Cooperative Q-learning: The Knowledge Sharing Issue

Majid Nili Ahmadabadi * **Masoud Asadpour *** **Eiji Nakano****

*Robotics and AI Laboratory, Dept. of Elect. and Comp. Eng., Faculty of Engineering, University of Tehran

*School of Intelligent Systems, Institute for Studies on Theoretical Physics and Mathematics

**Advanced Robotics Lab., GSIS, Tohoku University, Japan

E-mail: nily@sofe.ece.ut.ac.ir, asadpur@karun.ipm.ac.ir, majid,nakano@robotics.is.tohoku.ac.jp

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 given. 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 [12] [15]. Having these attributes, multi-agent systems are used to solve complicated problems, search in large domains, execute sophisticated tasks, and make more faults tolerant and reliable systems.

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

In approximately all of the present multi-agent teams, agents learn individually. However, 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 some advices from them. Agents can also share their information and learn from this data. Namely, the agent can cooperate in learning. Figure 1 shows an example of such system.

Cooperative learning can be observed in human and some animal societies. Because 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 [15]. Researchers have shown improvement in the learning even when simple cooperative learning methods are used [16].

Heretofore, cooperative learning has not been deeply 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 of the others according to their relative expertness.

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

Related researches are 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. 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 [14] 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 weak strategies of the learner. In Ant Colony System [4], some ants learn to solve the Travelling Salesman Problem by non-verbal communication through the pheromones on the edges of a graph.

Imitation [8] 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 [2]. For example in [8], a robot perceives a human doing a simple assembly task and learns to reproduce it again in different environments. Hayes and Demiris [7] 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 and others [19] 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, Inter-Mimetism Problem is occurred 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 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 [16]. Episode Sharing can be used to communicate the (state, action, reward) triples between the Reinforcement Learners [16]. Tan showed that, sharing episodes with an expert agent could significantly improve the group learning [16].
In Collective Memory method, the learners put the learnt strategy or experienced episodes on a shared memory [6] or have a unique memory and update it cooperatively [16].

A Cooperative Ensemble Learning System [9] 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 [13] 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[10] used Advice Taking scheme to help a Connectionist Reinforcement Learner. The Learner accepts advices 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 researches, cooperation is uni-directional between a pre-defined trainer and a pre-specified learner agent. But in the real world, all of the agents may learn something from each other; even from the non-experts.

In Strategy Sharing Method [16], members of a multi-agent system learn from all of the agents. The agents learn individually by 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 the environment changes [19].

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 Weighted Strategy-Sharing Method

In Weighted Strategy-Sharing 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 make the Hidden State Problem [5]).

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 gents are in Individual Learning Mode. Agent *i* executes the t_i learning trials, based on 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 Cooperation Time) all agents stop Individual Learning and switch to Cooperative Learning Mode.

In Cooperative Learning Mode, each learner assigns some weights to the other agents according to their expertness values[1]. Then, it takes a weighted average[2] of the others' Q-tables and uses the resulted table as its new Q-table.

3-1 Individual Learning based on Reinforcement Learning

In this paper, One-Step Q-Learning is used for Individual Learning Mode.

Reinforcement learning is one of the simplest and widely used learning method and has many similarities to the 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 One-Step Q-Learning algorithm [17][18] 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):

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

where τ is the temperature parameter and adjusts 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):

$$Q(x,a) \leftarrow (1-\beta)Q(x,a) + \beta\,(r + \gamma\, V(y)) \quad (2)$$

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

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

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

---

[1] Expertness measures are introduced in section 3.2.

[2] In Algorithm 1, multiplication (*) and Summation (+) operators must be specified according to the knowledge representation method. For example, in One-Step Q-Learning method, * is the scalar matrix multiplication operator and + represents the matrix summation.

3-2 Measuring the Expertness

In Weighted Strategy-Sharing 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 the 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 of the researches have a pre-specified expert agent(s). For example, in [16], 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 strategy sharing method [16], expertness of the agents are assumed to be equal. Nicolas Meuleau [11] used the user judgment for specifying the expert agent. This method requires continuous supervision by a human being.

In [1] different but fixed expertness values are assumed for the agents. Difference 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. [19] 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, the 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 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 in the beginning of the learning process –that is not faced many failures- is less than those learned for a longer time and have naturally faced more failures.

3-2-1 Some Experness Evaluation Criterion

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

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

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

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

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

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

6-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:

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

where $0 \le \alpha_i \le 1$ is **Impressibility Factor** and shows how much agent *i* relies on the others knowledge. e_i, e_j, and n 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:

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

$$Exprt(i) = \{ j \mid e_j > e_i \}$$

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 Positive Only, agents send only positive-value cells of their Q-tables to others and, the expertness of the agents are measured by Positive criterion. In Negative Only, agents communicate only negative-value cells of their Q-tables to others and the expertness are measured by Negative criterion.

