An Analysis of Q-Learning Algorithms with Strategies of Reward Function

Ms.S.Manju,
Asst Professor,
Department of Computer Science,
Sri Ramakrishna College of Arts & Science for Women
Coimbatore-641044, Tamil Nadu, India

Dr.Ms.M.Punithavalli,
Director,
Department of Computer Science,
S.N.S. Rajalakshmi College of Arts & Science,
Coimbatore, TamilNadu, India

Abstract - Q-Learning is a Reinforcement Learning technique that works by learning an action-value function that gives the expected utility of taking a given action in a given state and following a fixed policy thereafter. One of the strengths of Q-Learning is that it is able to compare the expected utility of the available actions without requiring a model of the environment. Reinforcement Learning is an approach where the agent needs no teacher to learn how to solve a problem. The only signal used by the agent to learn from his actions in reinforcement environment is the so called reward, a number which tells the agent if his last action was good (or) not. Q-Learning is a recent form of Reinforcement Learning algorithm that does not need a model of its environment and can be used on-line. This paper discusses about the different strategies of Q-Learning algorithms and reward function.

Keywords - Machine Learning; Reinforcement Learning; Q-Learning and Relative Q- Learning Methods.

I. INTRODUCTION

Machine Learning is concerned with the design and development of algorithms. Machine Learning research is focusing on Learning and recognizing complex patterns and to make intellectual decisions based on data. In the field of Machine Learning, algorithms are organized on the expected outcomes. Reinforcement Learning is a type of Machine Learning algorithm, which gains knowledge based on the observation of environment. The outcome of the environment is rewards. The main advantage of Reinforcement Learning is that, it provides most successful rewards even when the environment is too large or cannot be shortly described.

Reinforcement Learning techniques allow an agent to become competent simply by exploring its environment and observing the resulting percepts and rewards, gradually converging on estimates of the value of actions or states that allow it to behave optimally. In this Learning there is no requirement for a skilled human to provide training examples. Secondly, the exploration process allows the agent to become competent in areas of the state space that are seldom visited by human experts and for which no training examples may be available.

Q-Learning is a Reinforcement learning technique that works by learning an action-value function. The expected utility is achieved by a given action in a given state and following a fixed policy in future. Without requiring the model of the environment, Q-Learning is able to compare the expected utility of actions. A recent variation in Q-Learning algorithm has shown the valuable improvements in achieving positive rewards [3]. Reward denotes the recent desirability of environmental states. The value of the state is the final reward, by which an agent can decide to accumulate for future decisions. The decisions are based on the positive and negative results of the agents. Q-Learning algorithm takes more time to reach the optimal Q-value in the normal circumstances based upon the reward. This paper presents an analysis of different variations in Q-Learning algorithm to achieve better and higher rewards. The main aim is to improve the performance of Q-Learning algorithm and to reduce the number of iterations to achieve the optimal Q-value. Here, we also consider better reward as main objective to reach optimal value. In general point of view the agents tend to select only those actions which have higher reward value. This study includes different methodologies followed by the new forms of Q-learning algorithm to reach the higher reward.

II. Q-LEARNING

Q-Learning proposed by Watkins is a model free and online Reinforcement Learning algorithm. This algorithm works by estimating the value of state-action pairs and it comprises of three elements namely, environment state, agent's action and environment reward [1]. Q-Learning generates Q-table, Q(s, a) which uses state-action pairs to index a Q-value. In Q-Learning at every possible step, state(s) and reward(r) is updated. Reward function defines the goal in a reinforcement Learning problem. It maps each perceived state of the environment to a single number, a reward, indicating the intrinsic desirability of that state. A Reinforcement Learning agent's sole objective is to maximize the total reward it receives in the long run. The reward function defines what the good and bad events are for the agent. In Q-Learning there is an interaction between agent and environment where the agent has to go through numerous trials in order to find out the best action. An agent chooses that action which has maximum reward obtained from its environment. The reward signal may be positive or negative depends upon the environment.

Q-Learning is a well-known algorithm for reinforcement Learning. It leads an agent to acquire optimal control strategies from delayed rewards, even when there is no prior knowledge of the effects of its action on the environment. The strategy used to select an action to perform at each step is crucial to the performance of the algorithm. Some balance between exploration and exploitation must be found. In Boltzmann exploration approach, the probability of an action being selected increases with the current estimate of its Q-value. This means that sub-optimal but good actions tend to be selected more often than clearly poor actions [2].

