Implementation of Multiagent Learning Algorithms for Improved Decision Making

Deepak A. Vidhate[#1], Dr. Parag Kulkarni[*2]

[#]*Research Scholar, Department of Computer Engineering, College of Engineering, Pune, Maharashtra, India*
[*]*Director, EKLaT Research, Shivajinagar, Pune, Maharashtra, India*

***Abstract* — *The output of the system is a sequence of actions in some applications. There is no such measure as the best action in any in-between state; an action is excellent if it is part of a good policy. A single action is not important; the policy is important that is the sequence of correct actions to reach the goal. In such a case, machine learning program should be able to assess the goodness of policies and learn from past good action sequences to be able to generate a policy. A multi-agent environment is one in which there is more than one agent, where they interact with one another, and further, where there are restrictions on that environment such that agents may not at any given time know everything about the world that other agents know. Two features of multi-agent learning which establish its study as a separate field from ordinary machine learning. Parallelism, scalability, simpler construction and cost effectiveness are main characteristics of multi-agent systems. Multiagent learning model is given in this paper. Two multiagent learning algorithms i. e. Strategy Sharing & Joint Rewards algorithm are implemented. In Strategy Sharing algorithm simple averaging of Q tables is taken. Each Q-learning agent learns from all of its teammates by taking the average of Q-tables. Joint reward learning algorithm combines the Q learning with the idea of joint rewards. Paper shows result and performance comparison of the two multiagent learning algorithms.*

Keywords — *Joint Rewards, Multiagent, Q-Learning, Reinforcement Learning, Strategy Sharing*

I. Introduction

Consider the example market chain that has hundreds of stores all over a country selling thousands of goods to millions of customers. The point of sale terminals record the details of each transaction i.e. date, customer identification code, goods bought and their amount, total money spent and so forth. This typically generates gigabytes of data every day. What the mark et chain wants is to be able to predict who are the likely customers for a product. Again, the algorithm for this is not evident; it changes over time and by geographic location. If stored data is analyzed and turned into information then it becomes useful so that we can make use of an example to make predictions. We do not know exactly which people are likely to buy this product, or another product. We would not need any analysis of the data if we know it already. But because we do not know, we can only collect data and hope to extract the answers to questions from data.

We do believe that there is a process that explains the data we observe. Though we do not know the details of the process underlying the generation of data – for example, customer behavior - we know that it is not completely random. People do not go to markets and buy things at random. When they buy beer, they buy chips; they buy ice cream in summer and spices for Wine in winter. There are certain patterns in the data. We may not be able to recognize the process completely, but still we can construct a *good and useful approximation*. That approximation may not explain everything, but may still be able to account for some part of the data. Though identifying the complete process may not be possible, but still patterns or regularities can be detected.

Such patterns may help us to understand the process, or make predictions. Assuming that the near future will not be much different from the past and future predictions can also be expected to be right.

There are many real world problems that involve more than one entity for maximization of an outcome. For example, consider a scenario of retail shops in which shop A sales clothes, shop B sales jewelry, shop C sales footwear and wedding house D. In order to build a single system to automate (certain aspects of) the marketing process, the internals of all shops A, B, C, and D can be modeled. The only feasible solution is to allow the various stores to create their own policies that accurately represent their goals and interests. They must then be combined into the system with the aid of some of the techniques. The goal of each shop is to maximize the profit by an increase in sale i.e. yield maximization. Different parameters need to be considered in this: variation in seasons, the dependency of items, special schemes, discount, market conditions etc. Different shops can cooperate with each other for yield maximization in different situations. Several independent tasks that can be handled by separate agents could benefit from cooperative nature of agents.