4 Simulation Results on Hunter-Prey Problem

"Hunter and Prey" problem [16] 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 3 hunters, those search independently in a 10×10 environment to capture a prey agent. Hunters can move with speed between 0 and 1 and the prey can move with 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 3 is set for the prey. Greater visual field helps the prey to escape before the hunter see 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 unit, and 45 degrees respectively.

For complicating 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 the other researches, 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, four walls around the environment, the prey, and the hunter are assumed to be electropositive materials. Therefore, the prey selects the path with minimum potential. Repulsive force of the hunter is considered 1.5 times 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 figure 2.

In an environment with the Intelligent Prey, each hunter's movements may affect the prey. So, if several hunters learn together in an environment, 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 6 trials, then the second one is permitted to do 3 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 the punishment signals are one of 6 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 at a random situation. The One-Step Q-learning parameters is set to $\beta = 0.01$, $\gamma = 0.9$, and T=0.4. Q-table values have initialized to zero and all agents had $\alpha_i = 0.7$. Also, for trial n, the average number of the hunter actions to capture the prey over past n trials is calculated.

4-1 Equal Experiences

In figures 3 and 4, the average number of moves using six mentioned reinforcement values and the introduced expertness measuring criteria are shown for individual and cooperative learning methods. Also, improvements gained in learning over Independent Learning method are given in table 1.

All of the cooperative learning methods (except Positive Only and Negative Only) have approximately the same results as Independent Learning (Ind). Positive Only and Negative Only 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. So, the probability of selecting an inefficient action rises. Negative Only is worse than Positive Only, because even significant difference of two negative-value actions makes little change in the selection probability of the actions. But a little difference of two positive-value actions can raise the probability of choosing the better action significantly.

4-2 Different Experiences

In figures 5 and 6, average number of moves for the 6 described reinforcement values are shown for each cooperation method. Also improvement percent of each method is given in table 2.

Table 2 shows that, Absolute, Positive, and Negative are resulted in improvement in all cases. In the random-prey case, where the number of punishments is less than in the intelligent-prey simulator, Positive criterion has the best results. But, in intelligent-prey case, Negative criterion works better.

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

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 little, even when they have considerably different number of moves.

Negative Only has negative effect on learning in both cases and Positive Only causes no improvement. Simple Averaging (SA) has 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 figure 7. Circles show agents; agent 1 is the intelligent prey and agents 2 and 3 are the hunters using Positive 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 (figure 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. Pushing velocity is slow and the inertia forces are neglected. Therefore, considering figure 8, linear (a) and angular (α) accelerations can be computed as:

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

$$\alpha = \frac{F_1 . r . \sin(\theta_1) - F_2 . r . \sin(\theta_2)}{I} \qquad (7)$$

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

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(deg), respectively. Environment is 10×10 and since x and y can be between 10 and -10, the robots have 8×20×20=3200 states. Actions of the robots are specified based on the amount and angle of the forces.

Each force is applied 4 seconds and the robots follow the object until it stops. 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 occur after each 100 learning trials. 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 figure 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, the Simple Averaging has little positive effect on the learning, compared to the Independent Learning. Also in all cases, Weighted Strategy Sharing based on Normal, Absolute, Positive, and Negative criteria have better results than Simple Averaging. When the punishments are greater than the rewards at the beginning of learning, Negative is the best criterion and Positive is the worst one among the four above mentioned expertness measures. But, in the case where the rewards are greater than the punishments, Positive is the best criterion and Negative is the worst one. Also when the punishments and the rewards are approximately equal, Absolute has the best results. Since any little difference in the rewards and the punishments significantly affects the weights in this case, the Normal criterion has the worst results. Naturally, Positive and Negative have approximately the same results in this case.

6 Conclusion

In this paper, a cooperative learning method, named **Weighted Strategy-Sharing (WSS),** is introduced. Also, some criteria to evaluate 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.

Simple Averaging, Gradient, Normal and Average Move have 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 negative impact. Positive Only and Negative Only criteria have completely negative effect and are not useful for the tested cooperative learning problems.

The introduced criteria are sensitive to the reward but Absolute criterion has the minimum sensitivity. Because, it has the properties of both Positive and Negative measures.

When sum of the received rewards is greater than the punishments in the beginning of learning, Positive criterion is the best among Absolute, Normal, Positive, and Negative criteria and Negative has the worst results. In contrast, when sum of the received punishments is greater than the rewards, Negative is the best and Positive is the worst.
When difference of the rewards and the punishments is little, Normal measure has the worst results. Positive and Negative has approximately the same impact, and Absolute is the best method.

