World Engineering & Applied Sciences Journal 7 (4): 267-274, 2016
ISSN 2079-2204
© IDOSI Publications, 2016
DOI: 10.5829/idosi.weasj.2016.267.274

Efficient Knowledge Sharing and Fast Learning Agent Based on Enhanced Dyna-QPC for Multi-Agent Systems

[1]Sindhu and [2]Dr. P. Thambidurai

[1,2]Department of Computer science and Engineering, Bharathiar University, Coimbatore, Tamil Nadu, India
[2]Department of Computer science and Engineering,
Perunthalaivar Kamarajar Institute of Engineering & Technology, Nedungadu, Karaikal, India

Abstract: In a multi agent system, the accuracy of the learning process is a major task between peers for the access of agents' capabilities for environmental modelling. It can relieve the liability of investigation for invisible or unreached states. Meanwhile, in a limited period of time the development of accurate and effective model is a major task, specifically for difficult atmospheres. In this paper, the enhanced reinforcement learning method is proposed to have efficient modelling with reduced consumption of memory. The real time process generates the essential capabilities in a model which reduce the learning elapsed time and appropriate for sharing knowledge. The Enhanced Dyna-QPC (EDQPC) approach is proposed for efficient learning process and its strategy integrates the aptitude of the learning model with less time for training than the existing approach in real-world tasks. The learning agent involves the policy learning with the management of planning function which update the status of the learning. In order to attain fast learning policy between the modes of virtual and real is processed efficiently by the proposed approach. The simulation analysis is obtained to show better performance in efficiency, speed and accuracy than the existing learning approach.

Key words: Q-Learning · CMAC · Reinforcement Learning · Model Sharing · Dyna Agent · Sweeping

INTRODUCTION

Reinforcement learning (RL) does an examination and manipulation process to obtain the rewards from the evaluated results. The model with less time achieves less accuracy when compared to the high accurate model. The RL agent attains an accurate model in the application domain and the model can be used to perform the value iterations by indirect learning. It gives the same output in direct learning, but it takes simulated experiences generated by the model, instead of real experiences [1].

Generally, RL agents do not have prior knowledge about an environment [2]. They learn the optimal policy from series of trial –and-error based concepts. Because of this process, the time taken is longer to complete. Therefore, sample efficiency is generally required for the RL applications to learn an effective policy [3].

The Dyna architecture is a model based method which is extended from RL architectures [2] and also it includes policy learning and an internal world model. But Dyna architecture avoids the process of building internal model by using environmental modelling look up table methods. The table model is designed to decide the grid resolution in a continuous space. Though higher resolution it takes more time to obtain the output model with more accurate than the model which takes less time.

RL algorithms are extended to multi agent systems by the recent approaches, where agent deals with the task using decentralized structures for solving more complex problems [4-17]. Though agents the experience are shared with partners, they usually evaluate their own sophistication levels of knowledge. Therefore, with less knowledge agents take advantage of the ones with more experiences via sharing processes. The mechanism that purposely shares experiences to one another which are realistic to the RL model is named as policy sharing [4], [15-16]. As well, the mechanism applied to model-based RL for environmental modelling is referred to as model sharing [5, 9, 11].

Corresponding Author: Sindhu, Department of Computer science and Engineering, Bharathiar University, Coimbatore, Tamil Nadu, India.

There are two approaches in achieving this goal. They are: First approach, the state aggregation method into model-free method, in which value function is learned without using world model and generalizes the continuous state information that has a similar value function [7,18, 19]. The Second method is model-based algorithm, which forms the conversion function possibility and rewards them by limiting the amount of involvement with fewer environmental interactions [8]. The model-free methods learn their policy very slowly than the model-based methods.

There are some challenges faced with extending the model based RL on the applications of multi agent systems. First, the properties of environments can be classified into deterministic and stochastic. When it is under deterministic environment, the influence of transition probability is ignored. But in a stochastic environment, agents always transit onto next states with some degree of uncertainty such that transition probability should be brought into model learning methods appropriately. Moreover, in order to have an accurate virtual model of resembling the environment the environment is discovered systematically and repeatedly by the agents and it is essential. This process is quite time-consuming and costs tremendous computational power.