Normal Q-Learning algorithm is problematic because, they learn deterministic policies, whereas mixed strategies are generally needed and also the environment is generally non-stationary due to adaptation of other agents.

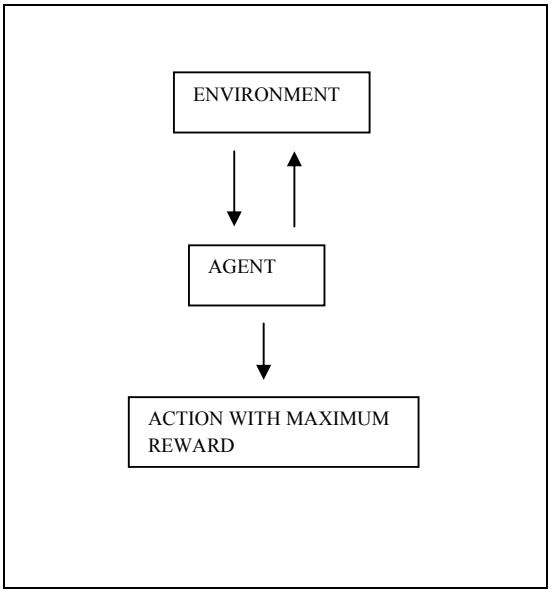

Fig. 1. Interaction between environment, agent and action

Other common disadvantage is it takes more time to reach the optimal Q-value and considerably takes more rounds (or) iterations to calculate the rewards. The traditional Q-Learning method has drawbacks in improving the positive and total number of rewards. In this work various extensions of Q-Learning methods are analyzed to reduce more number of rounds and positively to improve action with maximum rewards.

III VARIATIONS OF Q-LEARNING ALGORITHM

A. Relative Q-Learning (RQL*)*

In this new form of algorithm agents select only those actions which have higher immediate reward signal in comparison to previous one. This new algorithm helps to maximize the performance of algorithm and reduce the number of episode required to reach optimal Q-value. This form utilizes the relative reward approach to improve the learning capability of algorithm and decreases the number of iterations. This new method keeps Q-learning algorithm near to its goal in less time and less number of episode.

Relative reward method compares two immediate rewards. The objective of the learner is to choose actions maximizing discounted cumulative rewards overtime [4]. Let there is an agent in state st at time t and assume that he chooses action at. The immediate result is a reward rt received by the agent starting time t is given by:

$$r(t) = rt + \gamma rt+1 + \gamma 2\, rt+2 + \ldots\ldots\ldots + \gamma n\, rt+n + \ldots\ldots\ldots\ldots \quad (1)$$

Where r is discount factor in the range of (0:1). The immediate reward is based upon the action or move taken by an agent to reach the defined goal in each episode. The total discounted reward can maximize in less number of episode, if we select the higher immediate reward signal from previous. Relative reward based Q-learning is an approach towards maximizing the total discounted rewards. In this form, the selection is made based on the maximum immediate reward signal by comparing it with previous one. This is expressed by

$$Q(s, a) = Q(s, a) + \alpha\, [\max(r(s, a), r(s', a')) + \gamma \max i' Q(s', a') - Q(s, a)] \quad (2)$$

When this new form is tested on grid world environment, in traditional Q-learning, Q-value converges after executing 500 episodes, but Relative Q-learning takes only 300 episodes. This new concept allows the agent to learn uniformly and helps in such a way that, it will not deviate from its goal.

B. Hexagon Based Q-Learning (HBQL)

Q-Learning can be adapted to the real world environment. For e.g. state space can be harmonized with the physical space of the real world. Hexagon based Q-Learning enhance the area-based action making process so that, the learning process can be better adapted to real world situations [5].HBQL uses the different shape of the state space from the ordinary square based state space. This algorithm works by s possible state, with a possible action and r immediate reward value. Hexagon based Q-Learning can be expanded infinitely by its combination. HBQL has extra advantages that, it has fast responses and many directions of actions. This algorithm also enables agent to avoid obstacles during their search.