References

[1] Alpaydin E., *"Techniques for Combining Multiple Learners,"* In: Alpaydin E.(ed.), Proceedings of Engineering of Intelligent Systems'98 Conference, ICSC Press, Vol.2, pp.6-12, 1998.

[2] Bakker P. and Kuniyoshi Y., *"Robot See, Robot Do: An Overview of Robot Imitation,"* The Proceedings of the AISB workshop on Learning in Robots and Animals, pp. 3-11, 1996.

[3] Claus C. and Boutilier C., *"The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems,"* AAAI'97 workshop on Multiagent Learning, 1997.

[4] Dorigo M. and Gambardella L.M., "*Ant Colony System:A Cooperative Learning Approach to the Traveling Salesman Problem,*" IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, April 1997.

[5] Friedrich H., Kaiser M., Rogalla O., and Dillmann R., *"Learning and Communication in Multi-Agent Systems,"* Distributed Artificial Intelligence Meets Machine Learning, Lecture notes in AI, Vol.1221, 1996.

[6] Garland A. and Alterman R., *"Multiagent Learning through Collective Memory,"* AAAISS'96, 1996.

[7] Hayes G. and Demiris J., *"A Robot controller using learning by imitation,"* In: Borkowski A., Crowley J.L.(eds.), Proceeding of the 2nd International Symposium on Intelligent Robotic Systems, pp.198-204, Grenoble, France: LIFTA-IMAG, 1994.

[8] Kuniyoshi Y., Inaba M., and Inoue H., *"Learning by watching: Extracting reusable task knowledge from visual observation of human performance,"* IEEE Transaction on Robotics and Automation, Vol.10, No.6, pp.799-822, 1994.

[9] Liu Y. and Yao X., "*A Cooperative Ensemble Learning System,*" Proceeding of the 1998 IEEE International Joint Conference on Neural Networks (IJCNN'98), Anchorage, USA, pp.2202-2207, May 1998.

[10] Maclin R. and Shavlic J.W., *"Creating Advice-Taking Reinforcement Learners",* Machine Learning, Vol.22, pp.251-282, 1996.

[11] Meuleau, N., *"Simulating Co-Evolution with Mimetism,"* Proc. of the First European Conf. on Artificial Life(ECAL-91), pp.179-184, 1991.

[12] Nili Ahmadabadi M. and Nakano E., *"A "Constrain and Move" Approach to Distributed Object Manipulation,"* Accepted in IEEE Transaction on Robotics and Automation, 2001.

[13] Provost F.J. and Hennessy D.N., *"Scaling Up: Distributed Machine Learning with Cooperation,"* AAAI'96, 1996.

[14] Samuel A., *"Some Studies in Machine Learning Using the Game of Checkers,"* Computer and Thought, 1963.

[15] Stone P. and Veloso M.M., *"Multiagent Systems: A Survey from a Machine Learning Perspective,"* Autonomous Robotics, Vol.8, No.3, July 2000.

[16] Tan, M., *"Multi-agent reinforcement learning: independent vs. cooperative agents",* In:Machine Learning, Proceedings of the 10th International Conference, Amherst, Massachusetts, 1993.

[17] Watkins C. J.C.H., *Learning from Delayed Rewards,* Ph.D. thesis, King's College, May 1989.

[18] Watkins C.J.C.H. and Dayan P., "*Q-Learning (technical note)*," In: Sutton R.S.(ed.), "*Machine Learning: Special issue on reinforcement learning*", Vol. 8, May 1992.

[19] Yamaguchi T., Tanaka Y., and Yachida M., "*Speed up Reinforcement Learning between two Agents with Adaptive Mimetism,*" Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'97), pp. 594-600, 1997.

