Scientific Online

ARTIFICIAL INTELLIGENCE AND APPLICATIONS
ISSN(Print): 2374-4979 ISSN(Online): 2374-4987
Volume 1, Number 2, August 2014

ARTIFICIAL INTELLIGENCE AND APPLICATIONS

A Simulation Study on Reinforcement Learning for Navigation Application

Jaspreet Singh Bal, Nitaigour Premchand Mahalik*

Department of Industrial Technology, Jordan College of Agricultural Sciences and Technology California State University, Fresno, 2255 E Barstow Ave., California, 93740, USA.

***Corresponding author**: nmahalik@csufresno.edu

Abstract:
In this paper we have contributed work on implementation of Q-learning, a reinforcement, non-parameter based learning and decision making method. have formulated and demonstrated the Q-learning algorithm via simulation. The work includes formulation of a pseudo-code and development of algorithm taking into account of an application. The application of reinforcement learning is simulated through a conceptual agriculture field, where a robot is commanded to reach at trees and finally delivers the fruits to the storage point (goal). We have studied the effectiveness of γ and α. The results show that the learning parameter (γ) and the learning rate (α) are the two important parameters to be considered while developing Q-learning based reinforcement algorithm for specific application. We have also established the optimal values of γ and α of an application through a simulation study.

Keywords:
Re-enforcement Learning; Q-learning; Robot Navigation; Soft Computing, Automation and Control

1. BACKGROUND AND INTRODUCTION

Decision making is useful in organizing, planning, executing, and accomplishing a task. *Decision-making* has countless applications in many fields. The Markov decision process is solely based on the Markov property which states an action effect depends upon the current state. The actions in Markov decision process can be represented into two action forms known as deterministic and stochastic actions. In deterministic actions, a new state is defined for every action and state. The rewards obtained are summed up but the final result may be indistinct. The probability distribution for next states is specified for every action and state in the stochastic distribution. The focus on expected value and worst case allows a decision process to properly utilize approximation and sampling [1, 2]. Baye's theorem plays an important part in understanding the concept for the parameter decision making model. This theorem is useful in taking into account all the uncertainties connected with the modeling and parameter values. Maximum likelihood is solely responsible for the majority decision analyses, as fixing values for the parameters are involved. Bayesian approach has an advantage of implementing prior information as they specify the range of individual parameters [3–5]. Non-parameter based decision modeling shows a better performance in an artificial neural networks and decision trees. A statistical distribution is assumed

in this case, which is incompatible with multiple source data. The assumptions made in a distribution of data are not made in the non-parameter based decision model, thus avoiding any error [6, 7]. The reinforcement learning is a kind of a methodological approach applied to convert situations into actions and maximizing the reward points. The agent has to learn by itself, how to perform actions and which actions are going to fetch the most rewards. For that, we need to specify the preferences, record of actions, and specific time to perform these actions [8, 9]. The reinforcement learning have great applications in board games played by a computer, elevator control, network routing, data mining, robotic control, speech recognition, bioinformatics, and web and text data processing. It informs an agent when it is losing the rewards and suggests other alternative ways to capture a win [10]. This paper extensively studies the usefulness of the reinforcement learning algorithm. In particular, Q-learning algorithm has been studied and suggestions are made on the optimal values of the parameters involved. The results have been shown through a simulation.

2. REVIEW ON REINFORCEMENT AND Q-LEARNING

Reinforcement learning is a concept of the machine learning where an agent is given the directions to perform and attain the goal in an optimal way. Reinforcement learning has very many applications. Many reinforcement algorithms are correlated to the dynamic programming techniques. The reinforcement learning algorithms do not require information. This learning is different from the standard supervised learning in that it works in the absence of the inputs. It is concerned with unknown territory investigation and current knowledge utilization. A reinforcement learning model describes itself as (i) a set of actions A, (ii) a set of environment states B, (iii) rule describing the observation of agent, (iv) rule determining scalar instant reward change, and (v) rule concerning switching between states. Generally, random rules are observed in the reinforcement learning. It works based on the principle that the last transition is given an instant reward as per an observation. The picture of the current state of the environmental is fully observed by an agent. The agent scores as many reward points as possible. A reward r(t) is received by an agent after doing an observation o(t). An action a(t) is selected from a set of actions. So, action map is needed. The action map is around the surrounding or environment. The environment actually moves the agent to a new state with a new reward r (t+1). This action can be performed as a function of history or completely randomization of its selection. The agent that acts optimal way compares its earlier performance. The agent works in an optimal fashion and considers its long term results [11]. The intensity of reinforcement depends on the size of the environment and precise description. Expected values of actions in each state are continually updated. Each action is given a value for every feasible state, which depends upon both the immediate reward for performing the action and expected reward based on new state.

