Conditional Q-learning Algorithm for Path-Planning of a Mobile Robot

Indrani Goswami (Chakraborty)[1], Pradipta Kumar Das[2] and Amit Konar[1] and R. Janarthanan[3]
[1]ETCE Department, Jadavpur University, Kolkata, India
[2]Dhaneswar Rath Institute of Engineering and Management Studies
Tangi, Cuttack, orissa
[3]Jaya Engineering College, Tamilnadu
{konaramit@yahoo.co.in , daspradipta78@gmail.com, srmjana_73@yahoo.com}

Abstract — **In classical Q-learning, the Q-table is updated after each state-transition of the agent. This is not always economic. This paper provides an alternative approach to Q-learning, where the Q-value of a grid is updated until a Boolean variable Lock associated with the cell is set. Thus the proposed algorithm saves unnecessary updating in the Q-table. Complexity analysis reveals that there is a significant saving in time- and space- complexity of the proposed algorithm with respect to the classical Q-learning**

Keywords—Q-learning; Reinforcement learning; Motion planning, Mobile Robots.

I. INTODUCTION

Machine learning is often used in mobile robots to make the robot aware about its world map. In early research on mobility management of robots, supervised learning was generally employed to train a robot to determine its next position in a given map from the sensory readings about the environment. Supervised learning is a good choice for mobility management of robots in fixed maps. However, if there is a small change in the robot's world, the acquired knowledge is no longer useful to guide the robot to select its next position. A complete training of the robot with both old and new sensory data-action pairs is then needed to overcome the said problem.

Reinforcement learning is an alternative learning policy, which rests on the principle reward and punishment. No prior training instances are presumed in reinforcement learning. A learning agent here does an action on the environment, and receives a feedback from the environment based on its action. The feedback provides an immediate reward for the agent. The learning agent here usually adapts its parameter based on the current and cumulative (futuristic) rewards. Since the exact value of the futuristic reward is not known, it is guessed from the knowledge about the robot's world map. The primary advantage of reinforcement learning lies in its inherent power of automatic learning even in presence of small changes in the world map.

Motion planning is one of the important tasks in intelligent control of a mobile robot. The problem of motion planning is often decomposed into path planning and trajectory planning. In path planning, we need to generate a collision-free path in an environment with obstacles and optimize it with respect to some criterion [8], [9]. However, the environment may be imprecise, vast, dynamical and partially non-structured [7]. In such environment, path planning depends on the sensory information of the environment, which might be associated with imprecision and uncertainty. Thus, to have a suitable planning scheme in a cluttered environment, the controller of such kind of robots must have to be adaptive in nature. Trajectory planning, on the other hand, considers time and velocity of the robots, while planning its motion to a goal. It is important to note that path planning may ignore the information about time/velocity, and mainly considers path length as the optimality criterion. Several approaches have been proposed to address the problem of motion planning of a mobile robot. If the environment is a known static terrain and it generates a path in advance, it is said to be off-line algorithm. It is called on-line, if it is capable of producing a new path in response to environmental changes.

Many studies have been carried out on path planning for different types of mobile robots. The works in [5] involve mapping, navigation, and planning tasks for Khepera II mobile robot. In these works, the computationally intensive tasks, such as the planning and mapping tasks were not performed onboard. They were performed on a separate computer. The sensor readings and the motor commands are communicated between the robot and the computer via serial connection.

Usually, the planning involves an action policy to reach a desired goal state, through maximization of a value function [1]-[3], which designates sub-objectives and helps choosing the best path. For instance, the value function could be the shortest path, the path with the shortest time, the safest path, or any combination of different sub-objectives. The definition of a task in this class may contain, besides the value function, some *a priori* knowledge about the domain, e.g., environmental map, environmental dynamics, goal position. Such knowledge allows a robot planning, while the lack of such knowledge obliges the robot either to learn it previously or to make use of heuristic strategies, such as moving to goal direction while avoiding obstacles.