List of Algorithms

List of Figures

List of tables

Algorithm 1-Weighted Strategy Sharing 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) $V(y_i) := Max_{b \in actions} Q(y_i, b)$
        (12) $Q_i^{new}(x_i, a_i) := (1-\beta_i) Q_i^{old}(x_i, a_i) + \beta_i \ (r_i + \gamma_i \ V(y_i))$
        (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) $Q_i^{new} := 0$
        (20) for j:= 1 to n do
        (21) begin
                (22) $W_{ij}$ := ComputeWeights(i,j,$e_1$...$e_n$)
            (23) $Q_j^{old}$ := GetQ($A_j$)
            (24) $Q_i^{new} := Q_i^{new} + W_{ij} * Q_j^{old}$
        (25) end
    (26) end
(27) end
```

Table 1-Improvement Percents in equal experience case.

	SA	Nrm	Abs	P	N	PO	NO	G	AM
random-prey	-0.047	-1.36	-0.141	0.375	0.235	-4.972	-23.358	0.704	2.111
intelligent-prey	-3.136	-3.339	0.936	-1.95	-0.156	-17.663	-65.096	-1.264	-0.452

Table 2- Improvement Percents in different experience case.

	SA	Nrm	Abs	P	N	PO	NO	G	AM
random-prey	-2.735	-15.307	7.582	9.933	8.301	0.845	-5.777	-7.486	-6.286
intelligent-prey	-7.572	-51.153	13.9	14.244	14.44	-1.676	-44.841	-47.49	-0.654

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

reward, punishment	Independent	Simple Averaging	Normal	Absolute	Positive	Negative
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

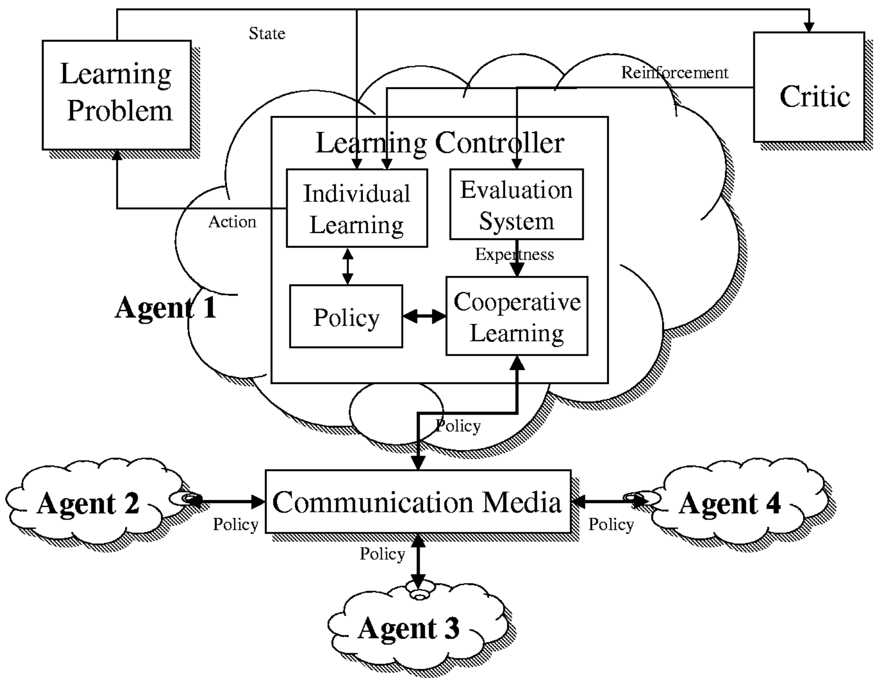

Figure 1- A Cooperative Learning System

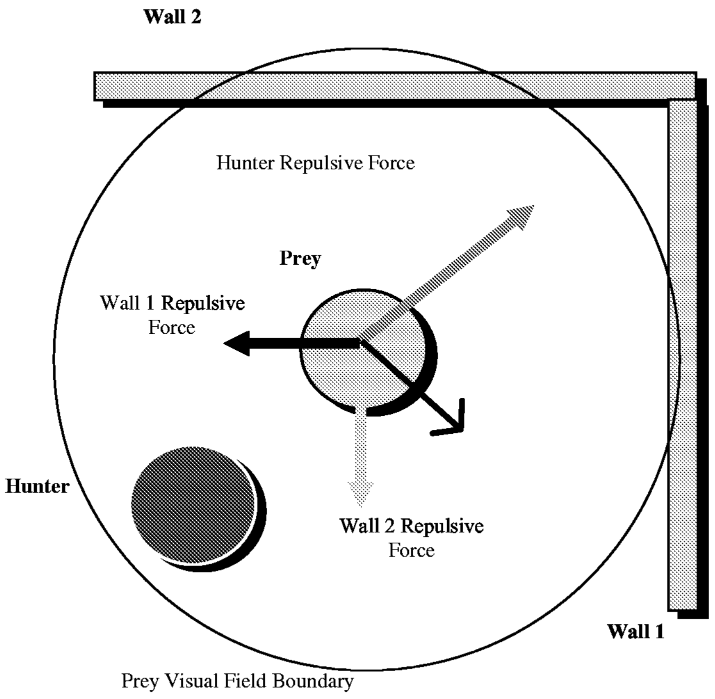