Sharing the knowledge between peers can decrease the learning effort and have saved time. It is easy to accomplish the agents having the same partition pattern in the state space or state aggregations so as to share their model or policy information straight forwardly. If each individual model is held by a heterogeneous structure than the information sharing becomes a corresponding problem.

The sharing methods assume that the agents learn policies in a deterministic and stochastic environment. They also share information between heterogeneous models. Tree structures are used to construct environmental models and to share their experience by considering the leaf node information. So, alleviate the loads on data transfer between agents and save the computational time of the sharing process [6]. As per Chebyshev's theorem the decision of interval is carried out for data appearances [14].

This paper includes the model in the learning agents, which are constructed by discrepant decision trees. The model sharing is treated technically as tree merging, so that the proposed methods are to be merged with the whole tree; that is, the methods need to transplant the whole decision tree to other agents [13]. This model is presented to achieve sample efficiency in task domains with continuous state spaces. The main task is to automatically generate various resolutions by using a decision tree and to approximate the transition probability between two successive states. Based on this, several sharing methods for multi agent systems are proposed. If the learning agents acquired sufficient experience, then they should share their experience with the others to construct their model [10].

The rest of the paper is organised in a section wise. In section II, the background of the learning and architecture is presented with the approaches. In section III, the proposed approach is discussed with the flow work and the procedures. In section IV, the simulation results of the proposed approach are obtained to show the analysis of performances that the proposed approach is improved than the existing. The results demonstrate the applicability and effectiveness of the algorithms. Finally, the conclusion is presented in Section V.

Background: In this section, the process of learning methods and the model survey is presented with the techniques. The following models are included in the approach to have a better approach.

Dyna Architecture: The architecture is extended from the RL approach with the world model and policy learning. In estimated model,the current stateis denoted as s_t, the input actions as a_s, s_{t+1} and r_{t+1}represented as next state and rewards respectively as outputs. The realexperiences are used to develop the model by updating the function and collecting information is defined as a direct RL. The indirect RL describe the virtual model with policy learning and it is also named as planning. The general architecture of the Dyna is shown in Fig.1.

In that diagram the interaction between the environment and the agent is represented as a bold arrow. The learning policy of the agent is represented as left dash arrow. From the interaction of direct RL the values are updated and the process of experience retrieves from model is indicated as control arrow of search. The final outcomes are stored on the memory table and the policies are updated from the experiences simulated. So this model is based on a table. Dyna architecture organizes with the Q-learning to have an efficient framework by including acting, direct RL, planning and learning model[19].

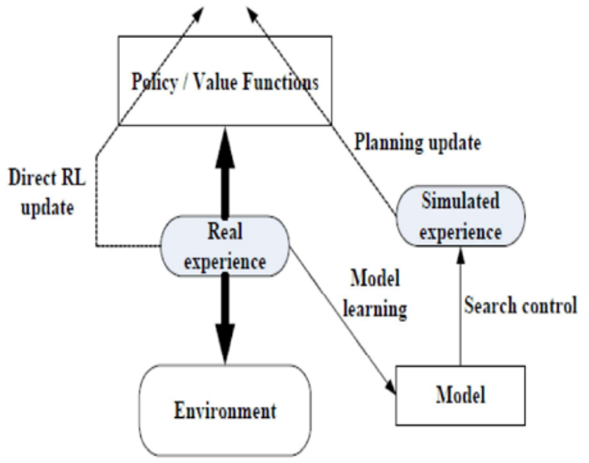

Fig.1: General flow work of Dyna Architecture

Markov Decision Process (MDP): In RL, off-policy method is processed with the strategy and uses the function of the optimal value of the optimal policy. The property of Markov is satisfied in the process of RL and defines the action (a), state (s) and next state (s') of the environment[20].

The probability of each transition is p(s'| s, a) and the reward function is defined as r(s, a, s'). The policy is determined when the action chosen in the given state at the time (t) step and exploits the predictable reduced upcoming accumulative return: $r_t+\gamma r_{t+1}+\gamma^2 r_{t+2}+...$, or reward, where the discount rate factor is $0 \leq \gamma \leq 1$.