Our research applies reinforcement learning techniques to real-world robots. Reinforcement learning has been tested in many simulated environments [3]- [11] but on a limited basis in real-world scenarios. A real-world environment poses more challenges than a simulated environment, such as enlarged state spaces [2], increased computational complexity, significant safety issues (a real robot can cause real damage), and longer turnaround times for results. This research measures how well reinforcement-learning

978-1-4244-8546-8/10/$26.00 ©2010 IEEE

technique, such as Q-learning, can apply to the real robot for navigational problem, and in our research we modified the classical Q-learning algorithm, hereafter called conditional Q-learning for increasing the performance of the classical one in the navigation problem.

The rest of the paper is organized as follows. Classical Q-learning is introduced in section II. Some properties of the Q-learning based on the concept of locking of states are derived in section III. The algorithm for the extended Q-learning is given in section IV. Experimental details are included in section V. Conclusions are listed in section VI.

II. THE CLASSICAL Q-LEARNING

In classical Q-learning, all possible states of an agent and its possible action in a given state are deterministically known. In other words, for a given agent A, let S_0, S_1, S_2,..., S_n, be n- possible states, where each state has m possible actions a_0, a_1, a_2,...,a_m. At a particular state-action pair, the specific reward that the agent acquires is known as immediate reward. For example $r(S_i, a_j)$ denotes the immediate reward that the agent A acquires by executing an action a_j at state S_i. An agent selects its next state from its current states by using a policy. This policy attempts to maximize the cumulative reward that the agent could have in subsequent transition of states from its next state. For example, let the agent be in state Si and is expecting to select the next best state. Then the Q-value at state S_i due to action of a_j is given in equation 1.

$$Q(S_i, a_j) = r(S_i, a_j) + \gamma \underset{a'}{Max}\, Q(\delta(S_i, a_j), a') \quad (1)$$

where $\delta(S_i, a_j)$ denotes the next state due to selection of action a_j at state S_i. Let the next state selected be S_k. Then $Q(\delta(S_i, a_j), a') = Q(S_k, a')$. Consequently selection of a' that maximizing $Q(S_i, a_j)$ is an interesting problem. One main drawback for the above Q-learning is to know the Q-value at a state S_k for all possible action a'. As a result, each time it accesses the memory to get Q-value for all possible actions at a particular state to determine the most appropriate next state. So it consumes more time to select the next state. Since the action a' for which $Q(S_k, a')$ is maximum needs to be evaluated, we can remodel the Q-learning equation by identifying the a' that moves the agent closer to the goal.

In the extended Q-learning to be presented, we have created only one field for action of each state. In this way, we save the space required for the Q-table. Thus the Q-table storing the Q-values for the best action in each state, requires small time for retrieval of Q-values, and hence saves significant time complexity.

III. PROPERTIES OF THE EXTENDED Q-LEARNING

Let for any state S_k, the distance between the goal state and the next feasible state of S_k are known. Let the next feasible state of S_k be $S \in \{S_a, S_b, S_c, S_d\}$. Let G be the goal and the distance between S_a, S_b, S_c, S_d and G be d_{aG}, d_{bG}, d_{cG} and d_{dG} respectively. Let the distance in order be $d_{bG} < d_{aG} < d_{cG} < d_{dG}$. Then the agent should select the next state S_b from its current state S_k. If the Q-value of the state S_b is known. We can evaluate the Q-value of state S_k by the following approach.

$$Q(S_k, a') = r(S_k, a') + \gamma \underset{a'}{Max}\; Q(\delta(S_k, a'), a'')$$
$$= 0 + \gamma \underset{a'}{Max}\, Q(\delta(S_k, a'), a'') \quad (2)$$

Now $\delta(S_k, a') = S_a \mid S_b \mid S_c \mid S_d$, where $\mid$ denotes OR operator. Therefore,

$$\underset{a'}{Max} Q(\delta(S_k, a'), a'')$$
$$= \underset{a'}{\text{Max}}\, Q\{S_a \mid S_b \mid S_c \mid S_d, a''\}$$
$$= Q(S_b, a''). \quad (\because d_{bG} < d_{aG} < d_{cG} < d_{dG})$$

Combining (2) and (3) we have: (3)

$$Q(S_k, a') = 0 + \gamma Q(S_b, a'').$$

Thus if the next state having the shortage distance with the goal is known, and the Q-value of this state is also known, then the Q-value of the current state is simply γ×Q-value of the next state.