C. Actor/Critic Q-Learning Algorithm

Actor/Critic algorithm is equivalent to Q-Learning algorithm by construction. Its equivalence is achieved by encoding Q-values with in the policy and value function of the actor and critic [6]. This algorithm is novel in two ways: it updates the critic only when the most probable action is executed from the given state, and its rewards the actor using criteria that depend on the relative probability of the action that was executed. In actor /critic learning systems, the actor implements a stochastic policy that maps states to action probability vectors, and the critic attempts to estimate the value of each state in order to provide more useful reinforcement feedback to the actor. The result is two interacting adaptive processes: the actor adapts to the critic, while the critic adapts to the actor. In Actor/critic algorithm, the exploration strategy is more constrained. Actor/critic estimates vector (v*) directly without a model of the underlying decision process. It uses exactly the same amount of storage as Q-Learning: one location for every state/action pair.

D. Continuous-Action-Q-Learning (CAQL)

This method is a direct extension of Q-Learning that generates continuous-valued actions. The continuous action relies upon a discrete number of actions and their corresponding Q-values. This algorithm considers two types of rewards, discrete-action average reward method and discrete-action discounted reward method [7]. In continuous action version, discounted reward methods perform better than average reward ones. The continuous method updates more Q-value per unit at every step and also updates its policy instantaneously to try to select better actions. Continuous action method outperforms other discrete-action versions. In continuous method discounted reward is always superior to the average reward methods. The advantage of using CAQL is that, it is clear in terms of both asymptotic performance and speed of learning.

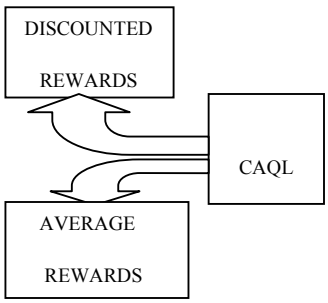

Fig. 2. Types of Rewards in CAQL

E. Bayesian Q-Learning (BQL)

In this method Watkins Q-learning is extended by maintaining and propagating probability distributions over the Q-values. These distributions are used to compute a myopic approximation to the value of information of each action and hence to select the action that best balances exploration and exploitation [8].This algorithm

can exhibit substantial improvements over other well-known model free exploration strategies. To ensure more robust behavior across the states space, exploration is crucial in allowing the agent to discover the reward structure of the environment and to determine the optimal policy. Without sufficient incentive to explore, the agent may quickly settle on a policy of low utility simply because it looks better than leaping in to the unknown. A good exploration method should balance the expected gains from exploration against the cost of trying possibly suboptimal actions when better ones are available to be exploited. Optimal solution of the exploration/exploitation tradeoff requires solving a Markov decision problem over information states that is the set of all possible probability distributions over environment models that can be arrived at by executing all possible action sequences and receiving any possible percept sequence and reward sequence. The aim is to find a policy for the agent that maximizes its expected reward. Although this problem is well-defined, given a prior distribution over possible environments, it is not easy to solve exactly. Solutions are known only for very restricted cases, mostly the bandit problems in which the environment has a single state, several actions, and unknown rewards.

Q-value sampling and Myopic-VPI (Value of Perfect Information) are the two new approaches to exploration. Q-value sampling is used for solving bandit problems. The idea is rewards as probability distributions; then, an action is selected stochastically according to the current probability that it is optimal. This probability depends monotonically not only on the current expected reward (exploitation) but also on the current level of uncertainty about the actual reward. Bayesian method is used for representing, updating, and propagating probability distributions over rewards [9].

Myopic value of perfect information provides an approximation to the utility of an information gathering action in terms of the expected improvement in decision quality resulting from the new information. This provides a direct way of evaluating the exploration/exploitation tradeoff. Like Q-value sampling, Myopic-VPI uses the current probability distributions over rewards to control exploratory behavior. Q-value sampling resembles, to some extent, Boltzmann exploration. It is stochastic exploration policy, where the probability of performing an action is related to the distribution of the associated Q-values. One drawback of Q-value sampling is that it only considers the probability that a is best action, and does not consider the amount by which choosing a might improve over the current policy.

Myopic-VPI considers quantitatively the question of policy improvement through exploration. It is based on value of information. Its application in this context is reminiscent of its use in tree search which can also be seen as a form of exploration – in the form of improved policies-against the expected cost of doing a potentially suboptimal action.

