OPSEARCH (Jul–Sep 2012) 49(3):223–236
DOI 10.1007/s12597-012-0077-2

APPLICATION ARTICLE

Reinforcement learning: exploration–exploitation dilemma in multi-agent foraging task

Mohan Yogeswaran · S. G. Ponnambalam

Accepted: 17 March 2012 / Published online: 10 April 2012

Abstract The exploration–exploitation dilemma has been an unresolved issue within the framework of multi-agent reinforcement learning. The agents have to explore in order to improve the state which potentially yields higher rewards in the future or exploit the state that yields the highest reward based on the existing knowledge. Pure exploration degrades the agent's learning but increases the flexibility of the agent to adapt in a dynamic environment. On the other hand pure exploitation drives the agent's learning process to locally optimal solutions. Various learning policies have been studied to address this issue. This paper presents critical experimental results on a number of learning policies reported in the open literatures. Learning policies namely greedy, ξ-greedy, Boltzmann Distribution (BD), Simulated Annealing (SA), Probability Matching (PM) and Optimistic Initial Values (OIV) are implemented to study on their performances on a multi-agent foraging-task modelled. Based on the numerical results that were obtained, the performances of the learning policies are discussed.

Keywords Reinforcement learning · Q-learning · Learning policies · Multi-agent foraging · Exploration–exploitation dilemma

1 Introduction

Reinforcement learning (RL) has been extensively used in many applications such as industrial control, time sequence prediction, robot soccer competition

M. Yogeswaran · S. G. Ponnambalam (✉)
School of Engineering, Monash University, Sunway Campus,
46150 Petaling Jaya, Selangor, Malaysia
e-mail: sgponnambalam@monash.edu

M. Yogeswaran
e-mail: yogeswaran.mohan@gmail.com

and more. One of the challenges that arise in RL is the exploration and exploitation dilemma [7, 10]. When a RL agent is in a state of an environment, the agent has to decide whether to explore the unknown state and try new actions in search for better ones to be adopted in the future or exploit already tested actions and adopt them. Pure exploration degrades the agent's learning process but increases the flexibility of the agent to adapt in a dynamic environment. On the other hand pure exploitation drives the agent's learning process to locally optimal solutions. Various learning policies have been reported in the open literatures to address this dilemma in RL.

Foraging is an act to go around searching for food or other supplies, and it can be considered the combination problem of parallel searching and transportation. Foraging was chosen because it is a nontrivial and biologically inspired task. Multi-agent foraging, as studied here, involves several agents collecting randomly distributed objects (pucks) and transporting them back to a single location (home region) within a planar arena. This idealization of collection and transport tasks has several applications including hazardous waste cleanup, urban search and rescue, surveillance systems, planet exploration and more. Multi-agent foraging is the problem domain most widely used to study group size scalability and hence the effects of inter-agent interference.

The main contribution of this paper is to investigate the effectiveness of the learning policies used in the open literature to tackle the exploration and exploitation dilemma in multi-agent RL. Because of the large search space, finding a good trade-off between exploration and exploitation is necessary to save computational time and accelerate the learning process of the agents. Learning policies selects an action, as a function of states and rewards experienced so far. Several widely used learning policies were considered namely random search, greedy, ξ-greedy, Boltzmann Distribution (BD), Simulated Annealing (SA), Probability Matching (PM) and Optimistic Initial Values (OIV). The performances of the learning policies evaluated based on the reward sum gathered within the simulation period.

This paper is structured as follows: in the second section, a brief information on RL and Q-learning are discussed. In the third section, the learning policies reported in the open literature are presented. The forth section explains the foraging-task modelling in detail. In the fifth section, the experimental procedures are discussed. In the sixth section, we will present motivational results to access the performances of the reported learning policies with our proposed multi-agent foraging-task and finally end this paper with conclusion.

2 Reinforcement learning

In RL, an agent interacts with an unknown environment, and attempts to choose actions that maximize its cumulative payoff [11]. The RL agents have to learn how to act directly from experience gathered through trial and error leading to the fundamental trade-off between exploitation and exploration dilemma: that is, should the agent exploit its cumulative experience so far, by executing the action that currently seems best, or should it execute a different action, with the hope of gaining information or experience that could lead to higher future rewards [5].