The updating of Q value is carried out by using $Q(s, a) = Q(s, a) + \alpha [r + \gamma \max a' Q (s', a')-Q(s, a)]$. Where, a represent action vector, r as reward, s as state vector, γ is called discount factor and α is the learning rate for controlling convergence. The objective is that the Q-values suggest choosing of action a that maximize state Q(s, a). Thus, the function Q(s, a) is incrementally learned to evaluate the ease of performing action a in state s. If the immediate reward r(s, a) and state s' is achieved when action a is accomplished at that stage, the updated Q-value will become $Q(s, a) = r(s, a) +\gamma \max Q (s', a')$. The factor γ determines whether or not the reward should come earlier than others. Its value lies between 0-1. Q-Learning can be applied in multi-agent coordination system.

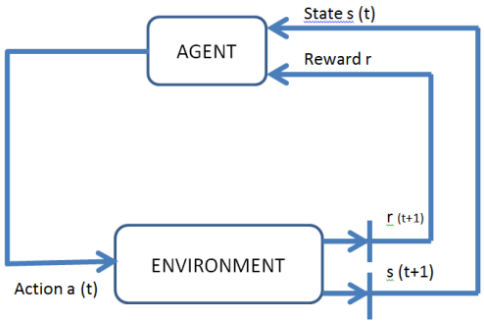


Figure 1. The graphical representation of reinforcement learning

3. IMPLEMENTATION OF Q-LEARNING ALGORITHM

In this paper we have contributed work on implementation of reinforcement learning, a non-parameter based decision making method. Q-Learning technique, a representative of reinforcement learning is considered in this study. We have formulated and demonstrated the Q-learning algorithm via simulation. The work includes formulation of a pseudo-code and analytical derivations. The results show that the learning parameter (γ) and the learning rate (α) are the two important parameters to be considered while developing Q-learning based reinforcement algorithm for specific application. We have established the optimal values of γ and α of an application through a simulation study. The application of reinforcement learning is simulated through a conceptual agriculture field, where a robot is commanded to reach the tree and finally delivers to the storage point. The robot, an agent is a learner and knows nothing about the path. Without any supervision, the agent will learn step by step for reaching the orange trees from the nine different trees. In the sequel, (i) pseudo-code, (ii) modeling of the environment, and (iii) state, agent and action are presented.

Pseudo-Code: The pseudo-code for the Q-learning is outlined below [12, 13].

Initialization: Set Q(S, a) = 0 for all S and a

Begin

Determine current state S

Select an action a at state S and execute it

Determine immediate reward r

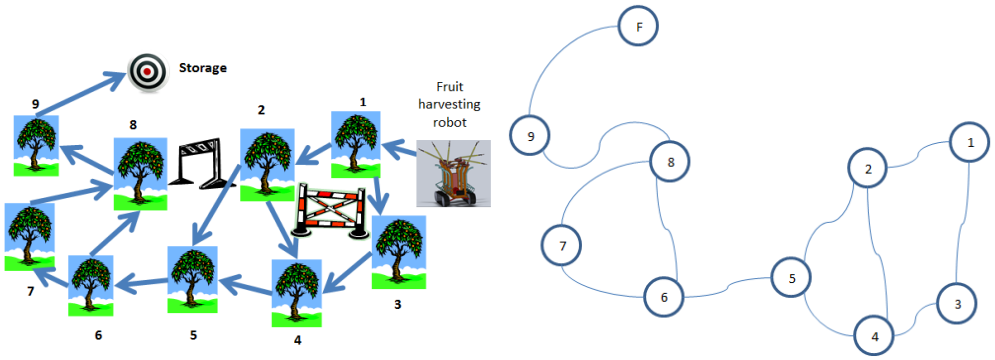
Find the new state S'

Update Q(S, a) by the following expression

$$Q(S,a) = r + \gamma.max(Q(\delta(S,a),a'))$$

S=S'

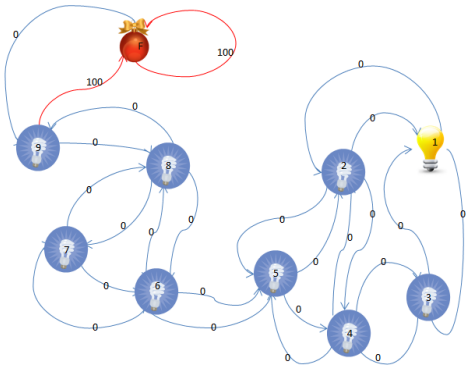
Modeling the Environment: Suppose, there are nine orange trees present in an orange field as shown in the Figure 9. A robot is employed to harvest oranges from the trees. The arrows indicate the path to be followed by the robot from one tree to another. Each tree has been numbered from 1 to 9 and target represents the storage point (goal). Notice that there is only one tree which leads to the destination point.