Figure 2-An example of computing the resultant force

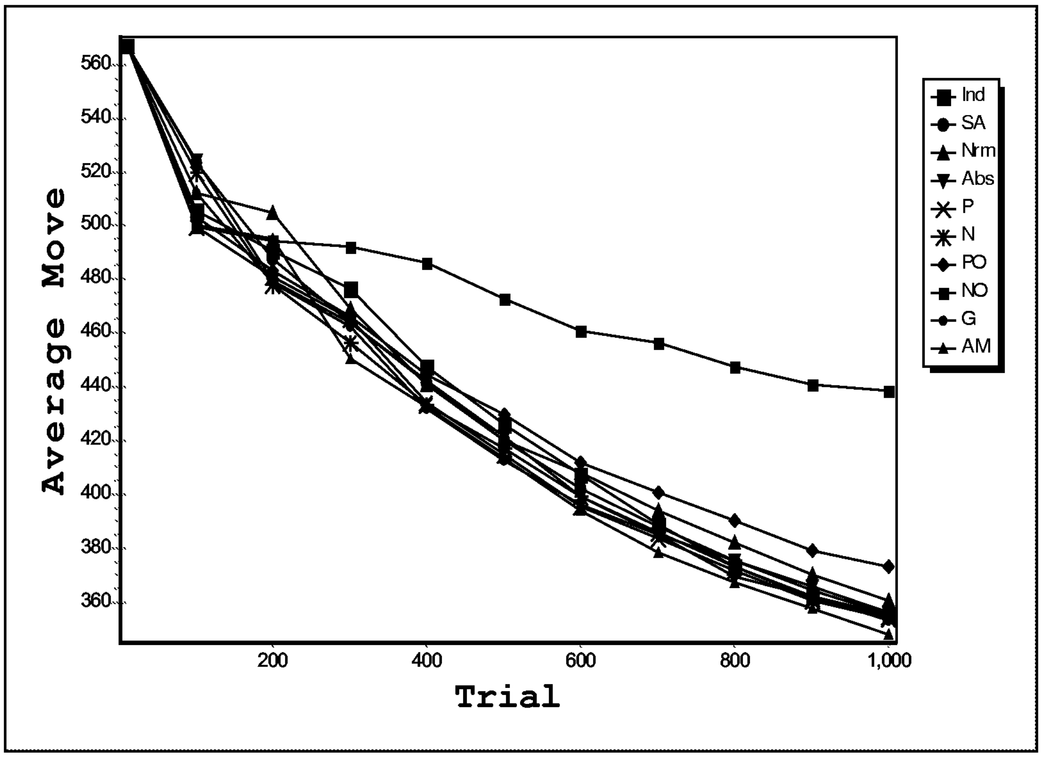

Figure 3-Average number of moves in random-prey and equal experience case

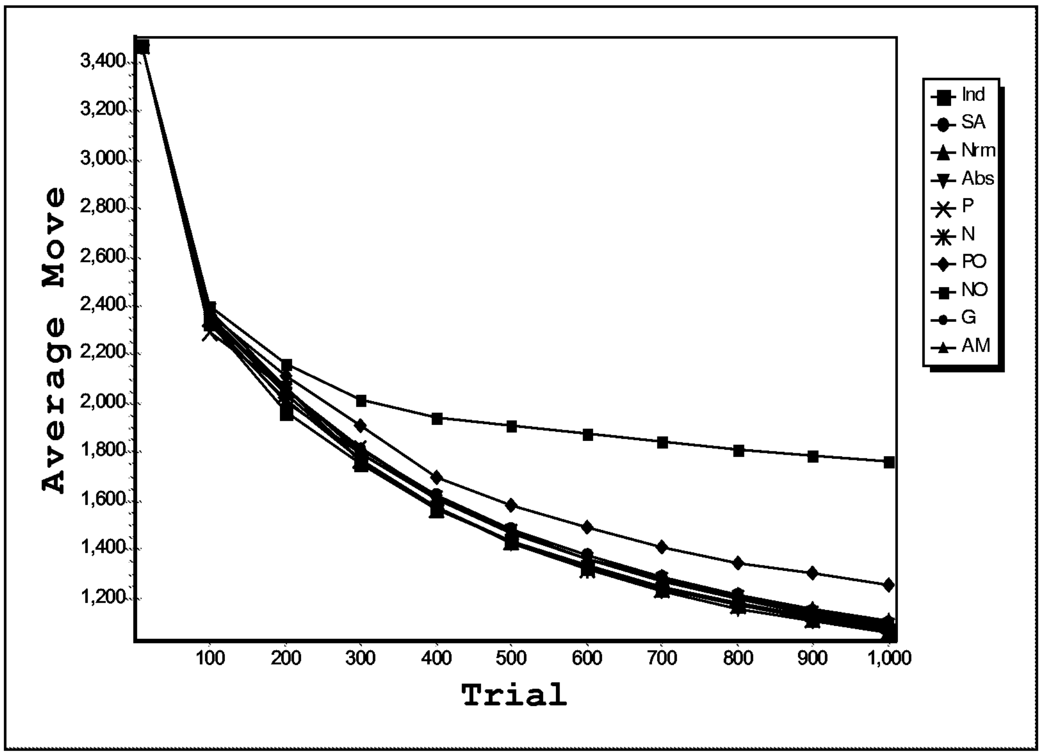

Figure 4-Average number of moves in intelligent-prey and equal experience case

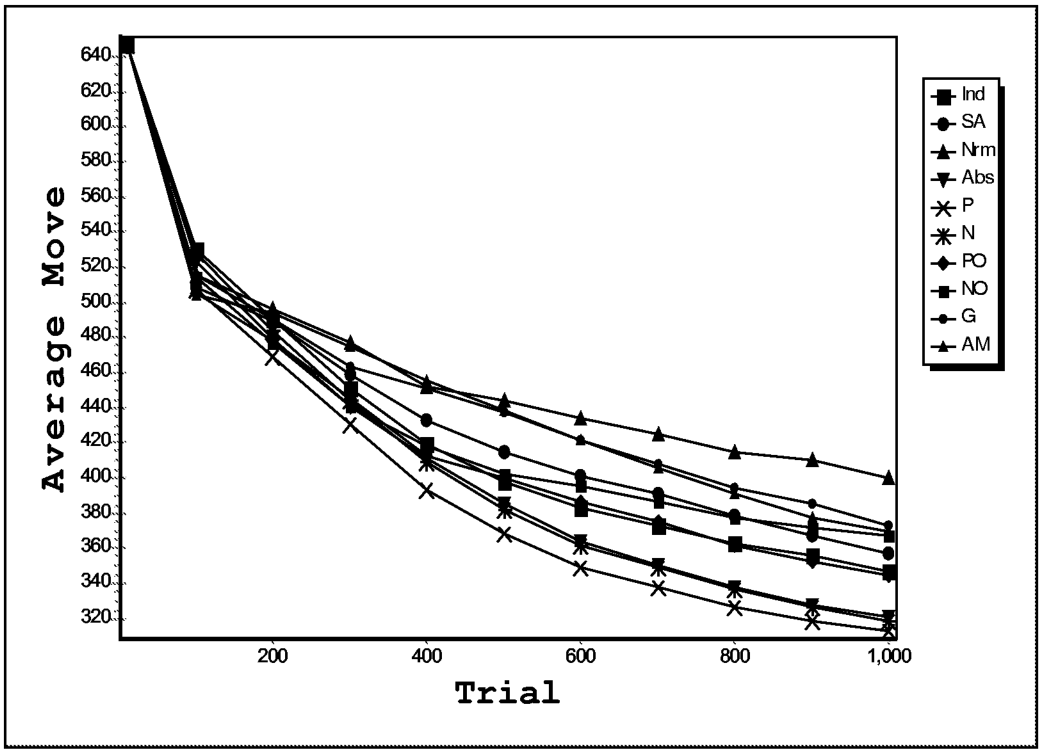

Figure 5-Average number of moves in random-prey and different experience case

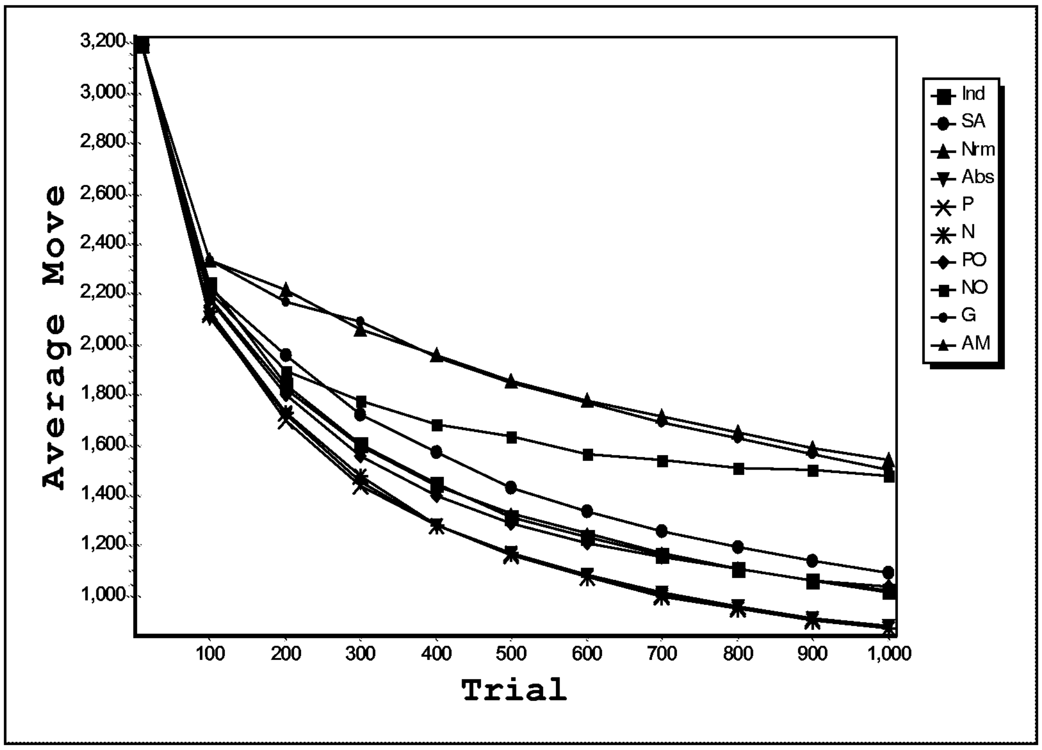