The domain the RL agents makes its decisions in is usually modelled as a Markov Decision Process (MDP) which is a 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$. $\mathcal{S}$ is a set of states which are assumed to be a finite set unless stated otherwise. Each state must satisfy the Markov property: Given the state the agent is currently in, the agentŠs history of past states and decisions is independent of the future states the agent finds itself in [17]. $\mathcal{A}$ is a finite set of actions available to the RL agents at every state, $s \in \mathcal{S}$. $\mathcal{P}\left(s' \mid s, a\right)$ is a function that defines the probability of transitioning to state, s' after action, $a \in \mathcal{A}$ is taken when the RL agents is in state, $s \in \mathcal{S}$. The function $\mathcal{R}(s, a)$ is the reward, $r \in \mathcal{R}$ given to the RL agents based on the action, $a \in \mathcal{A}$ taken with regard to state, $s \in \mathcal{S}$.

The RL agents modelled in for this foraging-task are independent learners (IL). Meaning the approach is to let each agent learn its strategy independently from the other agents and regard the other agents as part of the environment. This method prevents single point of failure for the system. This is an important characteristic since many of the applications rely on continued progress even if some agents in the system fail. Furthermore as the environment dimension becomes larger, the amount of agents will increase, hence lead to a higher computational time for each agent to communicate to each other and take actions based on their coordinate's information.

One of the widely used RL algorithm that we use in this paper is $\mathcal{Q}$-learning [15]. Example of implementation of $\mathcal{Q}$-learning can also be observed in [4] for mobile robots in dynamic environment. $\mathcal{Q}$-learning is an off-policy algorithm that estimates the optimal $\mathcal{Q}$-value function as follows:

$$\mathcal{Q}_{t+1}(s_t, a_t) = (1-\alpha)\,\mathcal{Q}_t(s_t, a_t) + \alpha\left[r_t + \gamma \max_b \left(\mathcal{Q}_t(s_{t+1}, b)\right)\right] \tag{1}$$

where $\mathcal{Q}_t$ is the $\mathcal{Q}$-value estimate at the tth time-step. s_t, a_t, r_t are the state, action, reward at time-step, t respectively. b represents various actions available at state, s_{t+1}. $0 \leq \alpha \leq 1$ is the learning rate and $0 \leq \gamma \leq 1$ is the discount rate. The learning rate weighs the influence of the received rewards in the learning process. A rate of 0 will make the agent not learn anything, while a rate of 1 would make the agent consider only the most recent reward. The discount factor weighs the influence of the future rewards. A rate of 0 will make the agent opportunistic by only considering current rewards, while a rate approaching 1 will make it venture for a long-term high reward.

3 Learning policies

A learning policy selects an action a at time-step t as a function of the history of states, actions, and rewards experienced so far. In this paper, we consider several learning policies that make decisions based on a summary of history consisting of the current time-step t, the current state s, the current estimate $\mathcal{Q}$ of the optimal $\mathcal{Q}$-value function. The learning policy alone is sufficient to determine behaviour [11] in RL agents. Its purpose is to complement the trade-off between exploitation and exploration such that the agent can reinforce the evaluation of the actions it already knows to be good but also explore new actions. In this section, the learning policies used are presented.

3.1 Greedy

Greedy approach is the commonly used learning policy that is associated with the standard $\mathcal{Q}$-learning. Implementation of the greedy learning policy can be seen in [9], where the authors described a formal and principled approach to imitation called implicit imitation between two agents.

$$a = \operatorname{argmax}_a \mathcal{Q}_{(s,a)} \tag{2}$$

Following the greedy learning policy, the agent selects action $a \in \mathcal{A}$ based on the highest $\mathcal{Q}(s, a)$ estimate of the available actions. The action a considered by the greedy learning policy may depend on actions made so far but not on future actions.

3.2 ξ-Greedy

The most popular learning policy is ξ-greedy [12]. Whiteson et al. [16] have applied this learning policy in two different tasks specifically mountain car task and server job scheduling and compared its performance with other exploration strategies. In ξ-greedy learning policy a single parameter ξ controls what fraction of the time the agent deviates from greedy behaviour. Each time the agent selects an action, it chooses probabilistically between exploration and exploitation. With probability ξ, it explores by selecting randomly from the available actions. With probability $1 - \xi$, it exploits by selecting the greedy action where ξ is a small positive value, $0 < \xi < 1$.

$$a = \begin{cases} rand\,(a_n) & rand\,(0, 1) \leq \xi \\ \operatorname{argmax}_a \mathcal{Q}(s, a) & otherwise \end{cases} \tag{3}$$

High values of ξ will force the agent to explore more frequently and as a result will prevent the agent from concentrating its choices to the optimal action, while giving the agent the ability to react rapidly to changes that takes place in the environment. Low value of ξ will drive the agent to exploit more optimal actions. For the studied case, $\xi = 0.2$ improved the quality of the solution marginally out of the set of tested $\xi = \{0.1, 0.2, 0.3, 0.4, 0.5\}$ values.