(a) An orange harvesting field comprising of nine trees from 1 to 9 and the future state F

(b) Nodal graph with states 1 to 9 and the future state F

Figure 2. The above figure shows an orange harvesting field comprising of nine trees from 1 to 9 and the future state F; (b) This figure depicts nodal graph with states 1 to 9 and the future state F



(a) Nine different states along with the future storage point

	1	2	3	4	5	6	7	8	9	F
1	X	0	0	X	X	X	X	X	X	X
2	0	X	X	0	0	X	X	X	X	X
3	0	X	X	0	X	X	X	X	X	X
4	X	0	0	X	0	X	X	X	X	X
5	X	0	X	0	X	0	X	X	X	X
6	X	X	X	X	0	X	0	0	X	X
7	X	X	X	X	X	0	X	0	X	X
8	X	X	X	X	X	0	0	X	0	X
9	X	X	X	X	X	X	X	0	X	100
F	X	X	X	X	X	X	X	X	0	100

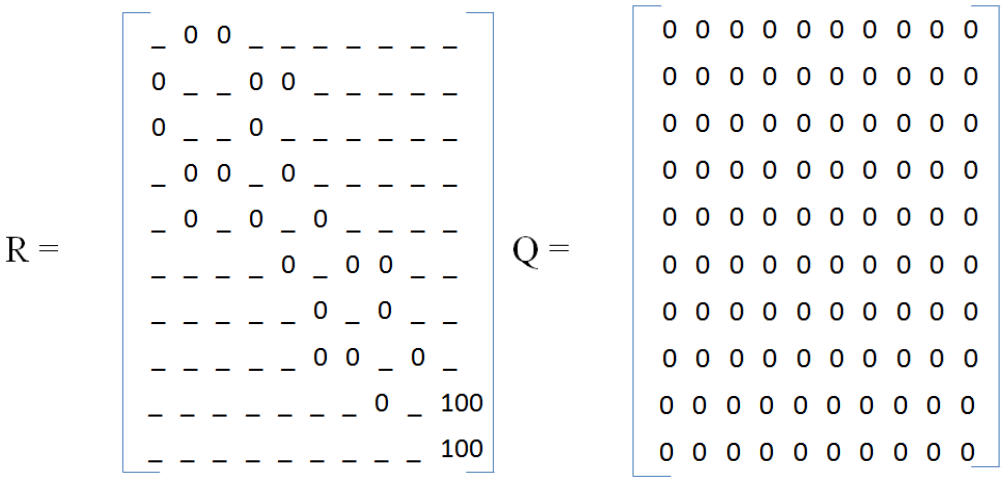
(b) Matrix R with rows as past and columns as the future states

Figure 3. The state diagram represents nine different states along with the future storage point; Figure 12. The table represents Matrix R with rows as past and columns as the future states

The following figure can also be shown by a graph, each tree as a node or vertex and path by a link. The robot is the agent set near to the first orange tree.

State, Agent and Action: Suppose the agent knows nothing about an environment, but it can learn through experience. In this case, the agent is an orange collecting robot. It is not having any idea regarding the sequence of the path to be followed. The agent is currently present at number one tree and wants to reach the storage point marked as a target. Let us define state and action. The agent reaches different trees from 1 to 9. Each tree is represented by a node in the nodal graph as shown above and is referred to as state. The arrows represent the action followed by an agent. The following state diagram shows the reward of 100 given immediately if the goal point is reached. The agent cannot reach the storage point directly from any other state and thus have zero reward. An instant reward value is enclosed by each arrow. The agent is rewarded with an additional 100 points, which encourages it to remain in that state forever.

