.* **10** (6), 799–822 (1994).
7. P. Bakker and Y. Kuniyoshi, Robot see, robot do: an overview of robot imitation, in: *Proc. AISB Workshop on Learning in Robots and Animals*, pp. 3–11 (1996).
8. G. Hayes and J. Demiris, A robot controller using learning by imitation, in: *Proc. 2nd Int. Symp. on Intelligent Robotic Systems*, A. Borkowski and J. L. Crowley (Eds), pp. 198–204, LIFTA-IMAG, Grenoble, France (1994).
9. T. Yamaguchi, Y. Tanaka and M. Yachida, Speed up reinforcement learning between two agents with adaptive mimetism, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Grenoble, France, pp. 594–600 (1997).
10. A. Garland and R. Alterman, Multiagent learning through collective memory, in: *Proc. AAAISS '96*, Stanford, Univ., CA, pp. 33–38 (1996).
11. Y. Liu and X. Yao, A cooperative ensemble learning system, in: *Proc. 1998 IEEE Int. Joint Conf. on Neural Networks (IJCNN '98)*, Anchorage, pp. 2202–2207 (1998).
12. F. J. Provost and D. N. Hennessy, Scaling up: distributed machine learning with cooperation, in: *Proc. AAAI '96*, Menlo Park, CA, pp. 74–79 (1996).
13. R. Maclin and J. W. Shavlic, Creating advice-taking reinforcement learners, *Machine Learning* **22**, 251–282 (1996).
14. H. Friedrich, M. Kaiser, O. Rogalla and R. Dillmann, Learning and communication in multi-agent systems, *Distributed Artificial Intelligence Meets Machine Learning, Lecture Notes in AI* **1221** (1996).
15. C. J. C. H. Watkins, Learning from delayed rewards, PhD Thesis, King's College (1989).
16. C. J. C. H. Watkins and P. Dayan, Q-learning (technical note), *Machine Learning: Special Issue on Reinforcement Learning* **8**, May (1992).

17. N. Meuleau, "Simulating co-evolution with mimetism, in: *Proc. First Eur. Conf. on Artificial Life (ECAL-91)*, pp. 179–184. MIT Press, Cambridge (1991).
18. E. Alpaydin, Techniques for combining multiple learners, in: *Proc. of Engineering of Intelligent Systems '98 Conf.*, E. Alpaydin (Ed.), ICSC Press, Teneriffe, Spain, Vol. 2, pp. 6–12 (1998).
19. C. Claus and C. Boutilier, The dynamics of reinforcement learning in cooperative multiagent systems, in: *Proc. AAAI '97 Workshop on Multiagent Learning*, Providence, pp. 13–18 (1997).

ABOUT THE AUTHORS

Majid Nili Ahmadabadi was born in 1967 and graduated from Sharif University of Technology, Iran in 1990. He received his MSc and PhD degrees in Information Sciences from the Graduate School of Information Science, Tohoku University, Japan in 1994 and 1997 respectively. In 1997, he joined the Advanced Robotics Laboratory at Tohoku University. Later he moved to the Department of Electrical and Computer Engineering, Faculty of Engineering, University of Tehran where he is the Head of the Robotics and AI Laboratory. He is also a Senior Researcher at the Institute for Studies on Theoretical Physics and Mathematics. He initialized the Iranian National Robot Contests in 1999 and is the President of the Executive Committee of these games. His main research interests are distributed robotics and artificial intelligence, mobile robots, and cooperative learning in multi-agent systems.

Masoud Asadpour was born in Iran in 1975 and received his BSc in Computer Software Engineering from Sharif University of Technology, Iran in 1977. He received his MSc in AI and Robotics from the University of Tehran, Iran in 1999. He is a Researcher at the Institute for studies on Theoretical Physics and Mathematics. His research interests are cooperative learning, cooperative robotics and multi-agent systems.

Eiji Nakano was born in 1942, graduated from the University of Tokyo in 1965 and finished his graduate course in 1970. In the same year, he joined the Mechanical Engineering Laboratory of the Ministry of International Trade and Industry, and started research into robotics. In 1987, he moved to Tohoku University as a Professor in the Faculty of Engineering. In 1993, he moved to the Graduate School of Information Sciences of the same university where he is the Head of the Advanced Robotics Laboratory. He founded the Robotics Society of Japan and acted as its Director from 1983 until 1987. He was a Director of the Society of Instrument and Control Engineers in 1988–89. He initialized the Intelligent Robot Contest in 1989 and is the President of the Executive Committee of these games. He has been the Director of the Society of Biomechanism (1995–97), the Chairman of the Committee of the Development of Forestry Robot in the Forestry Ministry of Japan (1990–), the Chairman of the Board of Directors of the Central Committee of International Robot Games Festival (1999–), and the President of the Research and Development Consortium of Intelligent Patient Care System (1999–). His main research interests are office messenger robots, the application of intelligent robot technologies for the disabled, mobile robots for rough terrains, omni-directional mobile robots, multiple cooperative robot systems, self-adjustable manipulators, and practical visual sensing systems.