3.3 Boltzmann Distribution (BD)

BD is a learning policy that reduces the tendency for exploration with time. It is based on the assumption that the current model improves as learning progresses. BD assigns a positive probability to any possible action according to its expected utility and according to a parameter T called temperature [1].

Morihiro et al. [8] implemented BD for a new multi-agents flocking framework using $\mathcal{Q}$-learning. BD assigns a positive probability for any possible action $a \in \mathcal{A}$ using Eq. 4.

$$P(a|s) = \frac{e^{(\mathcal{Q}(s,a)/T)}}{\sum_b e^{(\mathcal{Q}(s,b)/T)}} \tag{4}$$

$$T_{\text{new}} = e^{(-dj)} T_{\max} + 1 \tag{5}$$

Actions with high $\mathcal{Q}(s,a)$ are associated with higher probability P. T decreases as iteration j increases over time. Therefore, as learning progresses, the exploration tendency of the agents reduces and BD learning policy will tend to exploit actions with high $\mathcal{Q}(s,a)$. For the foraging-task in this paper, temperature $T_{\max}$ set to 500, decay rate d is set 0.009.

3.4 Simulated Annealing (SA)

The SA learning policy was introduced by Guo et al. [3] where the paper reports experimental results on a 22×17 puzzle problem and comparing the capabilities of the different exploration strategies.

$$a = \begin{cases} rand\,(a_n) & \xi < e^{\left(\mathcal{Q}(s,rand(a_n)) - \text{argmax}_a \mathcal{Q}(s,a)\right)/T} \\ \text{argmax}_a \mathcal{Q}(s,a) & otherwise \end{cases} \tag{6}$$

Following this learning policy, when the agent selects actions, it does not obey the policy learned so far, but also attempts to explore by increasing chances of selecting actions other than those adopted by the current (possibly sub-)optimal policy. The agent selects an arbitrary action $a \in \mathcal{A}$ and executes the action if $\xi <$ the probability defined in Eq. 6 where $\xi \in (0,1)$. Otherwise the agent selects and executes the action with the highest $\mathcal{Q}(s,a)$. T is a positive parameter called the temperature which is set through trial and error. At the beginning of the iteration, when the temperature is high, the SA learning policy allows more exploration. As the temperature drops based on Eq. 5, the SA learning policy reduces the exploration rate and drives the agent to select actions with the highest $\mathcal{Q}$-value. In this paper, T is set to 500.

3.5 Probability Matching (PM)

One of most recent work was by Koulouriotis and Xanthopoulos [6] where the paper examines multi-armed bandit tasks to solve the balancing problem of exploration and exploitation using policies using PM together with several other exploration strategies.

$$P_{\max} = 1 - (K-1)\, P_{\min},\, P_{\min} \in (0,1) \tag{7}$$

$$P_{a_i} = P_{\min} + (1 - K.P_{\min}) \frac{\mathcal{Q}(s,a_i)}{\sum_{n=1}^{K} \mathcal{Q}(s,a_n)} \tag{8}$$

The PM rule computes each action's selection probability P_{a_i} as the proportion of the action's $Q(s, a)$ to the sum of all possible action's $Q(s, a)$ estimates. To enforce a sufficient amount of exploration throughout the simulation period, the exploration rate for the available states are kept above a threshold value of $P_{\min}$. That way, actions leading to bad states will also be considered throughout the entire simulation period with lower probability estimation. The exploration rate for each action is also kept below $P_{\max}$. $P_{\max}$ is evaluated using Eq. 7. K is the number of available actions from the currently observed state. $P_{\min}$ is set to 0.1 in this experiment. $P_{\min}$ is adjusted to a proper value depending on the amount of the exploration is required for the environment.

3.6 Adaptive Pursuit strategy (AP)

Pursuit methods is a learning policy for the multi-armed bandit problem that were initially proposed by Thathachar and Sastry [13]. Adaptive pursuit policy was introduced by Thieren [14]. Adaptive pursuit is also similar to PM as the learning policy does not allow probabilities to be under the threshold of $P_{\min}$ or exceed the maximum value of $P_{\max} = 1 - (n - 1) P_{\min}$. The probability of all available actions are evaluated initially using the respective action's $Q(s, a)$ value. Then using the adaptive pursuit learning policy [6], the probability of the optimal action $P(a_g)$ (the action with the highest selection probability) is incremented towards $P_{\max}$ using Eq. 9 while all the remaining action's probabilities are decremented towards $P_{\min}$ using Eq. 10. The AP learning policy tries to accelerate the learning process by amplifying the selection probability for the optimal action from the beginning of the simulation. The α is a small positive parameter, set through trial-and-error using a set of values. $0 < \alpha < 1$ is set to 0.2 in this experiment.