Suppose, now the agent is situated at 1 and wants to reach the final point F. However, from state 1, the agent cannot go directly to state 4, 5, 6, 7, 8, 9 or final point F as there is no direct connection between them. From state 2 the agent can either go to state 4, 5 or back to state 1. If the agent is in state 5, the



(a) The R matrix (b) The Q matrix

Figure 4. (a) The R matrix, (b) The Q matrix

three possible actions are going to state 6, 2 or 4. Similarly, the arrows going in and out indicate their path between the states. Only two rewards are shown in the **Figure 3**. If the agent goes from state 9 to F or stays forever at F, it receives a reward of 100 points. The state diagram can be converted to the reward table or matrix R, with the current and future position of an agent. The cross sign represents the unavailability of the path from one to another state [12, 13]. The values 0 and 100 denote the reward points gained by an agent going from one position to another. The cross or dash values indicate that an agent cannot go from one state to another due to the presence of a barrier (**Figure 4(a)**). As we proceed further, an environmental reward matrix is designed from the above matrix R table. That is we need to compute and update a new matrix Q for the agent which will guide the agent throughout the task. Similar to R matrix, the column symbolize the future state attained from the agent's current state in row. The agent knows nothing about traversing the path. Therefore, the initial value of Q is a zero matrix (**Figure 4(b)**). It is assumed that the agent will take more time on a bigger tree than others. The barriers help in preventing the agent to go to other state, which is not stated in the algorithm. The following equation shows the determination of the Q values [12, 13].

$$Q(state,\ action) = \alpha(R\ (state,\ action) + \gamma\ .Max[Q(next\ state,\ all\ actions]) \tag{1}$$

The bracket values represent the state as rows and actions as columns. The learning parameter γ in this case is considered to be 0.75 and chosen depending upon such factors as the needed efficiency of an agent and the extent to which newly gained information can override the old one. The agent can understand the most recent information if learning factor is one, but cannot perform the task if the learning factor becomes zero.

Learning: Assume that the initial state is 1 with learning parameter $\gamma = 0.75$ and learning rate $\alpha = 0.1$. Initially, the Q matrix is put to zero.

Assume that the agent is at state 1 and can go only to state 2 or 3. Randomly choose state 2 as our action. Consider that the agent to be at state 2. From here, the agent can go to the state 1, 4 or 5 which can be written as follows.

$$Q(1,\ 2) = 0.1(R\ (1,\ 2) + \ 0.75.\text{Max}\left[Q(2,\ 1), Q(2,4), Q(2,5)\right]\ ) = 0 \tag{2}$$

Since, matrix Q and value of R (1, 2) is zero, the value of Q (1, 2) comes out to be zero. Now, let us determine the other values of Q which can be written as follows.

$$Q(1,\ 3) = 0.1(R\ (1,\ 3) + \ 0.75.\text{Max}\left[Q(3,\ 1), Q(3,4)\right]\ ) = 0 \tag{3}$$

This finishes first event of the agent's learning which started from the starting point to a second state. The agent can either go to state 2 or 3 from 1. Similarly, we can find out the Q values of other states coming in the path (see below).

$$Q(2,1) = 0.1(R\ (2,1) + \ 0.75.\text{Max}\left[Q(1,2), Q(1,3)\right]\ ) = 0 \tag{4}$$

$$Q(2,4) = 0.1(R\ (2,4) + \ 0.75.\text{Max}\left[Q(4,2), Q(4,3), Q(4,5)\right]\ ) = 0 \tag{5}$$

$$Q(2,5) = 0.1(R\ (2,5) + \ 0.75.\text{Max}\left[Q(5,2), Q(5,4), Q(5,6)\right]\ ) = 0 \tag{6}$$

$$Q(3,1) = 0.1(R\ (3,1) + \ 0.75.\text{Max}\left[Q(1,2), Q(1,3)\right]\ ) = 0 \tag{7}$$

$$Q(3,4) = 0.1(R\ (3,4) + \ 0.75.\text{Max}\left[Q(4,2), Q(4,3), Q(4,5)\right]\ ) = 0 \tag{8}$$

$$Q(4,2) = 0.1(R\ (4,2) + \ 0.75.\text{Max}[Q(2,1), Q(2,4), Q(2,5)]) = 0 \tag{9}$$

$$Q(4,3) = 0.1(R\ (4,3) + \ 0.75.\text{Max}\left[Q(3,1), Q(3,4)\right]\ ) = 0 \tag{10}$$

$$Q(4,5) = 0.1(R\ (4,5) + \ 0.75.\text{Max}\left[Q(5,2), Q(5,4), Q(5,6)\right]\ ) = 0 \tag{11}$$

$$Q(5,2) = 0.1(R\ (5,2) + \ 0.75.\text{Max}[Q(2,1), Q(2,4), Q(2,5)]) = 0 \tag{12}$$