Another example of a domain that requires cooperative learning is hospital scheduling. It requires different agents to represent the regard of different people within the hospital. Hospital employees have a different outlook. X-ray operators may want to maximize the throughput on their machines. Nurses in the hospital may want to minimize the patient’s time in the hospital. Since different people examine candidate with different criteria, they must be represented by cooperative agents. The output of the system is a sequence of actions in some applications. There is no such measure as the best action in any in-between state; an action is excellent if it is part of a good policy. A single action is not important; the policy is important that is the sequence of correct actions to reach the goal. To be able to generate a policy the machine learning programs should able to assess the quality of policies and learn from past good action sequences.

This paper is organized as: Section II gives the concept of multiagent agent learning, Section III describes multiagent model. Strategy sharing algorithm is given in Section IV and Joint Rewards algorithm is given in section V. Section VI gives experimental setup and Section VII put up the result comparisons of all both algorithms with final concluding remark and future scope.

II. MULTI AGENT LEARNING

An agent is a computational mechanism that reveals a high degree of autonomy. Based on information received from the environment, the agent performs actions in its environment. A multi-agent environment is one in which there is more than one agent, where they interact with one another, and further, where there are restrictions on that environment such that agents may not at any given time know everything about the world that other agents know[1]. Two features of multi-agent learning which establish its study as a separate field from ordinary machine learning. First, because multi-agent learning addresses the problem domains involving multiple agents. The search space considered is extraordinarily huge. Small changes in learned behaviours can often result in random changes in the resultant macro-level properties of the multi-agent group as a whole due to the communication of those agents. Second, multi-agent learning involves multiple learners, each learning and adapting in the context of others; this introduces complex issues to the learning process which are not yet fully understood[2].

Parallelism, scalability, simpler construction and cost effectiveness are main characteristics of multi-agent systems. Having these qualities, multiagent systems are used to resolve complex problems, search in large domains, execute sophisticated tasks, and make more fault-tolerant and reliable systems. In most of the existing systems, agents’ behaviour and coordination schemes are designed and fixed by the designer. But, an agent with incomplete and fixed knowledge and behaviour cannot be adequately efficient in a dynamic, complex or changing environment. Therefore, to have all benefits of applying a multi-agent system, agent team must learn to manage the fresh, hidden and dynamic situations[3].

In approximately all of the present multi-agent teams, agents learn independently. Agents are not required to learn all things from their own experiences. Each agent observes the others and learns from their situation and behaviour. Also, agents can check with more expert agents or get guidance from them. Agents can also share their information and learn from this information, i.e. the agent can cooperate in learning.

In the single-agent system, only one agent interacts with the environment. Multiagent system (MAS) consists of multiple agents. These agents all carry out actions and control their environment. Each agent selects actions individually, but it is the resulting joint action which manipulates the environment and generates the reward for the agents. This leads to severe consequences on the characteristics and the complexity of the problem. Work focused on cooperative MASs in which the agents have to optimize a shared performance measure[4].

III. MULTI-AGENT MODEL

Parameters for Multiagent Model

General model parameters for MAS are described. Most model parameters extend the parameters from single-agent systems[5]. A

Multiagent System can be described using the following model parameters:

- A discrete time step t = 0, 1, 2, 3, ...
- A group of n agents A = { A1, A2, . . . ,An }.
- A finite set of environment states S. A state $s^t \epsilon$ S describes the state of the system at time step t.
- A finite set of actions A_i for every agent i. The action selected by agent i, Ai at time step t is denoted by $a^t \epsilon$ A. The joint action a ∈ A = A1 × . . . × An is the vector of all individual actions.
- A reward function R : S × A → R which provides the agent i with a reward $r^{t+1} \in R(s^t, a^t)$ based on the joint action a^t taken in state s^t.
- A state transition function T: S×A×S → [0, 1] which gives the transition probability $p(s^t|a^{t-1}, s^{t-1})$ that the system moves to state s^t when the joint action a^{t-1} is performed in state s^{t-1}.

These parameters are very similar to the ones in the single-agent case. However, difficulties arise due to the decentralized nature of the problem. Each agent selects actions independently, but it is the resulting joint action that manipulates the environment and produces the reward.