$$P'\left(a_g\right) = P\left(a_g\right) + \alpha\left(P_{\max} - P\left(a_g\right)\right) \tag{9}$$

$$P'\left(a_i\right) = P\left(a_i\right) + \alpha\left(P_{\min} - P\left(a_i\right)\right), \forall i \neq g \tag{10}$$

3.7 Optimistic Initial Values (OIV)

Using OIV learning policy, the initial $Q(s, a)$ value of each state action pair can be set to some overwhelmingly high number. If a state s is visited often, then its estimated value will become more exact, and therefore, lower. Thus, the agent will try to reach the more rarely visited areas, where the estimated state values are still high. A work by Even-Dar and Mansour [2] gave theoretical justification for the method. In their paper, they proved that if the optimistic initial values are sufficiently high, Q-learning converges to a near-optimal solution [12]. In our experiment, the initial $Q(s, a)$ values are set to +10.0 to drive the agents to explore non-visited states.

4 Environment model

In order to validate the objectives of the experiment, a multi-agent foraging-task designed for comparing the learning policies using traditional RL approaches. This section describes the agents, experimental environment and rewards assignment to the modelled foraging-task.

4.1 RL agents

The agents are modelled strictly based on Khepera II mobile robot (Fig. 1). Each agent is simulated with a ring of 8 proximity sensors. The maximum sensing range of the proximity sensors are set to 50 mm and the minimum range to 20 mm. When an obstacle enters the minimum sensing range of the agent, the agent is considered to have collided with an obstacle in the environment.

Each agent is designed with a gripper module to manipulate pucks that are scattered in the environment. Based on the input from the proximity sensor, the agents will either grip the puck from the environment or drop the puck at the home location.

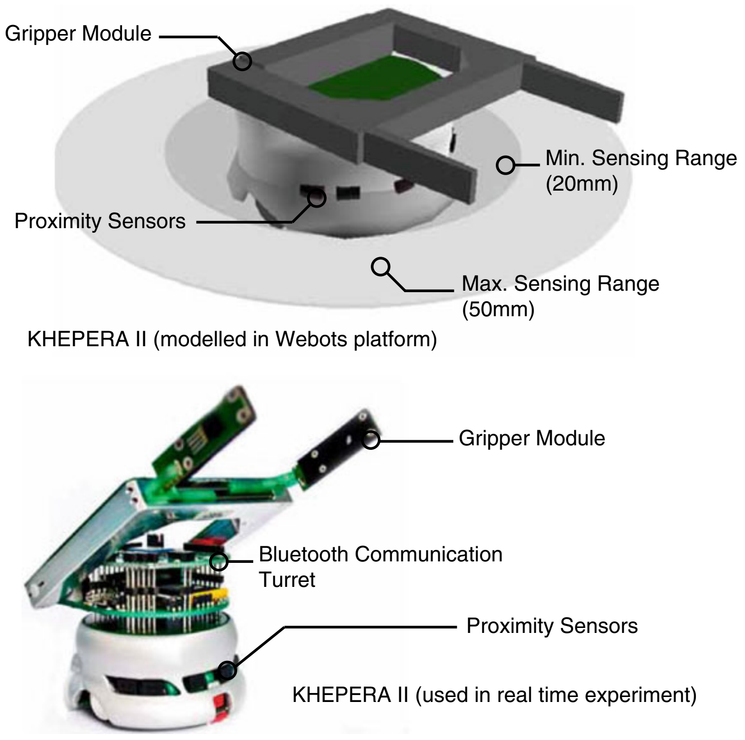
Gripper Module
Min. Sensing Range (20mm)
Proximity Sensors
Max. Sensing Range (50mm)
KHEPERA II (modelled in Webots platform)
Gripper Module
Bluetooth Communication Turret
Proximity Sensors
KHEPERA II (used in real time experiment)

Fig. 1 The RL agent (Khepera II model). The agent in the simulation is modelled as similar as possible to the real physical mobile-robot. The only difference will be in the Bluetooth communication turret

The physical Khepera II agent has an additional Bluetooth communication turret to enable real time communication with the CPU. The respective Q-values for each agent are stored in Q-table in the CPU where the Khepera II will read or update for each iteration. For this experiment, 3 Khepera II agents were used to collect and deposit 10 pucks that are distributed into a predetermined location in the designed environment.