$$Q(5,4) = 0.1(R\ (5,4) + \ 0.75.\text{Max}\left[Q(4,2), Q(4,3), Q(4,5)\right]\ ) = 0 \tag{13}$$

$$Q(5,6) = 0.1(R\ (5,6) + \ 0.75.\text{Max}[Q(6,5), Q(6,7), Q(6,8)]) = 0 \tag{14}$$

$$Q(6,5) = 0.1(R\ (6,5) + \ 0.75.\text{Max}\left[Q(5,2), Q(5,4), Q(5,6)\right]\ ) = 0 \tag{15}$$

$$Q(6,7) = 0.1(R\ (6,7) + \ 0.75.\text{Max}[Q(7,6), Q(7,8)]) = 0 \tag{16}$$

$$Q(6,8) = 0.1(R\ (6,8) + \ 0.75.\text{Max}\left[Q(8,6), Q(8,7), Q(8,9)\right]\ ) = 0 \tag{17}$$

$$Q(7,6) = 0.1(R\ (7,6) + \ 0.75.\text{Max}\left[Q(6,5), Q(6,7), Q(6,8)\right]\ ) = 0 \tag{18}$$

$$Q(7,8) = 0.1(R\ (7,8) + \ 0.75.\text{Max}[Q(8,6), Q(8,7), Q(8,9)]) = 0 \tag{19}$$

$$Q(8,6) = 0.1(R\ (8,6) + \ 0.75.\text{Max}\left[Q(6,5), Q(6,7), Q(6,8)\right]\ ) = 0 \tag{20}$$

$$Q(8,7) = 0.1(R\ (8,7) + \ 0.75.\text{Max}[Q(7,6), Q(7,8)]) = 0 \tag{21}$$

$$Q(8,9) = 0.1\,(R\ (8,9) + \ 0.75.\text{Max}\left[Q(9,8), Q(9,F)\right]\ ) = 0.1(0 + 0.75 * 100) = 7.5 \tag{22}$$

$$Q(9,8) = 0.1(R\ (9,8) + \ 0.75.\text{Max}\left[Q(8,6), Q(8,7), Q(8,9)\right]\ ) = 0 \tag{23}$$

$$Q(9,F) = 0.1(R\ (9,F) + \ 0.75.\text{Max}\left[F,F\right])\ = 100 + 0.75 * 100 = 100 + 75) = 17.5 \tag{24}$$

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 17.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 17.5 \end{bmatrix} \quad Q = \begin{bmatrix} 0 & 54 & 37 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 37 & 0 & 0 & 54 & 78 & 0 & 0 & 0 & 0 & 0 \\ 37 & 0 & 0 & 54 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 54 & 37 & 0 & 78 & 0 & 0 & 0 & 0 & 0 \\ 0 & 54 & 0 & 54 & 0 & 112 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 78 & 0 & 112 & 161 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 112 & 0 & 161 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 112 & 112 & 0 & 231 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 161 & 0 & 331 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 231 & 331 \end{bmatrix}$$

(a) After few iterations (b) after 1 Billions of iterations

Figure 5. Updating of Q matrix through episodes

$$Q(F,F) = 0.1(R\ (F,F) + 0.75.\text{Max}\,[Q(F,F)]) = 17.5 \tag{25}$$

As noted, the Q-values of an agent's states and actions after passing through the number of events are determined by putting the values in equation (1). By updating the new Q-values to our agent, a new Q matrix is obtained (**Figure 5**). This is called updating of episode. **Figure 5(b)** is Q matrix that is obtained, after the agent learns through 1 billions iterations or episodes. Since the highest reward was assumed to be 100 initially, all the valid entries are divided by the highest value 331 and then multiplied by 100, resulting into the state diagram. The reward located at the end makes harder for the user to update the values, after each move made by the robot. To overcome this limitation, the whole set of rewards are saved and updated in an opposite order at the end. The variations can be handled in an alternate technique, where past actions of the state and their rewards are stored periodically [7b].

In this paper we have studied the effectiveness of γ. The map is a q-learning map or state diagram (**Figure 6**) with $\alpha = 0.1$, $\gamma = 0.75$. For different values of α and γ a different state diagram will be obtained for the same number of iteration for example. The effect of chosen values of α and γ has been showed in **Figure 7**.

4. DISCUSSION

In this section brief discussion of important points related to this research is presented. The discussion covers (i) merits and demerits of reinforcement learning, (ii) Learning parameter γ value, and (iii) Problem with understanding and realizing the pseudo algorithm.