Reinforcement Learning: To solve the issues of mathematical the machine learning is the science of creating algorithms to solve mathematical problems. The issues are modelled as acting agent and the classic environment is expressed in the learning machine as a Markov decision process (MDP). Reinforcement learning (RL) is a dynamic user interface design used to split the issues into sub part to have a quick solution. In RL, the solutions are definable by the trial-and-error and have the aspects like the function of the value function, environment and reinforcement function. The function of reinforcement is optimized basedon the value function[21].

The reinforcement function is well-definedwith the exact function of upcoming rewards for maximizing the agent seeks. The states value mapping is defined by the value function and based on policy the state selection is processed by the action. Also, thereinforcement is maximized and finds the optimal solution for the function[22].

Cerebella Model Articulation Controllers (CMACs): CMACs are used to estimate the functions outcomes by mentioning the table of look-up which stores the synapticweights. The corresponding output is derived by summing up some of these synapticweights. The functions relationship is processed by the fine-tuning weight process and it can be approached by the CMAC. The flow of mapping sequence is $S \rightarrow C \rightarrow P \rightarrow O$. O = h(S) represents the overall mapping. The architecture of CMAC is shown in Fig. 2.

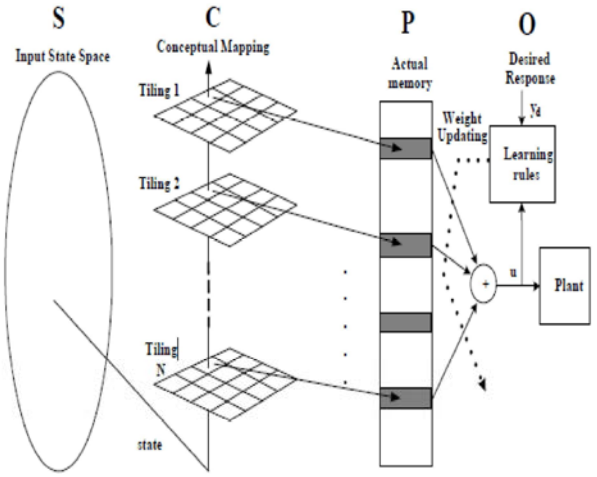

Fig. 2: Basic CMAC Architecture

Prioritized Sweeping: The interaction of the model considers the updating the values of Q which obtain during the interval of time. However, the value of Q represents the state-action effects. As per the measurement determination the updating of data is carried out with the usual prioritize and the algorithm is implemented as per the flow of architecture.

Q-Learning: The innovation of RL development is the reinforcement of the Q-learning algorithm and categorized as a model-free algorithm. Based on the estimation of the Q* best value the optimization of pairs is processed with the learned Q value andlearning rate (β) [22]. The updating process is carried out as given below.However, the issue of this algorithm is taking more time to interact with the environment in order to resolve the issues.

$$Q(s,a) = Q(s,a)+\beta(r+\gamma \max_{a'} Q(s'-a')-Q(s,a)) \quad (1)$$

Proposed Work: Enhanced Dyna – QPC (EDQPC): In this section, the explanation of the proposed technique with its model and the implementation is discussed.

In proposing technique, the prioritized sweeping, CMAC and Improved Q-learning algorithm are combined to have a minimum requirement of learning time. In the proposed architecture the process of CMACs, learning process and prioritized sweepingis implemented as same with the function of training time estimation but modified in the part of Q-Learning Algorithm.