Let S_p, S_n and S_G be the present, next and the goal states respectively. Let Q_p and Q_n be the Q-value at the present and the next states S_p and S_n respectively. Let d_{xy} be the Euclidean distance between the states S_x and S_y. We use a Boolean variable Lock: L_x to indicate that the Q_x value of a state is fixed permanently. We set lock L_n=1 if the Q-value of the state n is fixed, and won't change further after L_n is set to 1. The Lock variable for all states except the goal will be initialized zero in our proposed Q-learning algorithm. We observe four interesting properties as indicated below.

Property 1: If L_n=1 and $d_{pG} < d_{nG}$ then $Q_p = \gamma \times Q_n$ and set $L_p = 1$.

Proof. Let the neighborhood state of S_p be $S \in \{S_a, S_b, S_c, S_n\}$, and the agent selects S_n as the next state as $d_{nG} < d_{xG}$ for $x \in \{a, b, c, n\}$.

Now,

$$Q_p = Q(S_p, a)$$
$$= r(S_p, a) + \gamma \underset{a'}{Max}\, Q(\delta(S_p, a), a')$$
$$= 0 + \gamma \underset{a}{Max} Q(S_a \mid S_b \mid S_c \mid S_n, a')$$
$$= \gamma \times Q(S_n, a') \quad (\because d_{nG} \le d_{xG} \text{ for}$$

$x \in \{a, b, c, n\}$)

$= \gamma \times Q_n$.

Since $L_n = 1$, and $d_{pG} > d_{nG}$, $\therefore Q_p < Q_n$, and thus $Q_p = \gamma \times Q_n$ for $0 < \gamma < 1$ is the largest possible value of Q_p, and Q_p should not be updated further. So, $L_p = 1$ is set.

Property 2: If $L_n=0$ and $d_{pG} > d_{nG}$ then $Q_p = \text{Max}(Q_p, \gamma \times Q_n)$.

Proof. Let the neighborhood of state S_p be $S \in \{S_a, S_b, S_c, S_n\}$, and the agent selects S_n as the next state as $d_{nG} \le d_{xG}$ for $x \in \{a, b, c, n\}$.

Now,

$$\begin{aligned} Q_p &= Q(S_p, a) \\ &= r(S_p, a) + \gamma \underset{a'}{Max} Q(\delta(S_p, a), a') \\ &= 0 + \gamma \underset{a'}{Max} Q(S_a \mid S_b \mid S_c \mid S_n, a') \\ &= \gamma \times Q(S_n, a') \ (\because d_{nG} \le d_{xG} \text{ for } x \in \{a, b, c, n\}) \\ &= \gamma \times Q_n. \end{aligned} \quad (4)$$

Since $d_{pG} > d_{nG}$, $Q_p < Q_n$. So, $Q_p = \gamma \times Q_n$ is logically satisfied. However, as $L_n=0$, Q_n is not yet stable at the current iteration. So, Q_p is also not stable yet. Let t be the current iteration. Then

$$\left.\begin{aligned} Q_p(t) &= \gamma \times Q_n(t) \text{ if } Q_p(t-1) \le \gamma \times Q_n(t). \\ &= Q_p(t-1), \text{ otherwise.} \end{aligned}\right\} \quad (5)$$

The two alternatives in (5) can be put together as in (6).

$$Q_p(t) = Max(Q_p(t-1), \gamma Q_n(t)) \quad (6)$$

Ignoring the notion of time t from (6), we obtain $Q_p(t) = Max(Q_p, \gamma \times Q_n)$. □

Property 3: If $L_p = 1$ and $d_{nG} > d_{pG}$ then $Q_n = \gamma \times Q_p$ and set $L_n = 1$.