Stochastic games (SGs)

Stochastic games are a very natural extension of MDPs to multiple agents.

Definition 1: *A stochastic game is a tuple (n, S, A1…n ,T, R1…n),where n is the number of agents, S is a set of states, A_i is the set of actions available to agent i with A being the joint action space A1×…×An, T is a transition function S×A×S→[0,1], and R_i is a reward function for the i^{th} agent S×A→R. SGs are a very natural extension of MDPs to multiple agents[6].*

Model of Multiagent Q-learning

Model of multiagent Q-learning with ε-greedy exploration is presented here. Effect of ε -greedy mechanism and the presence of other agents on learning the process of one agent are studied to develop the model. The derivation of a continuous time equation for the Q-learning rule is firstly demonstrated. Then the limits of this equation for the case of a single learner are analysed. It indicates how they change dynamically when multiple learners are considered. Finally, it is proved that the ε-greedy mechanism affects the shape of the modeled function[7].

Consider the situation composed of 2 agents with 2 actions each and a single state. The reward functions of the agents, in this case can be described using tables of the form:

$$A=\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} \qquad B=\begin{bmatrix} b11 & b12 \\ b21 & b22 \end{bmatrix}$$

where A describes the rewards, for the first agent and B the rewards for the second agent. Q-learning update rule can be simplified for only one state as:

$Q_{ai} := Q_{ai} + \alpha(r_{ai} - Q_{ai})$……………….........……..(1)

where

Q_{ai} → Q-value of agent a for action i

r_{ai} → reward that agent a receives for executing action i.

a → agent and i → action

Analysis

The update rule for the first agent can be rewritten as:

$Q_{ai}(k + 1) - Q_{ai}(k) = \alpha(r_{ai}(k + 1) - Q_{ai}(k))$…….....(2)

This difference equation explains the absolute growth in Q_{ai} between times k and k + 1. To obtain its continuous time version, consider Δt ∈ [0, 1] to be a small amount of time.

$Q_{ai}(k + \Delta t) - Q_{ai}(k) \approx \Delta t \times \alpha(r_{ai}(k+\Delta t) - Q_{ai}(k))$..(3)

to be the approximate growth in Q_{ai} during Δt.

if Δt = 0 then the equation becomes identity.

If Δt = 1 then it becomes $Q_{ai}(k + 1) - Q_{ai}(k) = \alpha(r_{ai}(k + 1) - Q_{ai}(k))$ i.e. equation 5.

If Δt = {0,1} then it becomes linear approximation.

Divide both sides of the equation by Δt

$\frac{Q_{ai}(k+\Delta t) - Q_{ai}(k)}{\Delta t} \approx \alpha(r_{ai}(k+\Delta t) - Q_{ai}(k))$………...(4)

taking limits Δt→0 on both sides

$\lim_{\Delta t\to 0}\frac{Q_{ai}(k+\Delta t) - Q_{ai}(k)}{\Delta t} \approx \lim_{\Delta t\to 0} \alpha (r_{ai}(k + \Delta t) - Q_{ai}(k))$

$\lim_{\Delta t\to 0}\frac{Q_{ai}(k+\Delta t) - Q_{ai}(k)}{\Delta t} \approx \alpha (r_{ai}(k) - Q_{ai}(k))$

$\frac{dQ_{ai}(k)}{dt} \approx \alpha (r_{ai}(k) - Q_{ai}(k))$………. ……….....…(5)

It is an approximation for the continuous time version of Equation 2.

Solution to this equation found by integration as

$Q_{ai}(k)=Ce^{-\alpha t} + r_{ai}$…………...…………………......(6)

where C is the constant of integration.