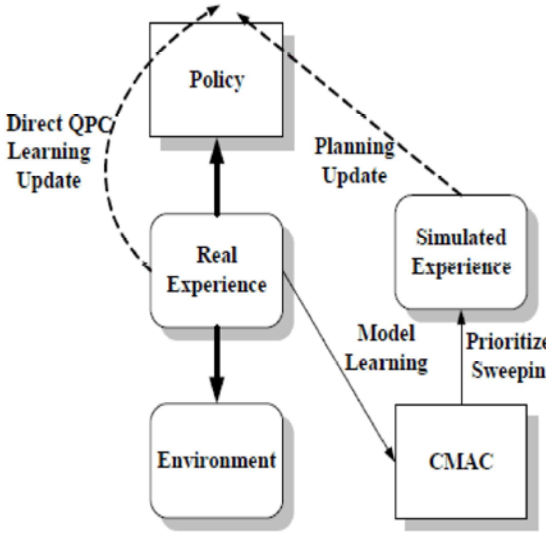

Fig. 3: Architecture of Dyna Agent

In Dyna architecture, the learning process of reinforcement is carried out by using the proposed approach of Enhanced Dyna – QPC (EDQPC). The EDQPC approach processed with the improvement of Q-Learning and policy for real-world tasks. As explained in the section background the techniques are used with the improvement. Fig.3 shows the Dyna agent architecture.

CMACs and Prioritized Sweeping: The properties of output superposition, dynamic computation and the local generalization are in CMAC. The structure simplifiesthe model estimate by gathering the real involvementoverinterfacefor a virtual world model development. The mapping sequence weight is updated in the table as per the function of CMAC. The storage of weight in the memory cell is represented as u_m; W_{mn} as indexed cell of weight, which update the error minimization between y_{dn} desired values.Fig. 4 shows the model architecture of CMAC and the cyclic path sequence is carried out based on the interaction of agent act.

$$u_m = W_m x_m = \sum_{n=1}^{R} W_{mn} x_m \quad (2)$$

$$\Delta w_{mn} = \frac{y_{dm} - W_m x_m}{R} \quad (3)$$

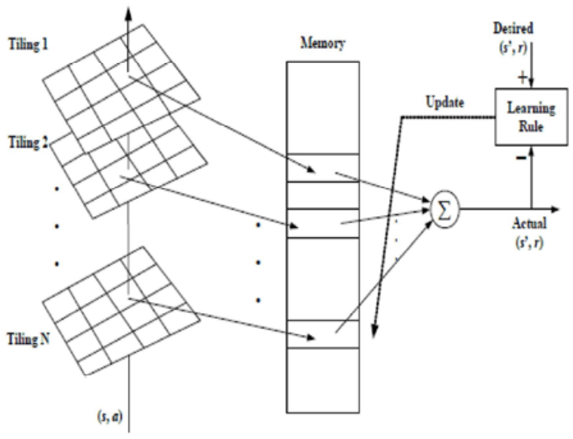

Fig. 4: Flow of Proposed system CMAC Architecture

Algorithm: CMAC

```
Clear memory, $w_{s,a}(F) = 0$, where ◎s states, ◎a actions, ◎F features
Do forever:
  1.If model learning: (s, a)? environment
  2.Else: (s, a) ←first(PQueue)
  3.F= features(s, a)// CMAC hit memory
  4.(s', r) = sum(F) // output
  5.If model learning: //update substances of hit memory
  6.$W_{s,a}(F) = w_{s,a}(F) + \gamma|c[desired(s', a) – actual(s', a)]$
  7.go to step 1;
  8.Else Exit.
```

Improved Q-Learning Algorithm: In Q-learning process, the environment knowledge is not needed and it is model-free reinforcement learning. It combines and expands the activitiesthrough error and trial process. In the framework of reinforcement learningthe state choosesasuitable action with instant reward (r) and maximizes the challengesof the long-termrewards. The procedure converges infinitely pairs visit with the probability.

In single-agent, it operates with the process of Markov Decision in a finite-discrete-time. The environment variations carried out based on the state transition probability function. The reward (r) function is determined based on the activities of travel length, duration, start time, travel time and attraction degree.

The reinforcement learning tasks obtain thevisualstrategy (◎◎*) to achieve maximumaggregate reward (◎◎$^{??}$)for every state. The accumulatedvalue (◎◎$_{??}$) accomplishedfrom initial state by therandom policy ◎◎.

$$(◎◎_{??})=r_t+\gamma r_{t+1}+\gamma^2 r_{t+2}+\ldots = \sum_{n=0}^{\infty} | \gamma^2 r_{t+n}, \qquad (4)$$

$$◎◎^*=\text{argmax}_{\pi} V^{\pi}(s),\ ◎(s) \qquad (5)$$

The maximum reward from the present state by the optimal policy (◎◎*) is denoted as $V^{\pi}(s)$. The function Q value is determined the instant reward plusof the successive state. At each stage the indexed Q-value is updated as given below.