Bayesian approach use probability distributions to represent the uncertainty the agent has about its estimate of the Q-value of each state. As is the case with undirected exploration techniques, we select actions to perform solely on the basis of local Q-value information. However, by keeping and propagating distributions over the Q-values, rather than point estimates, we can make more informed decisions. In Bayesian method, we need to consider prior distributions over Q-values, and then update these priors based on the agent's experiences. Formally, random variable denotes the total discounted reward received when action a, is executed in state s and an optimal policy is followed thereafter. Q-value sampling and Myopic VPI are used for action selection. Experimental evidence has shown that these approaches explore the state space more effectively than convectional model-free learning algorithms, and that their performance advantage appears to increase as the problems become larger. This is due to an action selection mechanism that takes advantage of much more information than previous approaches.

F. Hyper-Q Learning (HQL)

Hyper-Q is effective against many different types of adoptive agents if they are persistently dynamic. Against certain broad categories of adaptation, it is argued that Hyper-Q may converge to exact optimal time varying policies. Trial and error learning could develop good strategies by trying many actions in a number of environmental states, and observing which actions in combination with actions of other agents, lead to high cumulative reward. Hyper-Q learning's key idea is to learn the value of joint mixed strategies rather than joint base actions [10]. In Hyper-Q learning the agent needs to learn a mixed strategy, which may depend on the mixed strategies of other agents, an obvious idea is to have the Q function evaluate entire mixed strategies, rather than base actions and to include in the "state" description an observation or estimate of the other agents current mixed strategy this forms the basis of Hyper-Q learning. For notational simplicity, let x denote the Hyper-Q learners current mixed strategy, and let y denote an estimated joint mixed strategy of all other agents. At time t, the agent generates a base action according to x, and then observes a pay of r, a new state s', and a new estimated opponents strategy y'. Hyper-Q function is given by Q (s, y, x). Hyper-Q learning appears to be more versatile and general purpose than any published multi-agent extension of Q-learning to date.

G. Modular Q-Learning (MQL)

Modular Q-Learning is employed for multi-agent learning in reinforcement methods. In the environment

under consideration is dynamic and complex, reinforcement learning scheme should be employed for the selection of proper role. Reinforcement learning is the problem faced by an agent that learns its behavior through trial-and-error interactions with a dynamic environment [13]. The agent only knows the possible states and actions, not the transition probabilities or the reward structure. Among the reinforcement learning methods, Q-Learning can be used in the reinforcement scheme as it is applicable where there is no model of the environment. Modular Q-Learning is adopted using the concept of coupled agent and is useful in resolving agent conflicts. For example, consider two agents being engaged in a joint task. Here a joint task implies two agents working together to find an optimal solution. As many actions are needed in every state, it needs more memory space for multi-agent learning. Such an application of Q-Learning to the kind of multi-agent learning problems results in the explosion of the state and memory space. To overcome this problem, modular Q-Learning is employed. The main advantage of Modular Q-Learning is solving agent's problem in the action selection process. The concept of coupled agent was used to resolve agent conflicts.

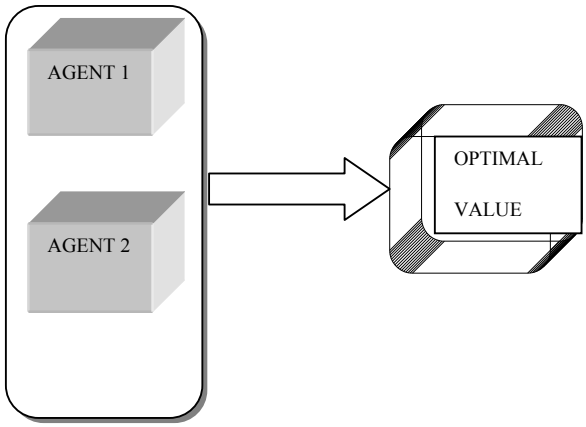

Fig. 3. Coupled Agent

H. Lazy Q-Learning (LQL)

Reinforcement Learning approach and Lazy Learning approach is combined to form Lazy Q-Learning. It's a very efficient single robot learning paradigm. Lazy learning is also called instance-based learning, promotes the principle of delaying the use of the gathered information until the necessity arises it is impossible to predict the acquisition of new targets. If we define the reinforcement function as positive if one or several targets have been acquired by the group, and negative if one or several targets have been lost, then an increase in system performance can only be sought by reducing negative rewards [11].