As e^{-x} is a monotonic function and $\lim x\to\infty e^{-x}=0$, it is easy to observe that the limit of Equation 3 when t→ is r_{ai}:

$\lim_{t\to\infty} Q_{ai}(K) = \lim_{t\to\infty} Ce^{-\alpha t} + \lim_{t\to\infty} r_{ai} = r_{ai}$….....……..(7)

It is considered that only the first agent is learning and that the second agent is using a pure strategy then it will always generate the same reward for the first agent. In this case, the derivation above is sufficient to prove that Q_{ai} will monotonically increase or decrease towards r_{ai}, for any initial value of Q_{ai}. More particularly, the function is monotonically increasing if Q_{ai} (0) < r_{ai} and monotonically decreasing if Q_{ai} (0) > r_{ai}. The model build here will be used to determine how one learning strategy affects the action selection of another learning agent.

IV. STRATEGY SHARING ALGORITHM

One way for knowledge sharing in multi-agents is knowledge averaging. We can divide averaging in two general categories, simple averaging and weighted averaging. In simple averaging, which is called strategy sharing, each *Q*-learning agent learns from all of its teammates by taking the average of *Q*-tables[8].

$Q_i^{new} = \frac{1}{n}\sum_{j=1}^{n} Q_j^{old}$……………………………(8)

The SS treats all agents similarly, ignoring their level of knowledge. This method does not consider agents' different expertise levels. SS algorithm is described as:

Algorithm 1 : Strategy Sharing Algorithm

```
1.  initialize
2.  while not EndOfLearning do
3.  begin
    4.  if InIndividualLearning Mode 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. end
    14. else {Multiagent Learning}
    15. begin
        16. for j := 1 to n do
        17. $Q_i^{new}$ := 0
        18. for j := 1 to  n do
        19. begin
            20. $Q_i^{new} := Q_i^{new} + 1/n \sum Q_j^{old}$
                // strategy sharing by simple averaging of Q tables
        21. end
```

V. JOINT REWARDS ALGORITHM

To extend Q-learning into a multi-agent system, two main challenges exist. One is how to deal with the huge Q-table with the increment of dimension in a multi-agent system. The other challenge is how to utilize the cooperative behaviour among agents to obtain more efficient learning results. Joint rewards ensure agents to learn in a multiagent environment. The experiment results show the efficiency and well convergence of the algorithm. It is not enough for each agent to proceed selfishly in order to reach a globally optimal strategy in a multi-agent environment. It is not possible to accomplish with only one agent to accomplish a task.

The section presents a joint reward learning algorithm which combined the Q-learning with the idea of joint rewards to meet above two challenges partly. In a multi-agent system, every agent needs to maintain a Q table which contains the information about its and others agent' states and actions, i.e. the situation of the whole environment. In order to encourage cooperative behaviour among agents to get global optimal rewards, it should take account of the other agents' actions. Here, we use a simplified of vicarious rewards for feasible to realize. We call 'joint reward'[9],

$jr_i = b.pr_i + (1 - b).\frac{_{1}\, pr_j}{m}$ …………………(9)

where *pr* is the personal reward of agent *i*, and $^{n}_{=1}\, pr_j$ the sum of rewards of other agents except i and 0 < *b* ≤ 1 is the personal weight, denoting how much importance is given to agent's personal reward compared to that of other agents. The improved update rule of Q-learning values of agent *i* can be formulated as

$Q_i^{new}(x_i, a_i) := (1 - \alpha)\, Q_i^{Old}(x_i, a_i) + \alpha\, (jr_i + \beta \max_a Q(s, a))$……………………………………………………(10)