$$Q(s,a)=(1-\propto)Q(s,a)+\propto(\beta(r+\gamma \max_{a'} Q(S',a')-Q(s,a)) \qquad (6)$$

Algorithm: Improved Q-Learning

```
Require: Initialize Q(s,a) with arbitrary values
For all episodes do
  1.Initialize s(0)
  2.t← 0
  3.Repeat
     a.Select action a(t) in state s(t), using a policy
       derived from Q;
     b.Execute action a(t), observe r(t+1) and s(t+1);
     c.$Q(s(t),a(t))=(1-\propto)Q(s(t),a(t))+\propto(\beta(r(t+1)$
          $+\gamma \max_{a'} Q(s(t+1)',a(t)') -$
          $Q(s(t),a(t)))$
     d.t←t+1
  4.Until s (t) being a terminal state
  End for
```

The procedure of the Q-Learning is given below:

- The ◎◎-values initialization
- The starting state ◎◎ is selected randomly, at least with one promising action.
- Choose one action and the possible action leads to the next state
- As per the policy the state-action pair Q-Value is updated.
- If possible actions available at the new state, then go back to Step 3
  Else Step 2.

Prioritized Sweeping: In the proposed architecture, prioritized sweeping is implemented with the improved function of the model. It is used for managing the system of Markov with efficient prediction and accurate process to have a real time presentation.

Algorithm: Prioritized Sweeping

```
Initialize Q(s, a), Model(s, a), for all s, a and PQueue to
empty
Do repeatedly:
  1.s← current (nonterminal) state
  2.a← policy(s, Q)
  3.Execute action (a); observe resultant state (s') and
    reward (r)
  4.Model(s, a) ←s', r
  5.p←|r + $\gamma$ max$_{a'}$Q(s', a') – Q(s, a)|
  6.if p>, then insert s, a with priority p into Pqueue
  7.While PQueue is not empty,Repeat N time:
      s, a← first(PQueue)
      s', r← models(s, a)
      Q(a, s); // equation [6]
  8.Repeat, for all s,a predicted to lead to s:
      r← predicted reward
      p←r+ $\gamma$ max$_a$Q(s, a) – Q(s, a)|
      Go to Step 6;
```

The significant difficulty in prioritized sweeping is that the discrete state assumption. If in any state any changes occur, then the preceding state computation may be affected. However, it is not clearlypointed about the efficient process of identification.In this paper, CMAC used an approximate model and resist with the effect of variation of its Q value, from CMAC the affected states are rescued. According to the procedure given below the function of the model is carried out.

Simulation Result: In this section, the performance of the proposed approach is simulated and obtains the result of analysis to show the improvement than the existing. The simulated is carried out for the issues of mountain car, maze and acrobot. Also, the performance of the proposed approach Reinforcement Learning is illustrated.