The Lazy Q-Learning algorithm is able to compute for each team member a lower bound of the utility of executing an action in a given situation. In order to express different behavior, the memory must be searched. The memory is searched with reinforcement function. Their objective is to provide a method for predicting the rewards for some state-action pairs without explicitly experiencing them. They call algorithm as Lazy Q-Learning. For the current real world situation, a situation matcher locates all the states in the memory that are within a given distance. If the situation matcher has failed to find any nearby situations, the action comparator selects an action at random. Otherwise, the action comparator examines the expected rewards associated with each of these situations and selects the action with the highest expected reward. This action is then executed, resulting in a new situation [12]. There is a fixed probability of generating a random action regardless of the outcome of the situation matcher. New situation-action pairs are added to the memory, along with a Q-value computed in the classical way. Among similar situation action pairs in the memory, an update of the stored Q-value is made. There is an important performance gain associated with the Lazy Q-Learning over a purely random selection policy. This clearly demonstrates the importance of lazy Q-Learning as a learning technique. Even more interestingly, lazy Q-Learning performs better than the user defined non-cooperative policy. This algorithm is able to compute for each team member a lower bound of the utility of executing an action in a given situation. The result show that this algorithm accounts for increased performance of the groups. In Lazy Q-Learning rewards are distributed and the main important issues are: quality of the reinforcement function, quality of the sampling (size), and quality of probing process (generalization).

IV DISCUSSIONS

In this paper Q-Learning variations are analyzed based on the reward function to make right decision. The above mentioned algorithms reach the optimal value in different angles. In Relative Q-Learning algorithm agents select only those actions which have higher immediate reward signal in comparison to previous one.

This new algorithm helps to maximize the performance of algorithm and reduce the number of episode required to reach optimal Q-value.

Hexagon based method works by s possible state, with a possible action and r immediate reward value. Hexagon based Q-Learning can be expanded infinitely by its combination. HBQL has fast responses and many directions of actions. This algorithm also enables agent to avoid obstacles during their search. Actor/Critic algorithm is novel in two ways: it updates the critic only when the most probable action is executed from the given state, and it rewards the actor using criteria that depend on the relative probability of the action that was executed.

Continuous-action algorithm considers two types of rewards, discrete-action average reward method and discrete-action discounted reward method. In continuous action version, discounted reward methods perform better than average reward ones. The continuous method updates more Q-value per unit at every step and also updates its policy instantaneously to try to select better actions. Bayesian Q-Learning Algorithm can exhibit substantial improvements over other well-known model free exploration strategies. To ensure more robust behavior across the states space, exploration is crucial in allowing the agent to discover the reward structure of the environment and to determine the optimal policy.

TABLE 1 - Strategies used in variations of Q-Learning

S.NO	VARIATIONS OF Q-LEARNING	STATEGIES APPLIED
1	RQL	Selecting Immediate Rewards
2	HBQL	Possible state, action and immediate reward value
3	ACTOR/CRITIC	Rewards the actor based on Relative Probability
4	CAQL	Discounted reward methods
5	BQL	Exploration to discover reward structure
6	HQL	Trial and Error Learning
7	MQL	Coupled Agent
8	LQL	Using unexperienced rewards

Hyper-Q algorithm converges to exact optimal time with varying policies. Trial and error learning could develop good strategies by trying many actions in a number of environmental states, and observing which actions in combination with actions of other agents, lead to high cumulative reward. Hyper-Q learning's key idea is to learn the value of joint mixed strategies rather than joint base actions.

Modular Q-Learning is adopted using the concept of coupled agent and is useful in resolving agent conflicts. The main advantage of Modular Q-Learning is solving agent's problem in the action selection process. The concept of coupled agent was used to resolve agent conflicts. The Lazy Q-Learning algorithm is able to compute for each team member a lower bound of the utility of executing an action in a given situation. In order to express different behavior, the memory must be searched. The memory is searched with reinforcement function. Their objective is to provide a method for predicting the rewards for some state-action pairs without explicitly experiencing them.

These algorithms are analyzed with their key ideas which improve the expected rewards to reach the optimal Q-value. The nature of environment and the situation plays major role in selecting the actions. In case of multi-agent environments different strategies are used to satisfy the conditions of the agents. These variations discussed above could be used to improve the reward structure and also selection may be done by considering type of environment we are going to use. The main aims of this study it to achieve optimal solutions. When the performance of the reward function is increased, automatically we can achieve optimal solutions.