Algorithm 2 : Multiagent learning using Joint Reward Algorithm

```
1. For each agent i (0 < i < m) the learning
   procedure as
2. for all a ∈ A and s ∈S initialize Q(s,a) = 0
3. let t = 0, $jr_t^i$ = 0
4. Loop:
```

```
5.  select action $a^i$ which has max Q value
6.  execute action $a^i$
7.  receive an immediate reward $pr_t^i$
8.  observe the new state s
9.  calculate joint rewards $jr_t^i$ as
10. $jr_i = b.pr_i + (1 - b).\frac{\sum_{j=1}^{m} pr_j}{m}$    //
        $\sum_{j=1}^{m} pr_j$ sum of rewards of other
        agents except i
11. update Q learning values of agent i as
        $Q_i^{new}(x_i, a_i) := (1 - \alpha)\ Q_i^{Old}(x_i, a_i)$
        $+ \alpha\ (jr_i + \beta\ max_a\ Q(s, a))$
12. end
```

VI. EXPERIMENTAL SETUP

Model design:

Maximize the sale of products that depends on price of product, customer age and period of sale. These are the information available to each agent i.e. shop. So it becomes the state of environment. Final result is to maximize profit by increasing total sale of products[10].

Input Data set:

We define the action set as the sale of possible product. i.e. A={p1,p2,p3…….p10}

Hence action a ∈ A. State of the system is queue of customer in the particular month for the given shop agent. So state can be described as

X(t) = { x1(t), x2(t),m }

where

x1 → customer queue with age ==> { Y, M, O } i.e. young, middle and Old age customer

x2 → price of product queue ==> { H, M, L } i.e. High, Medium, Low

m → month of product sale ==> { 1,2,3,4……..12 }

In the system minimum, 108 states and actions are possible. The number of state-action increases as the number of transactions increases. For simplicity, it is assumed that single state for each transaction else the state space becomes infinitely large. Shop agent observes the queue and decides product i.e. action for each customer/state. After every sale reward is given to the agent. The table shows the snapshot of the dataset generated for single shop agent.

Table 1: Snapshot of Dataset used

TID	Age	Price	Month	Action Selected (Product)
1	Y	L	1	P1,P2,P4
2	Y	M	1	P2,P3
3	Y	H	1	P3,P4
4	M	L	1	P1,P2
5	M	M	1	P1,P2,P3
6	M	H	1	P4,P2
7	O	L	1	P1,P3

In a particular season, the sale of one shop increases. With the help of cooperative learning, other shops learn about an increase in the sale & they can take necessary actions for their profit maximization[11]. At time 0, the process X(t) is observed and classified into one of the states in the possible set of states (denoted by S). After identification of state, the agent chooses a product action from A. If the process is in state i and agent chooses a ∈ A, then

i. The process transition into state j ∈ S with probability $P_{ij}(a)$

ii. Conditional on the event that the next state is j. The time until next transition is a random variable with probability distribution Fij(./a)

After the transition occurs, product sale action is chosen again by the agent and (i) and (ii) are repeated.

VII. RESULTS

The simulation results show the efficiency of the learning algorithm. In multiagent learning algorithms are applied to shop dataset of cloth, jewellery & footwear shop and the result analysis is done for a year, the specific number of products is purchased by particular customer age group. Shop agent will understand that in a year number of products is to be sold to the customers having different age group. Fig. 1 shows the results of Strategy Sharing Algorithm for Products Vs Customer Age count and Fig. 2 gives the results of Joint Reward Algorithm for Products Vs Customer Age count.