4.2 Environment

The width and length of the environment each are respectively 2000 mm which is separated into grid of 100 by 100 mm for the agent to explore and exploit (Fig. 2). Walls are located in the environment to act as obstacle in the environment and also as a boundary for the environment. The home location is located exactly in the middle of the environment. The agent will deposit the collected pucks at the home location. Pucks and agents are placed in the environment at predetermined locations at the beginning of the simulation.

Set of states observed by the agents are of $s = \langle x_i, z_i, p \rangle$ where x_i is the coordinate location in the x-axis, z_i is the coordinate location in the z-axis and

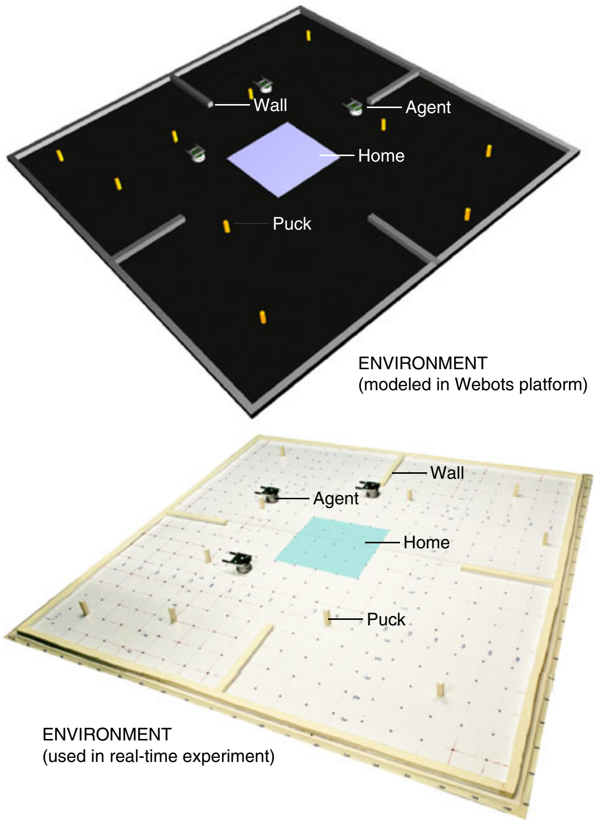

Fig. 2 The model of the environment. The physical environment is exactly the same in terms of dimension and the locations of the agents and pucks in relation to the simulation

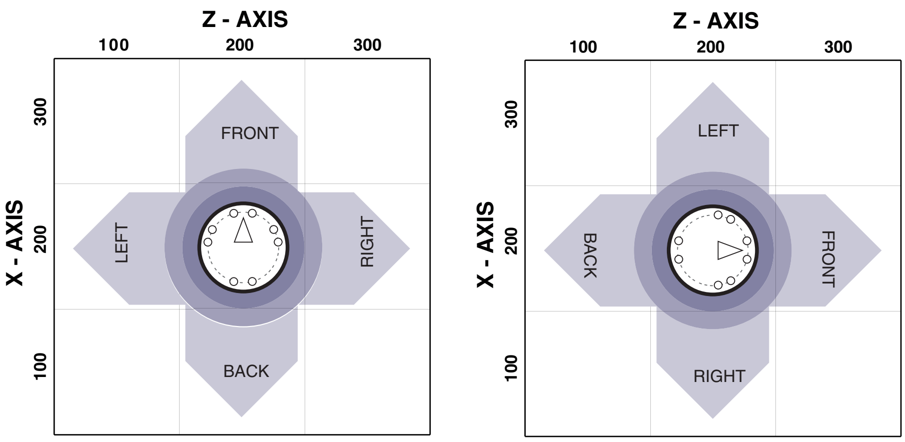

Fig. 3 Available states and actions. The actions are relative to the states observed

p is the variable representing the presence of puck in the gripper module of the agent. p is set to 0 if there is no puck in the gripper module and set to 1 otherwise. The agents have a set of actions $\mathcal{A} = \langle front; back; left; right \rangle$ to be taken in observed states, $\mathcal{S}$ (Fig. 3). Based on the adopted learning policy the agents will choose the desirable action $a \in \mathcal{A}$ at the current available states $s \in \mathcal{S}$. Including the puck availability p, there are 882 states overall for the agents to explore and exploit in the foraging-task modelled in this paper. $\mathcal{Q}$-values that are updated or reinforced are stored into a table called $\mathcal{Q}$-table (Table 1).