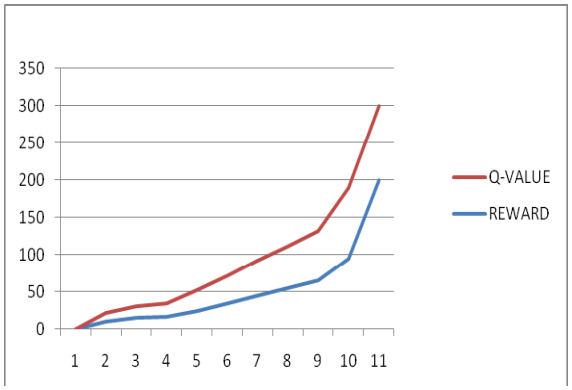

Fig. 4. Increase in Performance of Reward function leads to optimal Q-value

V CONCLUSIONS

This paper presents improved strategies of native Q-Learning algorithm and adapting to different variations will lead native methods to work more efficiently. The main problem of native Q-Learning is it takes more time to reach the optimal value in normal circumstances with the obtained rewards. The positive rewards are not consistent in traditional methods. Before realizing a negative reward more iteration will be performed. Here the major disadvantage is maximum time is wasted especially in online applications and takes more iterations. There is no any adaptive method to use the rewards. The variations of Q-Learning are analyzed in the aspects of using cumulative rewards in different way to achieve the expected solution. For example, in relative method immediate rewards are selected, in continuous-action method discounted rewards are selected, in Hyper-Q with cumulative reward mixed strategies are used, in Modular method with rewards agents are coupled to reach the desired goal and the Lazy Q-Learning's objective is to provide a method for predicting the rewards for some state-action pairs without explicitly experiencing them. The above discussed algorithms work better when compared to native methods. These methodologies overcome the main draw backs (time consuming and more iteration) of native Q-Learning algorithm. With these new strategies we can achieve the higher positive rewards and the optimal Q-values. However, these new strategies of the above mentioned variations will work based on the environment and the agents we are going to select. In future, we can concentrate on adaptive agents for different types of environment and agent's knowledge in analyzing positive rewards.

REFERENCES

[1] C.Watkins"Learning from Delayed Rewards", PhD thesis, Cambridge University, Cambridge, England, 1989.
[2] Technical Note Q-Learning Christopher J.C.H. Watkins and Peter Dayan Centre for Cognitive Science, University of Edinburgh, Scotland Machine Learning, 8, 279-292(1992).
[3] L.P.Kaelbling, M.L.Littman, A.W. Moore. Reinforcement learning: A survey Journal of Artificial Intelligence Research, 4, 1996, 237-285.
[4] Punit Pandey, Deepshikha Pandey, Dr.Shishir Kumar, Reinforcement Learning by Comparing Immediate Reward, IJCSIS, Vol.8, No.5, Aug.2010.
[5] Hyun-Chang Yung, Ho-Duck Kim, Han-Ui Yoon, In-Hun Jang, and Kwee-Bo Sim, Hexagon-Based Q-Learning Algorithm and Applications, International Journal of Control, Automation and Systems, Vol.5, No.5, pp.570-576, Oct.2007.
[6] Jose Del Millan, Daniele Posenato, Eric Dedieu "Continous-Action Q-Learning, Joint Research Centre, European Commission, Machine Learning,49, 247-265, 2002.
[7] Robert H. Crites, Andrew G.Barto, An Actor/Critic Algorithm that is Equivalent to Q-Learning, Advance in Neural Information Processing Systems 7, MIT Press, Cambridge MA, 1995.
[8] D.A.Berry and B. Fristedt. Bandit Problems: Sequential Allocation of Experiments. Chapman and Hall, 1985.
[9] Richard Dearden, Nir Friedman, Stuart Russell – Bayesian Q-Learning.
[10] Gerald Tesauro, Extending Q-Learning to General Adaptive Multi-Agent Systems.
[11] Sheppard, J.W. and S.L. Salzberg (1997), A Teaching Strategy for Memory-Based Control, Lazy Learning, D.Aha (Ed.), Kluwer Academic Publishers,pp.343-370.
[12] Claude F.Touzet, Distributed Lazy Q-Learning for Cooperative Mobile Robots.
[13] Kui-Hong Park, Yong-Jae Kim, and Jong-Hwan Kim, Modified Uni-Vector Field Navigation and Modular Q-Learning for Soccer Robots, Proceedings of the International Symposium on Robotics, April 2001.