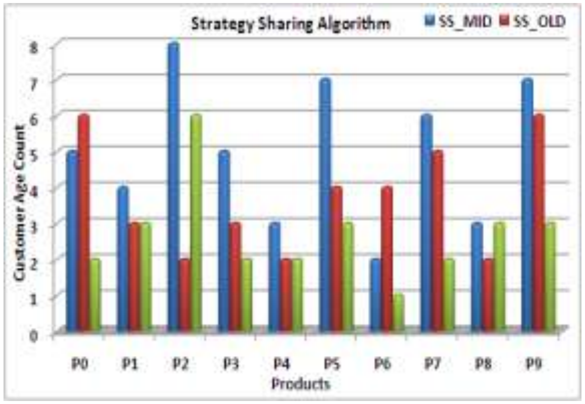

Fig. 1: Strategy Sharing Algorithm Products Vs Customer Age Count

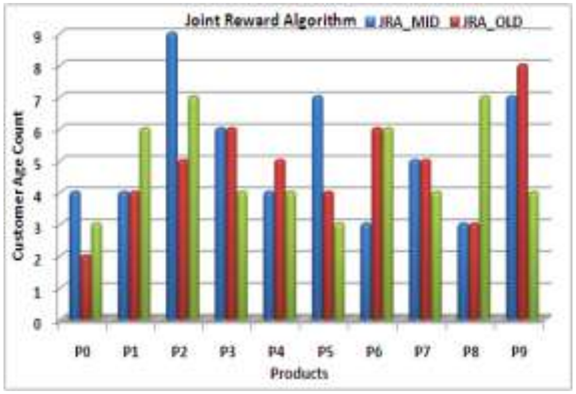

Fig. 2: Joint Reward Algorithm Products Vs Customer Age Count

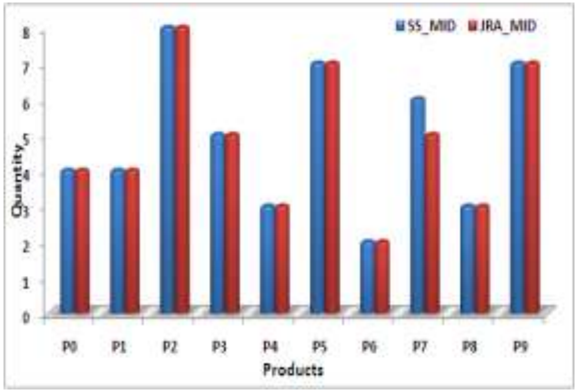

Fig. 3: Comparison of SS & JRA Algorithms for MID customer age group for Products Vs Quantity

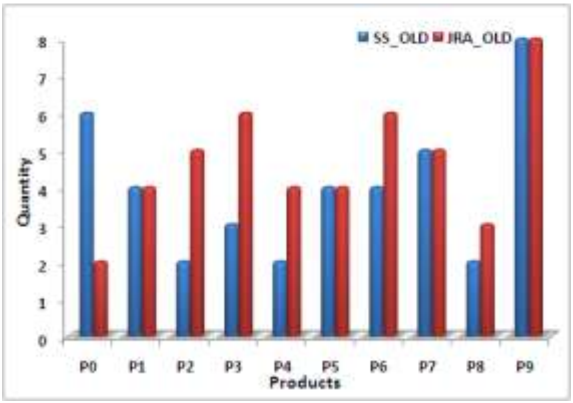

Fig. 4: Comparison of SS & JRA Algorithms for OLD customer age group for Products Vs Quantity

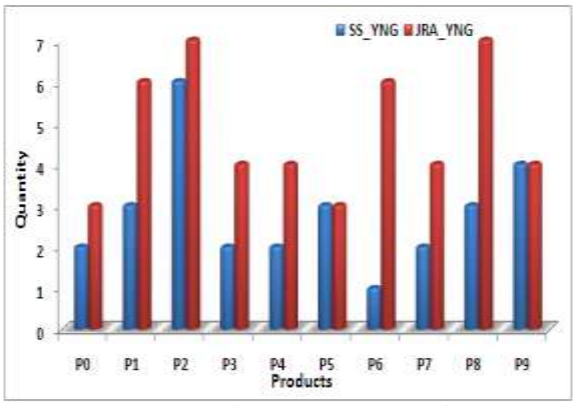

Fig. 5: Comparison of SS & JRA Algorithms for YOUNG customer age group for Products Vs Quantity

Fig 3 gives the results of comparison of SS & JRA Algorithms for MID customer age group for Products Vs Quantity. Fig. 4 shows the results of comparison of SS & JRA Algorithms for OLD customer age group for Products Vs Quantity and Fig.5 gives the results of comparison of SS & JRA Algorithms for YOUNG customer age group for Products Vs Quantity