Proof. Let the neighborhood of the next state S_n be $S \in \{S_a, S_b, S_c, S_p\}$. Since $d_{pG} \le d_{xG}$ for $x \in \{a, b, c, p\}$, the agent will select the state S_p during its transition form S_n.

Now, $Q_n = Q(S_n, a)$

$$\begin{aligned} &= r(S_n, a) + \gamma \underset{a'}{Max} Q(S_a \mid S_b \mid S_c \mid S_p, a') \\ &= 0 + \gamma \underset{a'}{Max} Q(S_a \mid S_b \mid S_c \mid S_p, a') \\ &= \gamma \times Q(S_p, a') \ (\because d_{pG} \le d_{xG}, \forall x). \\ &= \gamma \times Q_p. \end{aligned}$$

Since $d_{nG} > d_{pG}$, $Q_n < Q_p$. So, $Q_n = \gamma \times Q_p$ is logically acceptable for $0 < \gamma < 1$. Further, as $L_p=1$, and S_n is the nearest state to S_p with respect to given distance metric, therefore L_n is set to 1. □

Property 4: *If $L_p=0$* and $d_{nG} > d_{pG}$ then $Q_n = \text{Max}(Q_n, \gamma \times Q_p)$.

Proof. Let the neighborhood of state S_n be $S \in \{S_a, S_b, S_c, S_p\}$. Let $d_{pG} > d_{xG}$ for $x \in \{a, b, c, p\}$. Then the agent will select S_p during its transition from S_n,

Now, $Q_n = Q(S_n, a)$

$$\begin{aligned} &= r(S_n, a) + \gamma \underset{a'}{Max} Q(\delta(S_n, a), a') \\ &= r(S_n, a) + \gamma \underset{a'}{Max} Q(S_a \mid S_b \mid S_c \mid S_p, a') \\ &= 0 + \gamma \times Q(S_p, a') \ (\because d_{pG} \le d_{xG}, \forall x) \\ &= \gamma \times Q_p. \end{aligned}$$

Since $d_{nG} > d_{pG}$, $\therefore Q_n < Q_p$. So, $Q_n = \gamma \times Q_p$ for $0 < \gamma < 1$ is logically satisfactory. Now, as $L_p=0$, Q_p is not fixed until this iteration; So Q_n is also not fixed in the current iteration. Now, adding the notion of iteration t in Q_n and Q_p, we have;

$$\left.\begin{aligned} Q_n(t) &= \gamma \times Q_p(t), \text{ if } Q_n(t-1) < \gamma \times Q_p(t), \\ (5) \quad &_n(t-1), \text{otherwise.} \end{aligned}\right\} \quad (7)$$

Combining the expressions under (7) by a single expression, we write:

$$Q_n(t) = Max(Q_n(t-1), \gamma \times Q_p(t)) \quad (8)$$

Now, eliminating t from both sides of (8), we obtain,

$$Q_n = Max(Q_n, \gamma \times Q_p). \quad \square$$

IV. The Extended Q-Learning Algorithm

In this section we proposes a new Q-learning algorithm, called Conditional-Q learning. The algorithm has three main steps. The first two steps initialize Lock variables for the states, γ and the Q-table. The updating of the Q-table is done in step 3 using the properties introduced in section III.