4.3 Rewards

Rewards show the desirability of the action taken by the agent towards the perceived states. If the action taken is attractive, a positive reward is given to the agent. If the action taken is not attractive then a negative reward is given to the agent. A +10.0 reward is given to the agents for picking up a puck from the environment. This triggers the agents to visit the same location again to look for more rewards. A reward of +10.0 is also given to the agents for dropping

Table 1 $\mathcal{Q}$-Table

i	States			$\mathcal{Q}(s, a)$
	x_i	y_i	p	
1	−1000	−1000	0	0
2	−1000	−1000	1	0
3	−1000	−900	0	0
4	−1000	−900	1	0
...	...	...	0	0
...	...	...	1	0
881	1000	1000	0	0
882	1000	1000	1	0

the puck at the home location. The agents are given a reward of −0.1 if the agents wander in the home region without a puck in the gripper module. This will lead the agents to escape the home location and perform search for the pucks in the environment. The agents are given a reward of −0.1 if the agents wander in the environment (not in the home region) with a puck in the gripper module. This is to drive the agents to approach the home region. A reward of −1.0 is also given if the agents experience any collision with the obstacles in the environment.

5 Experimental procedures

The environment or the search space contains 10 pucks, 3 agents and a depositing plane referred to as home. The simulation starts by setting the episode counter, timer and $\mathcal{Q}(s,a)$ values in the $\mathcal{Q}$-table are set to 0 (except for the OIV learning policy is set to +10.0). In the initialization stage, the timer count is set to 0 followed by the resetting of agent's and the puck's locations to their respective coordinate in the environment. At the execution of the simulation, the agents will start wandering in the environment searching for pucks. The agent will observe the available states, $\mathcal{S}$ in the environment from the current state s. The agent will then select an action $a \in \mathcal{A}$ based on the adopted learning policy. Reward $r \in \mathcal{R}$ for the action $a \in \mathcal{A}$ taken is calculated. The relevant $\mathcal{Q}(s,a)$ value in the $\mathcal{Q}$-table is updated using Eq. 1. Then the current state s is set to s'. The episode counter is incremented by 1. If an agent encounters with a puck in the environment, the agent will grab the puck and a reward of +10.0 will be given to the agent. This reward will be updated in the $\mathcal{Q}$-table using Eq. 1.

After the update, the agent will search for the home location to deposit the puck. The agent will observe the available states, $\mathcal{S}$ in the environment from the current state s. The agent will then select an action $a \in \mathcal{A}$ based on the adopted learning policy. Reward $r \in \mathcal{R}$ for the action $a \in \mathcal{A}$ taken is calculated. The relevant $\mathcal{Q}(s,a)$ value in the $\mathcal{Q}$-table is updated using Eq. 1. Then the current state s is set to s'. The episode counter is incremented by 1. If an agent is in the home region, the agent will deposit the puck and a reward of +10.0 will be given to the agent. This reward will be updated in the $\mathcal{Q}$-table using Eq. 1.

If collision occurs while the agent is performing wandering or homing, a reward of −1.0 will be given to the agent. The relevant $\mathcal{Q}(s,a)$ value in the $\mathcal{Q}$-table is updated using Eq. 1. The agent will return back to its original state which is the current state s. The episode counter is incremented by 1.

If 10 hours of the simulation time is complete or the numbers of pucks collected are 10, the simulation will be reset again which means a trial is over. At the beginning of each trial, the timer will be set to 0, pucks and agents will be placed again in the environment at the predetermined location. Else the agents will continue to map the environment until the termination conditions are met. The simulation runs for 30 trials for each learning policy and the simulation commences until 10 sets of 30 trials are completed. At the beginning of each set, the initialization stage is carried out again. Finally the average data acquired for 30 trials are presented and discussed.

6 Results and discussion

The results of the simulation experiments conducted are presented in this section. Each policy is tested for 10 sets of 30000 episodes and the average results are presented.

It is observed in Fig. 4 that OIV learning policy performed better compared to greedy, ξ-greedy, SA, BD, PM and AP learning policy. The greedy learning policy prevents exploitation of undesired states as the learning process converges. This characteristic helps to avoid collision with obstacles in the environment, hence reducing -ve rewards throughout the learning process. However the performance of the greedy strategy is poor due to the lack of exploration in the environment. This drives the agents to the local optima and far from achieving the overall objective.

Although ξ-greedy strategy promises convergence, it displays a number weaknesses. The agent under this learning policy have no control over when or how to explore. Therefore, the agent may choose to take a random action when an obvious action might lead to a better reward and chose an optimal action without knowing whether the taken action is appropriate or not.