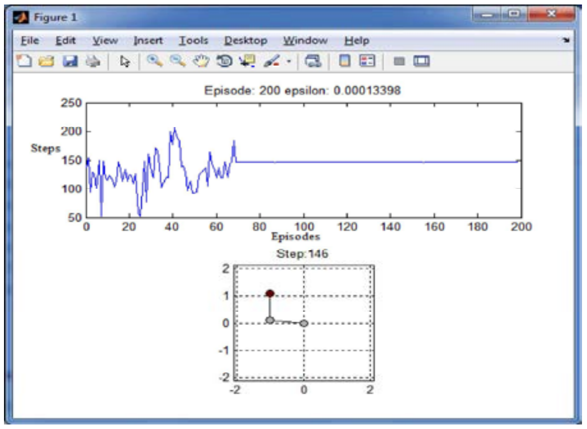

Fig.5: Simulation Results of Acrobot

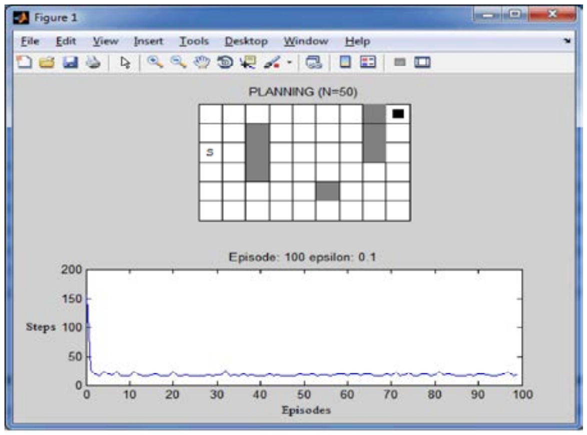
Figure 1
File Edit View Insert Tools Desktop Window Help
PLANNING (N=50)
S
Episode: 100 epsilon: 0.1
Steps
Episodes

Fig. 6: Simulation Results of Maze

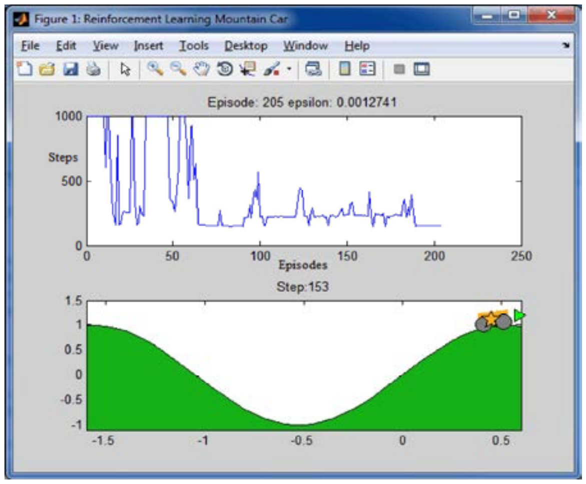
Figure 1: Reinforcement Learning Mountain Car
File Edit View Insert Tools Desktop Window Help
Episode: 205 epsilon: 0.0012741
Steps
Episodes
Step:153

Fig. 7: Simulation Results of Mountain Car

The acrobot defines the arm of theunder actuated robot and in machine learning the control process of it is an issue. The performance of the learning method in simulation of acrobot is shown in Fig. 5. The maximum number of steps per episode for acrobot is 1000, the discount factor is 1.0, the random selection of actionprobability is 0.001 and the learning rate is 0.5.

The issue of puzzles is referred in maze issues. It processes the path collection to move from starting position to the end position and some recent games are related to it. The simulation result of the maze is shown in Fig.6. The maximum number of steps per episode for the maze is 2000, the learning rate is 0.1, the random selection of actionprobability is 0.1 and the discount factor is 0.95.

In Mountain Car simulation, the car in the valley has to reach the peak directly. As per the technique the performance of the mountain car is carried out as shown in Fig. 7. The proposed agent can reduce the average steps and obtain its objective in the 5th episodes. Epsilon 10% andLambda 0.3 is obtained from the result. The maximum number of steps per episode is 1000, the discount factor is 0.8 and the learning rate is 0.7.