Joint Reward Learning algorithm gives precise results than Strategy Sharing learning algorithm and gives good predictions of the products. It gives the pattern of product sale with customer age group for a period. The Q function values are tabulated for obtaining some insights. Q tables show the best action (that is optimal the product) for different individual states. By knowing the Q function, the shop agent can compute best possible product for a given state that gives maximum profit to it. Single agent learning, multi-agent learning, cooperative learning and improved cooperative learning algorithms are implemented and results are compared.

It has shown how a shop agent can effectively use reinforcement learning in setting products dynamically so as to maximize its profit matrix. It is believed that this is a promising approach for profit maximization in retail market environments with limited information.

In multiagent learning the result analysis is done for a year, the specific number of products is purchased by particular customer age group. Shop agent will understand that in a year number of products is to be sold to the customers having the different age group. Joint Reward Learning algorithm gives precise results than Strategy Sharing learning algorithm and gives good predictions of the products. These products are combined together for the increase in sale.

VIII. CONCLUSION

Learning algorithms are best suitable for decision making. Multiagent learning has more knowledge and information available. In this method, sharing of information and policy is possible. Multiagent learning always performs better compared to single agent learning. However, multiagent learning is still lacking in proper communication between agents. Sharing of more knowledge and information is possible and all agents' knowledge is used equally, jointly solves the problem cooperatively is the future scope of this paper.

REFERENCES

[1] Adnan M. Al-Khatib "Cooperative Machine Learning Method" World of Computer Science and Information Technology Journal (WCSIT) ISSN: 2221-0741 Vol.1, No.9, 380-383, 2011.

[2] Babak Nadjar Araabi, Sahar Mastoureshgh, and Majid Nili Ahmadabadi "A Study on Expertise of Agents and Its Effects on Cooperative Q-Learning" IEEE Transactions on Evolutionary Computation, vol:14, pp:23-57, 2010

[3] Dr. Hamid R. Berenji David Vengerov "Learning, Cooperation, and Coordination in Multi-Agent Systems", in Proceedings of 9th IEEE International Conference on Fuzzy Systems, 2000.

[4] Ethem Alpaydin "Introduction to Machine Learning" Second Edition, MIT Press by PHI.

[5] Jun-Yuan Tao, De-Sheng Li "Cooperative Strategy Learning In Multi-Agent Environment With Continuous State Space", IEEE International Conference on Machine Learning and Cybernetics, pp.2107 – 2111, 2006.

[6] La-mei GAO, Jun ZENG, Jie WU, Min LI "Cooperative Reinforcement Learning Algorithm to Distributed Power System based on Multi-Agent" 2009 3rd International Conference on Power Electronics Systems and Applications Digital Reference: K210509035

[7] Liviu Panait Sean Luke "Cooperative Multi-Agent Learning: The State of the Art", published in Journal of Autonomous Agents and Multi-Agent Systems Volume 11 Issue 3, pp. 387 – 434, 05.

[8] M.V. Nagendra Prasad & Victor R. Lesser "Learning Situation-Specific Coordination in Cooperative Multi-agent Systems" in Journal of Autonomous Agents and Multi-Agent Systems, Volume 2 Issue 2, pp. 173 – 207, 1999.

[9] Michael Kinney & Costas Tsatsoulis "Learning Communication Strategies in Multiagent Systems", in Journal of Applied Intelligence, Volume 9 Issue 1, pp 71-91, 1998.

[10] Ronen Brafman & Moshe Tennenholtz "Learning to Coordinate Efficiently: A Model-based Approach", in Journal of Artificial Intelligence Research, Volume 19 Issue 1, pp. 11-23, 2003.

[11] Tom Mitchell "Machine Learning" McGraw Hill International Edition.