The $Q(s, a)$ values are similar when the search space is large for example the foraging-task modelled in this paper. Therefore, the action leading to a state that is one step closer to the goal only has a slightly higher $Q(s, a)$ value than the action leading to a state that is one step further away from the goal. The probability of selecting the two actions will also be close to each other,

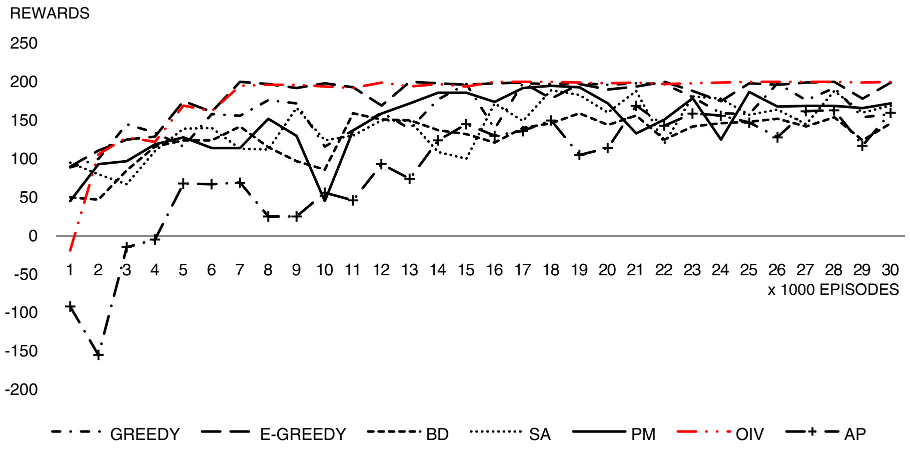

Fig. 4 Rewards gathered for the foraging-task

Table 2 Performance summary of learning policies

Exploration strategy	Average collision	Average puck	Average time (ms)
Random	43.067	5.200	36000
Greedy	**5.013**	8.367	28565.4
ξ-greedy	7.901	9.433	25550.4
BD	32.333	8.810	29947.1
SA	17.437	8.577	30073.8
PM	16.619	8.928	25832.1
OIV	12.211	**9.567**	**22533.6**

Bold entries indicate the best performance for the respective objectives

although one of the actions is clearly better than the other. Therefore, the BD and SA learning policy performed poorly. Furthermore, the SA and BD learning policy requires more time to converge which makes it not suitable for tasks with limited learning time.

The PM and AP learning policy assigns relative probabilities to all actions based on the $Q(s, a)$ values. The actions with lower $Q(s, a)$ values are still chosen even when the agents have found the optimal solution. This leads to the poor performance of the learning policy. The OIV learning policy is a simple method of biasing the initial $Q(s, a)$ values to a very high value. Therefore, unexplored actions have greater $Q(s, a)$ value estimates than explored ones, so unexplored actions tend to be selected. When all actions have been explored a sufficient number of times, the true $Q(s, a)$ value function overrides the initial $Q(s, a)$ value function estimates. This ensures the exploration is done to all the available states before making a valid judgment or decision in the action selection.

The data in Table 2 shows the individual contribution of the learning policies in the foraging-task modelled. From Table 2 shows that the greedy learning policy performed well for collision avoidance in the environment. The OIV learning policy outperformed other learning policies in puck retrieval and

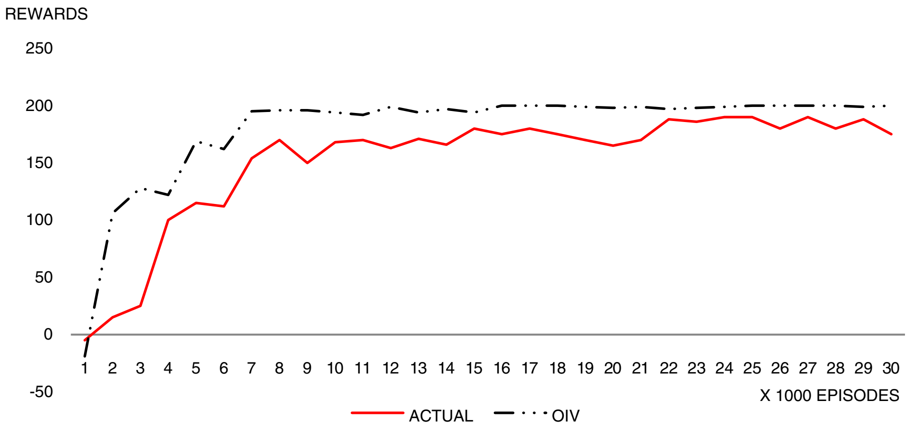

Fig. 5 Comparison between physical agents and simulation result

shortest time in collecting all the 10 pucks distributed in the environment. Based on the observation, we conclude that the OIV learning policy is a better learning policy overall for the foraging-task modelled in this paper.