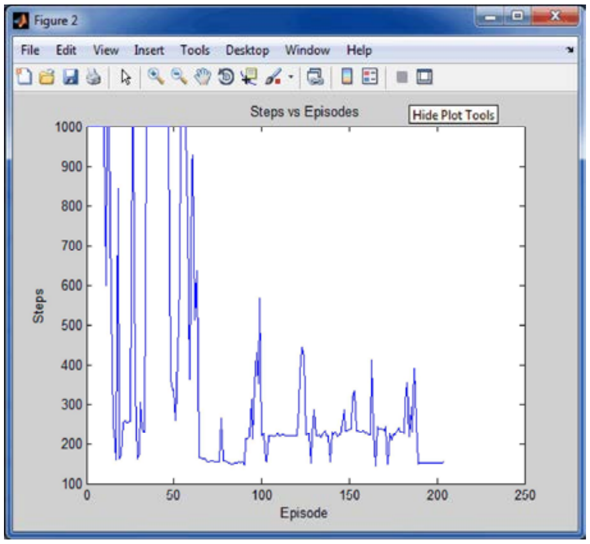

Fig.8: Simulation Results of Steps Vs Episode

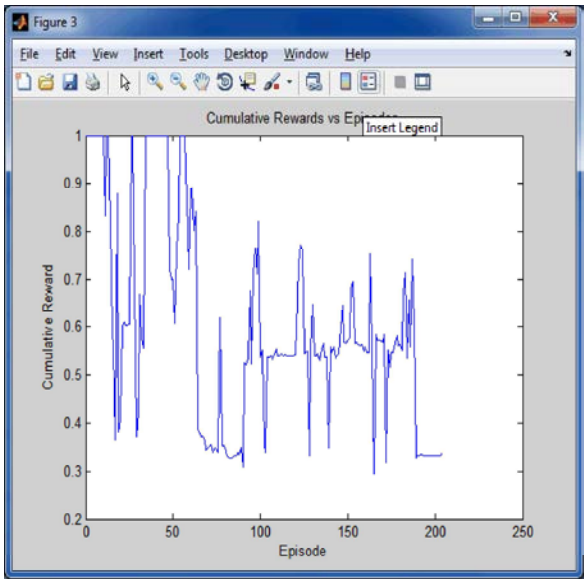

Fig. 9: Simulation Results of Cumulative Rewards Vs Episode

The performance of the proposed reinforcement learning approach Steps Vs Episode and Cumulative Rewards Vs Episode is shown in Fig. 8 and Fig. 9 respectively. Finally the learning performance of the proposed RL approach is shown in Fig. 10. Table [1] shows the training time comparison of various methods. The proposed EDQPC achieves 5 % faster than the Dyna-QPC and 46% faster than Table based Dyna-Q.

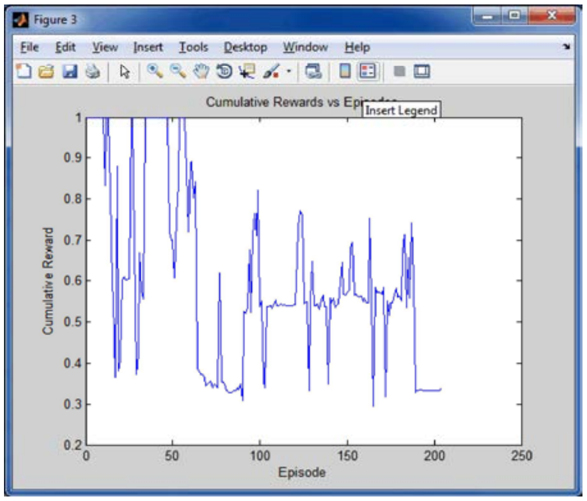

Fig.10: Simulation Results of Learning Performance

Table 1: Comparison of Various Methods for training time

Methods	Training Time
Q-learning	Unable to meet conditions
Table-basedDyna-Q	32193 seconds˜ 8.9 hours
Dyna-QPC	18618 seconds˜5.2 hours
EDQPC	14895 seconds˜4.01 hours

CONCLUSION

In this paper, the Enhanced Dyna-QPC (EDQPC) is proposed to have efficient performances and to resolve the issues occurred in learning process. In EDQPC, the input and output of the CMAC technique are estimated according to the process and provides sequential state-reward pair form the state-action pair. The function searching control is processed to retrieve the appropriatestate-action pairs by using prioritized sweeping technique. The Q-Learning model is improved by the Dyna agent process torecover the experiences effective and updates the reinforcement learning between agents as per the time period. The simulation and researchoutcomesestablish the performance of EDQPC. Also, the time required for training is reduced in the proposed method than the existing methods.