Figure 6-Average number of moves in random-prey and different experience case

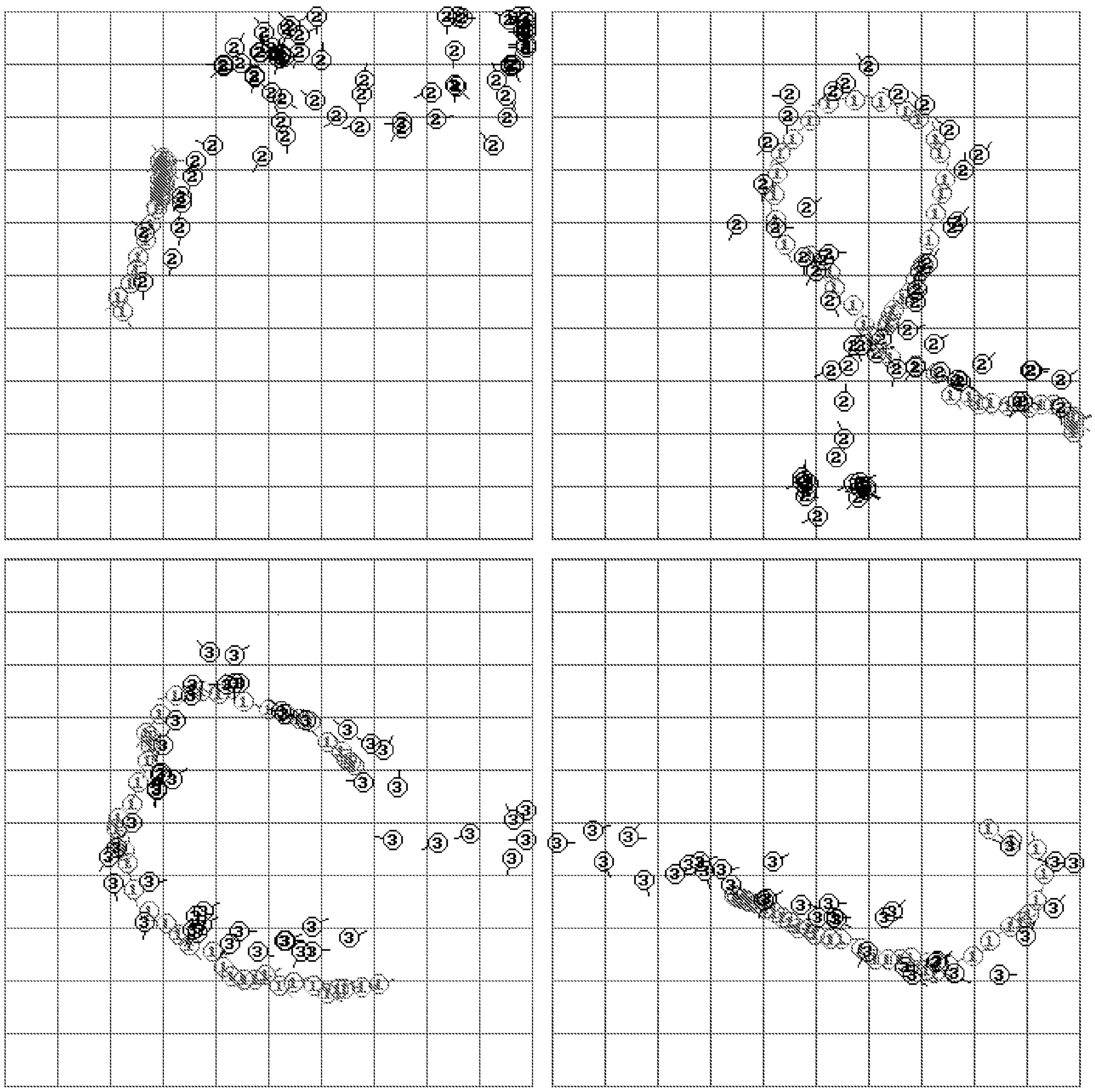

Figure 7- Four samples of hunting the intelligent prey. Circles show agents; agent 1 is the intelligent prey and agents 2 and 3 are the hunters using Positive expertness measure

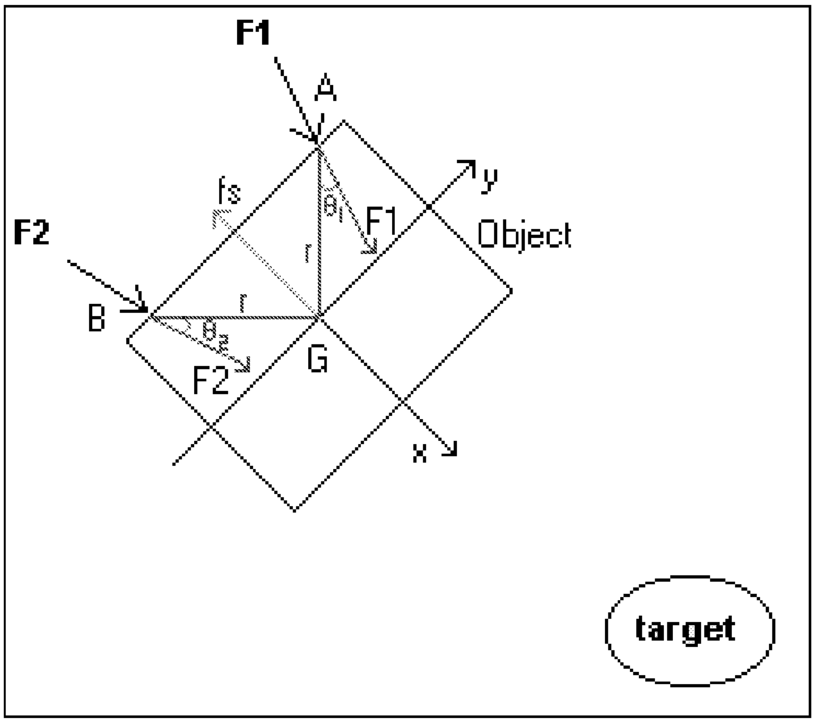

Figure 8- The object pushing model

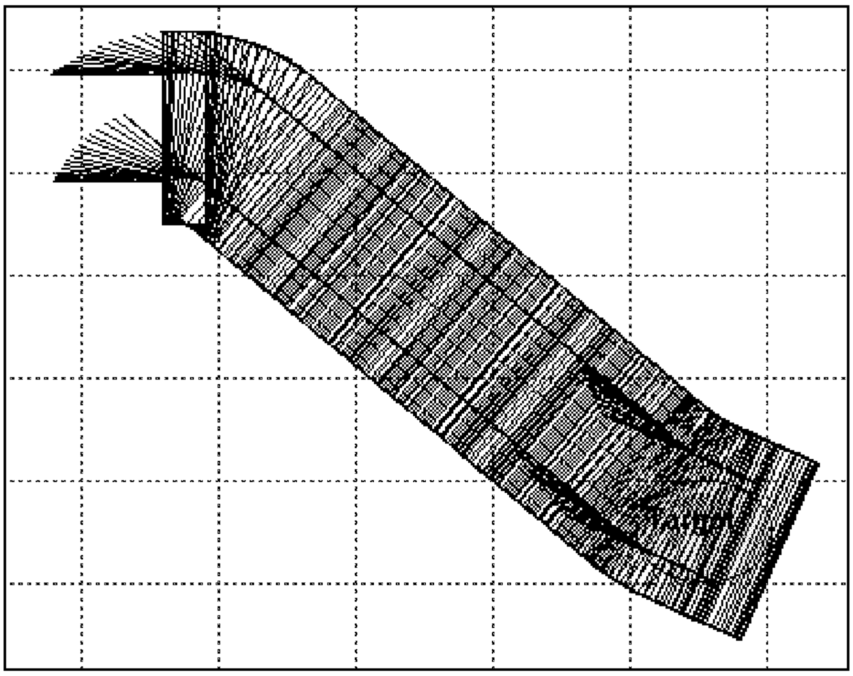

Figure 9- A sample of the learned object pushing path