Pseudo Code for Conditional Q-learning

1. Initialization
   For all $S_i, i = 1$ to n except $S_i = S_G$
   {set $L_i = 0; Q_i = 0;$ action= φ};
   L_G (for goal S_G) = 1;
   Q_G (for goal)= 100;
2. Assign γ in (0,1) and initial state= S_p;
3. Update Q– table:
   Repeat
   { a) Select a_i from A = {$a_1, a_2, \ldots, a_m$};
   b) Determine d_{nG} and d_{pG};
   If ($d_{nG} < d_{pG}$)
   Then if $L_n = 1$
   Then{$Q_p = \gamma \times Q_n; L_p = 1;$
   save action of S_p = "a_i"}
   Else if ($Q_p > \gamma \times Q_n$) then
   {save $Q_p = \gamma \times Q_n$ and action of S_p = "a_i"};
   Else if $L_p = 1$ then{$Q_n = \gamma \times Q_p$;
   save action of S_n = opposite action
   of "a_i"; $L_n = 1$};
   Else if ($Q_n > \gamma \times Q_p$) then
   {$Q_n = \gamma \times Q_p$ and
   save action of S_n = opposite action of "a_i"}
   }
   Until $L_i = 1$ for all i;

Time-Complexity

In classical Q-learning, the updating of Q-values in a given state requires determining the largest Q-value, in that cell for all possible actions. Thus if there are m possible actions at a given state, maximization of m possible Q-values, require m –1 comparison. Consequently, if we have n states, the updating of Q values of the entire Q table by classical method requires n(m –1) comparisons. Unlike the classical case, here we do not require any such comparison to evaluate the Q values at a state S_p from the next state S_n. But we need to know whether state n is locked i.e., Q-value of S_n is permanent and stable. Thus if we have n number of states, we require n number of comparison. Consequently, we save n(m –1) – n = nm – 2n= n(m–2).

Space-Complexity

In classical Q-learning, if we have n states and m action per state, then the Q-table has a dimension $(m \times n)$. In the conditional Q-learning, we only store Q-value at a state for the best action. Further, we need to store whether the Q-value is already stable or changing. Naturally, we need to store the status of lock variable for each state. Consequently, for each state we require 3 storages, such as Q-value, lock and the best action at that state. Thus for n number of states, we require $(3 \times n)$ number of storage. The saving in memory in the present context with respect to classical Q thus is given by mn – 3n = n(m – 3).

V. EXPERIMENTS WITH KHEPERA II ROBOT

Khepera II (Figure 1) is a miniature robot (diameter of 8cm) equipped with 8 built-in infrared range and light sensors, and 2 relatively accurate encoders for the two motors. The range sensors are positioned at fixed angles and have limited range detection capabilities. We numbered the sensors clockwise from the leftmost sensor to be sensor 0 to sensor 7(Figure 2). Sensor values are numerical ranging from 0 (for distance > 5 cm) to 1023 (approximately 2 cm).The onboard Microprocessor has a flash memory size of 256KB, and the CPU of 8 MHz. Khepera can be used on a desk, connected to a workstation through a wired serial link. This configuration allows an optional experimental configuration with everything at hand: the robot, the environment and the host computer.

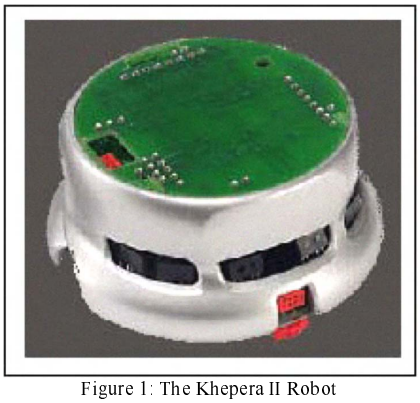

Figure 1: The Khepera II Robot

Snapshots of the planning steps realized on Khepera II are given in Figure 3 below. The extended Q-learning algorithm is used in the first phase to learn the movement steps from each grid in the map to its neighbor. After the learning phase is over, the planning algorithm is executed with the snapshots as indicated in the Figure 3.