REFERENCES

1. Hwang Kao-Shing, Wei-Cheng Jiang and Yu-Jen Chen, 2015. Model Learning and Knowledge Sharing for a Multiagent System With Dyna-Q Learning, in Cybernetics, IEEE Transactions on, 45(5): 978-990.
2. Bianchi, R.A.C., M.F. Martins, C.H.C. Ribeiro and A.H.R. Costa, 2014. Heuristically-accelerated multiagent reinforcement learning, IEEE Trans. Cybern., 44(2): 252-265.
3. Konar, A., I.G. Chakraborty, S.J. Singh, L.C. Jain and A.K. Nagar, 2013.A deterministic improved Q-learning for path planning of a mobile robot, IEEE Trans. Syst., Man, Cybern. B, Cybern., 43(5): 1141–1152.
4. Chen, Y.J., K.S. Hwang and W.C. Jiang, 2013. Policy sharing between multiple mobile robots using decision trees, Inf. Sci., 234(10): 112-120.
5. Santos, M., H.J.A. Martín, V. López and G. Botella, 2012. Dyna-H: A heuristic planning reinforcement learning algorithm applied to role-playing game strategy decision systems, Knowl.-Based Syst., 32: 28-36.
6. Hwang, Kao-Shing, Wei-Cheng Jiang and Yu-Jen Chen, 2012. Tree-based Dyna-Q agent,in Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on, pp: 1077-1080.
7. Hwang, K.S., H.Y. Lin, Y.P. Hsu and H.H. Yu, 2011. Self-organizing state aggregation for architecture design of Q-learning,Inf. Sci., 181(13): 2813-2822.
8. Hester, T. and P. Stone, 2011. Learning and using models, in Reinforcement Learning: State of the Art, M. Wiering and M. van Otterlo, Eds. Berlin, Germany: Springer Verlag.
9. Viet, H.H., S.H. An and T.C. Chung, 2011. Extended Dyna-Q algorithm for path planning of mobile robots, J. Meas. Sci. Instrum., 2(3): 283-287.
10. Busoniu, L., R. Babuska and B. De Schutter, 2008. A comprehensive survey of multiagent reinforcement learning,IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., 38(2): 156-172.
11. Tateyama, T., S. Kawata and Y. Shimomura, 2007. Parallel reinforcement learning systems using exploration agents and Dyna-Q algorithm, in Proc. SICE Annu. Conf., Takamatsu, Japan, pp: 2774-2778.
12. Araabi, B.N., S. Mastoureshgh and M.N. Ahmadabadi, 2007. A study on expertise of agents and its effects on cooperative Q-learning, IEEE Trans. Syst., Man, Cybern. B, Cybern., 37(2): 398-409.
13. Asadpour, M., M.N. Ahmadabadi and R. Siegwart, 2006. Heterogeneous and hierarchical cooperative learning via combining decision trees,in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Beijing, China, pp: 2684-2690.
14. Amidan, B.G., T.A. Ferryman and S.K. Cooley, 2005. Data outlier detection using the Chebyshev theorem, in Proc. IEEE Aerosp. Conf., Big Sky, MT, USA, pp: 3814-3819.

15. Ito, K., Y. Imoto, H. Taguchi and A. Gofuku, 2004. A study of reinforcement learning with knowledge sharing-applications to real mobile robots, in Proc. IEEE Int. Conf. Robot. Biomimetics, Shenyang, China, pp: 175-180.
16. Ito, K., A. Gofuku, Y. Imoto and M. Takeshita, 2003. A study of reinforcement learning with knowledge sharing for distributed autonomous system, in Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom., pp: 1120-1125.
17. Ahmadabadi, M.N. and M. Asadpour, 2002. Expertness based cooperative Q-learning, IEEE Trans. Syst., Man, Cybern. B, Cybern., 32(1): 66-76.
18. Pyeatt, L.D. and A.E. Howe, 2001. Decision tree function approximation in reinforcement learning, in Proc. 3rd Int. Symp. Adapt. Syst. Evol. Comput. Probabilist. Graph. Models, pp: 70-77.
19. Uther, W.T.B. and M.M. Veloso, 1998. Tree based discretization for continuous state space reinforcement learning,in Proc. 15th Nat. Conf.Artif.Intell (AAAI-98), Madison, WI, USA, pp: 769-774.
20. Sutton, R.S. and A.G. Barto, 1998. Reinforcement Learning: An Introduction. Cambridge, U.K.: MIT Press.
21. Kuvayev, L. and R.S. Sutton, 1996. Model-based reinforcement learning with an approximate, learned model, in Proc. 9th Yale Workshop Adapt. Learn. Syst., New Haven, CT, USA, pp: 101-105.
22. Watkins, C.J.C.H. and P. Dayan, 1992. Technical note: Q-learning, Mach. Learn., 8(3-4): 279-292.