Reinforcement learning is a form of machine learning in which tradeoff between exploitation and exploration occurs [14]. The other two forms of the machine learning can either be supervised or unsupervised. The advantages of the reinforcement learning are (i) complete knowledge is not required for developing the algorithm for an application, (ii) it is simple and easy to implement in agents, (iii) algorithms for every task can be prepared by creating programs in Java, C++ or Microsoft excel file, and (iv) it can be applied to those applications where supervised learning is not relevant [15–19]. The

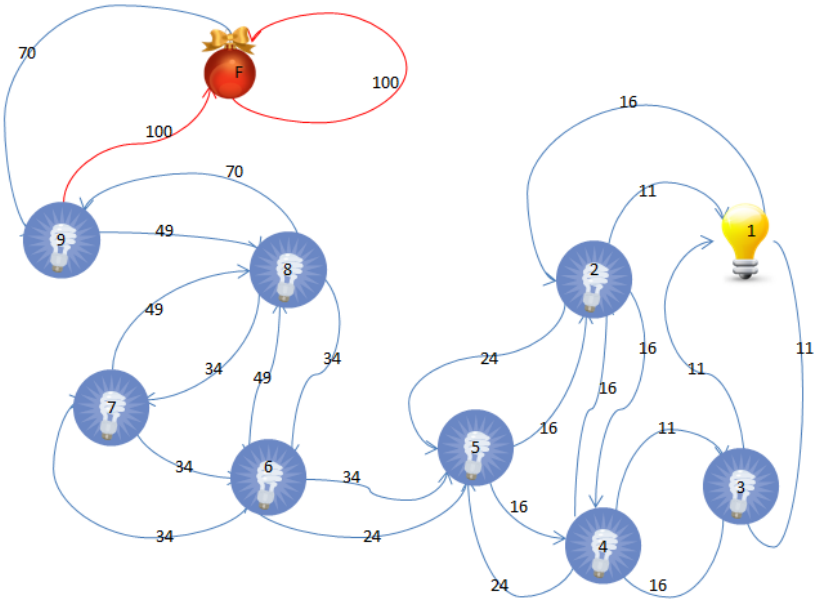


Figure 6. The figure represents the state diagram with values of α= 0.1, γ= 0.75

determination of value for the learning parameter is critical part of an application. The value of γ for an agent reaching at a tree from nine different trees was chosen 0.75. Firstly, assume that the value of gamma is chosen to be 0.2. The agent will try to move further from one block to another, but the probability of finding the agent in the second block would be zero. The agent would take more time to reach the destination point as the value of the learning parameter is low. Now select the value to be 0.8. It was found that an agent is reaching the next block with speed but their probability to reach the destination point is not up to our expectation. Finally, the value of the learning parameter is set to be 0.75 depending upon the efficiency, time taken between each states and nature of the path traversed. Similarly, the learning rate α determines the randomness of the decisions. The agent is at a state and chooses upon action to go to another state. So, to set the Q-value of the state action pair, it looks at the next action, checks the maximum reward, performs the calculations with the Q-learning algorithm, and then gets a value to put in the Q-matrix (Dar, & Mansour, 2003). If we assume alpha to be used at a higher value (i.e. 0.7, 0.8, and 0.9), then it will choose the action that gives the highest future rewards. However, if alpha is low (i.e. 0.1, 0.2, 0.3), then it will more often discovers new paths by choosing the random actions, thereby apparently it is not caring for future rewards. It will be a good practice to choose lower values of alpha initially. By increasing the value, the agent learns about the environment first, and then starts following the optimal path. Note that (i) If alpha = 0.9, 90% it will do optimal, 10% random actions; (i) If alpha = 0.4, 40% it will do optimal, 60% random actions; (iii) If alpha = 0.2, 20% it will do optimal, 80% random actions. Finally, although the pseudo code forms a base for the realization of actual algorithms, it still need to be defined as a series of steps. Although it is easy to understand, but in some occasions, difficult to implement in the real life problems due to the reason that there is no reference how to choose alpha and gamma [20, 21]. This is the reason this research was carried out.