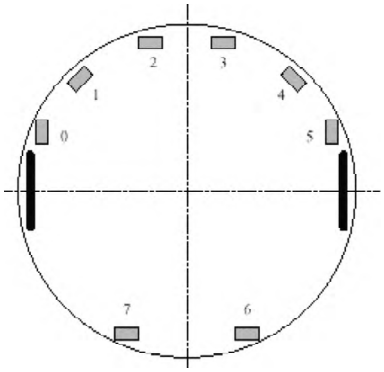

Figure 2: Position of the sensors of Khepera II

.

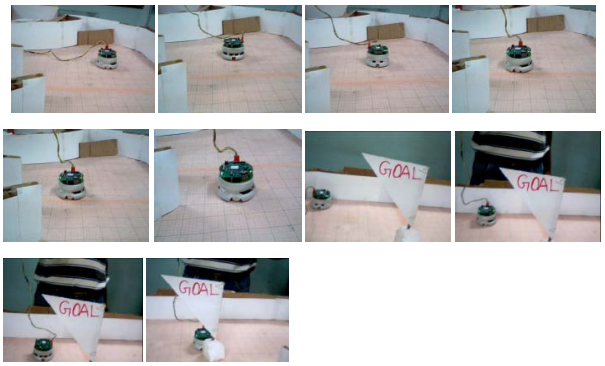

Figure 3. Snapshots of experimental planning instances in order.

VI. CONCLUSIONS

The paper presented an extension of classical Q-learning algorithm, so as to reduce both space and time-complexity by economically updating the Q-table, only when such updating is mandatory. The economic selection of specific entries of the Q-table is performed by certain properties of the extended Q-learning algorithm. The claim of this paper is four fundamental properties, and utilization of these properties to set conditional checking on updating of Q-table entries. This is realized in a pseudo code given in the paper. The reduction in both space-and time-complexity is indicated. Verification of the algorithm has been performed on Khepera II platform.

REFERENCES

[1] Dean, T., Basye, K. and Shewchuk, J. "Reinforcement learning for planning and Control". In: Minton, S. (ed.) Machine Learning Methods for Planning and Scheduling. Morgan Kaufmann 1993.

[2] Bellman, R.E., "Dynamic programming ", Princeton, NJ: Princeton University Press, p. 957.

[3] Watkins, C. and Dayan, P., "Q-learning," Machine Learning, Vol. 8, pp. 279-292, 1992

[4] Konar, A., Computational Intelligence: Principles, Techniques and Applications. Springer-Verlag, 2005

[5] Busoniu, L., Babushka, R., Schutter, B.De., Ernst, D., *Reinforcement Learning and Dynamic Programming Using Function Approximators*, CRC Press, Taylor & Francis group, Boca Raton, FL, 2010.

[6] Chakraborty, J., Konar A., Jain, L.C., and Chakraborty, U., "Cooperative Multi-Robot Path Planning Using Differential Evolution" *Journal of Intelligent & Fuzzy Systems*, Vol. 20, Pp.13-27, 2009.

[7] Gerke, M., and Hoyer, H., Planning of Optimal paths for autonomous agents moving in inhomogeneous environments, in: *Proceedings of the 8th Int. Conf. on Advanced Robotics*, July 1997, pp.347-352.

[8] Xiao, J., Michalewicz, Z., Zhang, L., and Trojanowski, K., Adaptive Evolutionary Planner/ Navigator for Mobile robots, *IEEE Transactions on evolutionary Computation* 1 (1), April 1997.

[9] Bien, Z., and Lee, J., A Minimum–Time trajectory planning Method for Two Robots, *IEEE Trans on Robotics and Automation* 8(3).PP.443-450, JUNE 1992.

[10] Moll, M., and Kavraki, L.E., Path Planning for minimal Energy Curves of Constant Length, in: Proceedings of the 2004 *IEEE Int. Conf. on Robotics and Automation*, pp.2826-2831, April 2004.

[11] Regele, R., and Levi, P., Cooperative Multi-Robot Path Planning by Heuristic Priority Adjustment, in: Proceedings of the *IEEE/RSJ Int Conf on Intelligent Robots and Systems*, 2006.