Figure 5 shows the performance of the physical agents comparing with simulation results using the OIV learning policy. The physical agents performed well and achieved about 83% of the simulation results. The discrepency is due to the delay created by the Bluetooth communication. The physical agents have to send request to the main CPU and wait for the response from the CPU via Bluetooth communication turret on the agent. Therefore, less number of actions is taken within the time constraint compared to the agents in the simulation as there are no delays involved in executing the actions. The noise introduced by the sensors and effectors also leads to wrong estimation of the state hence leading the physical agents to generate unpredictable behaviours.

7 Conclusion

This work has focused on effects of various learning policies in multi-agent foraging domain. The behavior of a Q-learning agents using the reported policies namely greedy, ξ-greedy, BD, SA, PM, AP and OIV have been studied in a foraging-task and the results are reported. Then the best learning policy identified in the simulation was tested using the physical agents. We demonstrated the advantages and disadvantages of the policies by implementing them on a group of 3 Khepera II agents in the foraging-task modelled. Through the experiments conducted, although all the strategies carry their own advantages and disadvantages, it is clear that OIV learning policy is much more practical and effective compared to the other strategies studied for the multi-agent foraging-task studied. It is necessary to identify the suitable policies in Q-learning for different types of tasks as the suitable policies helps to accelerate the convergence of the agent's learning process.

Acknowledgement This research is funded by e-Science grant provided by Ministry of Science, Technology and Innovation (MOSTI), Malaysia, Project Number: 03-02-10-SF0036.

References

1. Carmel, D., Markovitch, S.: Exploration strategies for model-based learning in multi-agent systems: exploration strategies. Auton. Agent Multi-Ag. **2**(2), 141–172 (1999)
2. Even-Dar, E., Mansour, Y.: Convergence of optimistic and incremental Q-learning. Adv. Neural Inf. Process. Syst. **2**, 1499–1506 (2002)
3. Guo, M., Liu, Y., Malec, J.: A new Q-learning algorithm based on the metropolis criterion. IEEE Trans. Syst. Man Cybern. **34**(5), 2141 (2004)
4. Kareem Jaradat, M., Al-Rousan, M., Quadan, L.: Reinforcement based mobile robot navigation in dynamic environment. Robot. Comput.-Integr. Manuf. **27**(1), 135–149 (2011)
5. Kearns, M., Singh, S.: Near-optimal reinforcement learning in polynomial time. Mach. Learn. **49**(2), 209–232 (2002)

6. Koulouriotis, D., Xanthopoulos, A.: Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems. Appl. Math. Comput. **196**(2), 913–922 (2008)
7. Kroemer, O., Peters, J.: Active exploration for robot parameter selection in episodic reinforcement learning. In: 2011 IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), pp. 25–31. IEEE (2011)
8. Morihiro, K., Isokawa, T., Nishimura, H., Matsui, N.: Emergence of flocking behavior based on reinforcement learning. In: Knowledge-Based Intelligent Information and Engineering Systems, pp. 699–706. Springer (2006)
9. Price, B., Boutilier, C.: Accelerating reinforcement learning through implicit imitation. J. Artif. Intell. Res. **19**(1), 569–629 (2003)
10. Shen, Y., Zeng, C.: An Adaptive Approach for the Exploration-Exploitation Dilemma in Non-stationary Environment. In: 2008 International Conference on Computer Science and Software Engineering, vol. 1, pp. 497–500. IEEE (2008)
11. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. The MIT Press (1998)
12. Szita, I., Lőrincz, A.: The many faces of optimism: a unifying approach. In: Proceedings of the 25th International Conference on Machine Learning, pp. 1048–1055. ACM (2008)
13. Thathachar, M., Sastry, P.: A class of rapidly converging algorithms for learning automata. IEEE Trans. Syst. Man Cybern. **15**, 168–175 (1985)
14. Thierens, D.: An adaptive pursuit strategy for allocating operator probabilities. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1539–1546. ACM (2005)
15. Watkins, C., Dayan, P.: Q-learning. Mach. Learn. **8**(3), 279–292 (1992)
16. Whiteson, S., Taylor, M., Stone, P.: Empirical studies in action selection with reinforcement learning. Adapt. Behav. **15**(1), 33 (2007)
17. Wiewiora, E.: Efficient Exploration for Reinforcement Learning (2004). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.124.2602