Lower gamma means faster learning and lower reinforcement. Also lower gamma means more exploration. This implies, the agent has lower affinity towards high reward path. Conversely, higher gamma implies slower learning, higher reinforcement. As a result, higher gamma expects less exploration. Consequently, the agent has a higher affinity towards high reward path. A Q-learning approach looks

$$Q = \begin{bmatrix} 0 & 59 & 53 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 53 & 0 & 0 & 59 & 65 & 0 & 0 & 0 & 0 & 0 \\ 53 & 0 & 0 & 59 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 59 & 53 & 0 & 65 & 0 & 0 & 0 & 0 & 0 \\ 0 & 59 & 0 & 59 & 0 & 73 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 65 & 0 & 73 & 81 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 73 & 0 & 81 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 73 & 73 & 0 & 81 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 81 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 100 \end{bmatrix}$$

(a) gamma = 0.9 and alpha = 0.1

$$Q = \begin{bmatrix} 0 & 7 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 7 & 13 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 4 & 0 & 13 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 7 & 0 & 21 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 13 & 0 & 21 & 35 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 21 & 0 & 35 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 21 & 21 & 0 & 60 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 36 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 60 & 100 \end{bmatrix}$$

(b) gamma = 0.8 and alpha = 0.1

$$Q = \begin{bmatrix} 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .6 & 2.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 2 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0.6 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 6 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 12 & 12 & 0 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 40 & 100 \end{bmatrix}$$

(c) gamma = 0.6 and alpha = 0.1

$$Q = \begin{bmatrix} 0 & 32 & 26 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 26 & 0 & 0 & 32 & 41 & 0 & 0 & 0 & 0 & 0 \\ 32 & 0 & 0 & 26 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 26 & 32 & 0 & 41 & 0 & 0 & 0 & 0 & 0 \\ 0 & 26 & 0 & 26 & 0 & 51 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 41 & 0 & 51 & 64 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 51 & 0 & 64 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 51 & 51 & 0 & 80 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 80 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 100 \end{bmatrix}$$

(d) gamma = 0.5 and alpha = 0.1

$$Q = \begin{bmatrix} 0 & 3 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 3 & 6 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 2 & 0 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 3 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 12 & 25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 12 & 0 & 25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 12 & 12 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50 & 100 \end{bmatrix}$$

(e) gamma = 0.4 and alpha = 0.1

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 100 \end{bmatrix}$$

(f) gamma = 0.1 and alpha = 0.1

Figure 7. (a) gamma = 0.9 and alpha = 0.1; (b) gamma = 0.8 and alpha = 0.1; (c) gamma = 0.6 and alpha = 0.1; (d) gamma = 0.5 and alpha = 0.1; (e) gamma = 0.4 and alpha = 0.1; (f) gamma = 0.1 and alpha = 0.1

promising. Using Q-learning, prototype mobile robot software was designed to navigate through an agricultural field in a simulation setting. It is possible to design a program utilizing q-learning through programming in C/C++ language. While the front end was designed in LabView, the backend calculations were done through a C compiler. As mentioned, different gamma values can lead to different types of learning.

5. CONCLUSION

In this research a simulation study on implementation of Q-learning algorithm was conducted. The Q-learning parameters play an important role in determining the learning behavior of an agent [22]. The focus of this research was to select the appropriate values of the learning parameters an agent, as they differ from one situation to another. The parameter learning rate determines the randomness of the decisions and performs more random actions if the value is lower. The learning parameter gamma determines the importance of the future rewards. The value of the learning rate alpha was 0.1 while that of gamma is realized to be 0.75. Q-learning algorithm is simulated by taking a real world example of nine orange trees with defined path and an agent is instructed to reach out the trees following that path. Although, the values of the learning parameters were chosen as 0.1 and 0.75, respectively, however, the values are not optimal and chosen randomly during the operation. As a result, this research work presents different combinations of alpha and gamma in developing Q-learning algorithms. That is the effectiveness of each combination was compared. The future work will include optimization of gamma and alpha and developing software to solve the Q-matrices.

ACKNOWLEDGMENTS

The authors acknowledge the Deans of the Jordan College of Agricultural Sciences and Technology for providing the Release Time and Publication Fee for writing this paper.

References

[1] M. D. Awheda and H. M. Schwartz, "Exponential moving average Q-learning algorithm," in *Adaptive Dynamic Programming And Reinforcement Learning(ADPRL), 2013 IEEE Symposium on*, pp. 31–38, IEEE, 2013.

[2] B. Givan and R. Parr, *An introduction to Markov decision processes*. http://www.cs.rice.edu/~vardi/dag01/givan1.pdf.

[3] M. M. Triola, *Bayes' Theorem*. http://www.faculty.washington.edu/tamre/BayesTheorem.pdf.

[4] B. E. Cline, "Tuning Q-learning parameters with a genetic algorithm," in *Proceedings of the Journal of Machine Learning Research*, vol. 5, 2004.

[5] H. Xu, "Robust decision making and its applications in machine learning," in *Proceedings of the Thesis*, pp. 2–5, McGill University, 2009.

[6] T. Ishiguro, T. Matsui, N. Inuzuka, and K. Wada, "Reinforcement learning methods to handle actions with differing costs in mdps," in *Knowledge-Based Intelligent Information and Engineering Systems*, pp. 553–560, Springer, 2003.

[7] J. Martyna, "Q-Learning algorithm used by secondary users for QoS support in cognitive radio network," in *Modern Advances in Applied Intelligence*, pp. 389–398, Springer, 2014.

[8] P. Dayan, *Reinforcement Learning*. http://www.gatsby.ucl.ac.uk/~dayan/papers/dw01.pdf.

[9] M. Ahmed, "Optimum short part finder for robot using Q-learning," *Diyala Journal of Engineering Sciences*, vol. 05, no. 01, pp. 13–24, 2012.

[10] A. Ng, *Applications of Reinforcement Learning*. http://academicearth.org/lectures/applications-of-reinforcement-learning.

[11] D. Pandey and P. Pandey, "Approximate Q-learning," in *Proceedings of the Second International Conference on Machine Learning and Computing*, pp. 317–320, IEEE, 2010.

[12] K. Teknomo, *Q-learning by examples*. 2006. http://people.revoledu.com/kardi/tutorial/ReinforcementLearning/index.html.

[13] E. Even-Dar and Y. Mansour, "Learning rates for Q-learning," in *Proceedings of the Journal of Machine Learning Research*, vol. 5, pp. 1–25, 2003.

[14] A. R. Cassandra, *Partially observable Markov decision processes*. 2009. http://www.cassandra.org/pomdp/index.shtml.

[15] A. Barto and S. Sutton, "Reinforcement learning: an introduction,"

[16] J. D. R. Millán, D. Posenato, and E. Dedieu, "Continuous action Q-learning," *Machine Learning*, vol. 49, no. 2-3, pp. 247–265, 2002.

[17] J.-F. Chamberland and V. V. Veeravalli, *Wireless Sensor Networks: Signal Processing and Communications Perspectives*. Wiley, 2008.

[18] A. Chapman, *Problem solving and decision making*. 2010. http://www.businessballs.com/problemsolving.htm.

[19] Y. Chen and Q. Zhao, "Wireless Sensor Networks: Signal Processing and Communications Perspectives," 2008.

[20] D. W. Engels and S. E. Sarma, "A hierarchical Q-learning algorithm to solve the reader collision problem," in *Proceedings of the International Symposium on Applications and Internet Workshops*, pp. 1–4, IEEE, 2005.

[21] S. Karumanchi, T. Allen, and S. Scheding, "Non-parametric learning to aid path planning over slopes,"

[22] S. Patnaik and N. Mahalik, "Multiagent coordination utilizing Q-learning," pp. 361–379, 2007.

About This Journal

AIA is an open access journal published by Scientific Online Publishing. This journal focus on the following scopes (but not limited to):

- Artificial Neural Networks
- Bayesian Networks
- Bioinformatics
- Cognitive Science
- Computing and the Mind
- Data Mining
- DNA Computing and Quantum Computing
- Evolutionary Inspired Computing
- Foundations of AI
- Fuzzy Methods
- Intelligent Web
- Machine Learning
- Multiagent Systems
- Natural Computing
- Natural Language Processing
- Neuroinformatics
- Non-classical Computing and Novel Computing Models
- Pervasive Computing and Ambient Intelligence
- Philosophy and AI
- Robotics
- Soft Computing Theory and Applications

Welcome to submit your original manuscripts to us. For more information, please visit our website: http://www.scipublish.com/journals/AIA/

You can click the bellows to follow us:

- Facebook: https://www.facebook.com/scipublish
- Twitter: https://twitter.com/scionlinepub
- LinkedIn: https://www.linkedin.com/company/scientific-online-publishing-usa
- Google+: https://google.com/+ScipublishSOP

SOP welcomes authors to contribute their research outcomes under the following rules:

- Although glad to publish all original and new research achievements, SOP can't bear any misbehavior: plagiarism, forgery or manipulation of experimental data.
- As an international publisher, SOP highly values different cultures and adopts cautious attitude towards religion, politics, race, war and ethics.
- SOP helps to propagate scientific results but shares no responsibility of any legal risks or harmful effects caused by article along with the authors.
- SOP maintains the strictest peer review, but holds a neutral attitude for all the published articles.
- SOP is an open platform, waiting for senior experts serving on the editorial boards to advance the progress of research together.