2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)

Multi Q-Table Q-Learning

Nitchakun Kantasewi
Dept. of Electrical Engineering,
Faculty of Engineering, Kasetsart University,
Bangkok, Thailand
nitchakun.k@ku.ac.th

Sanparith Marukatat
National Electronics and Computer Technology Center,
Pathum Thani, Thailand
Sanparith.Marukatat@nectec.or.th

Somying Thainimit*
Dept. of Electrical Engineering,
Faculty of Engineering, Kasetsart University,
Bangkok, Thailand
fengsyt@ku.ac.th

Okumura Manabu
Dept. of Information and Communications Engineering
Tokyo Institute of Technology,
Yokohama, Japan
oku@lr.pi.titech.ac.jp

Abstract*—Q-learning is a popular reinforcement learning technique for solving shortest path (STP) problem. In a maze with multiple sub-tasks such as collecting treasures and avoiding traps, it has been observed that the Q-learning converges to the optimal path. However, the sum of obtained rewards along the path in average is moderate. This paper proposes Multi-Q-Table Q-learning to address a problem of low average sum of rewards. The proposed method constructs a new Q-table whenever a sub-goal is reached. This modification let an agent to learn that the sub-reward is already collect and it can be obtained only once. Our experimental results show that a modified algorithm can achieve an optimal answer to collect all treasures (positive rewards), avoid pit and reach goal with the shortest path. With a small size of maze, the proposed algorithm uses the larger amount of time to achieved optimal solution compared to the conventional Q-learning.

Keywords*—Reinforcement Learning; Q-Learning; Machine Learning; Maze; Agent; Multi Q-Table, Multiple Q-Table;

I. INTRODUCTION

Reinforcement Learning (RL) is a major area of Machine Learning, aiming to find suitable action model that would maximize outcome of its task. The RL algorithm trains agent to select and execute action under specific environment. The agent, then, receive rewards and punishments as system feedback for positive and negative experiences. Based on these experiences, agent learns which action is the most effective action to take in particular situation in the future encounter. The RL learning is *trial-and-error* approach, aiming to maximizing long-term reward [1].

RL is widely used in many applications such as game playing agent [1], finding the right parameters for sensors in robotic [2], finding the right time to trigger system in order to achieve the maximum potency [3] mimicking human decision making [4] and finding the shortest path (STP), i.e. Travelling Sale Man and maze application. The STP is an algorithm of finding the shortest path from source to destination. In Maze solving applications, the shortest path from staring point to exit point is searched [5]. Finding the STP is one of the best problem to describe RL's ability since RL's agents learn from observed system environment. Thus, RL can handle dynamic changes in its environment better than the static approaches such as A* star and Dijkstra's algorithm.

One of the first RL algorithms is called Q-learning [6]. Q-Learning uses a matrix to realize the transit between states in a maze. Each state refers to current situation of the agent. If there exists direct transit between states they give value 0, otherwise $-\infty$. Every time agent takes an action, value in the matrix will be updated depending on received reward. The learning matrix is trained off-line. This phase is called '*exploration*' in which agent randomly explores the environment. After enough random exploration actions, the '*exploitation'* phase is taken placed. In this phase, the most optimal action can be chosen based on already learned Q-values. The agent is able to determine the optimal action from the entire history of states.

Reference [7] compares the exploration strategies, which are UCB-1, ε-greedy, softmax and pursuit, to determine which strategies give the best potential. Experimental mazes used in [7] has one optimal goal and two suboptimal goals. The task for agents is to find the optimal goal. The results from the experiments indicated that ε-greedy strategy performed the worst among other exploration strategies in form a maze of size 10×10 and 20×20. softmax outperforms all other strategies for 10×10 maze, whereas the UCB-1 performs the best for 20×20 maze. However, softmax is able to find optimal goal state in more than 40% of the cases, whereas the ε-greedy performs worst and can't even find the optimal goal.

Reference [3] improves efficiency of RL by placing sub-reward as an additional reward for agent i.e. placing a reward in the state near an exit state. By doing this, agent tends to choose the STP toward sub-reward. However, the existence of sub-reward tears down the learning process. Agent behaves like a round trip to obtain more sub-rewards. To prevent such problem, the forgetting mechanism that decreases the value of the sub-reward was implemented. The forgetting mechanism is active when instructional signal is received. Sub-reward's value is terminated in the later iteration when the path to end point is vaguely realized. Because the path to end state has already been

978-1-7281-1921-2/19/$31.00 ©2019 IEEE

ambiguously construct, agent will exploit and find the shortest path by itself. Based on experiments of 1,000 simulations. 2,000 trials of goal seeking for each simulation, the proposed method can reduce the total episode length of first 50 trials up to 1,400 episodes compared to the conventional approach. This method improves the learning speed.

This comes an interesting question; what if a maze is not an ordinary maze. Examples are adding treasures for agent to collect before finding an exit, adding traps or pits to give agent a negative reward so that an agent needs to avoid walking through this stage.

Reference [8] proposed solving multiple objective reinforcement learning using the reward shaping. The reward shaping is additional included in the reward function in order to speed up learning phase. The algorithm requires an understanding of environment and actions. Basically reward shaping function depends on objective and environment, for example collecting treasures in maze, reward shaping function might be the distance between agent and treasure. Result from using this method can drastically speed up in the learning phase but cannot improve the last accumulate reward due to RL always give the maximum outcome of its task.

In this paper, we propose an alternative way to solving maze with sub-task. The proposed method is aimed to achieve solving a multiple task without additional information such reward shaping algorithm. By simple adding sub-task into maze, agent tends to iteratively come back to the treasure state based on past experience of getting reward when reaching this state. Since the reward appears only once, the agent then gets penalty every time it loops back to the treasure state. The loop is continued until the summation of the obtained penalties is large enough. The looping behavior is called 'reward loop'. In this paper, we proposed multi Q-table Q-learning to address a reward loop problem.

The paper is organized as follows: section II is a principle of Reinforcement Learning, Q-learning and exploration policy used in this paper. Section III describes the proposed method used to solve a maze with sub-task problem, Section IV is experiment setup and Section V reports results of the proposed method. Finally, the conclusion will be presents in section VI.

II. Reinforcement Learning

Reinforcement Learning is a machine learning technique in which an agent learns by interacting on environment. Each time, the environment is in state s. Based on state s_t at time t, a RL agent chooses and executes an action a_t; resulting in state transition from state s_t to state s_{t+1} and the agent receives an amount of reward from environment. These rewards will determine which action is best suitable for that state. After agent spending time to learn an environment, agent will then know how to maximize the amount of reward it can make. This learning method is called *trial and error*. The foundation of this decision making method is called Decision Process which is mathematically defined by:

- A discrete set of n states $S = \{s_1, s_2, ..., s_n\}$, where $s_t \in S$ describes the state at a time t in environment.
- A discrete set of m actions $A = \{a_1, a_2, ..., a_m\}$, where a_t describes the selected action at time t by the agent.
- A transition function $T(s,a,s')$ describe that selecting action a in the state s cause changing state to state s'.
- A reward function $R(s,a,s')$ denotes a reward obtained from changing state from s to s' by using action a, r_t reward is obtained at time t
- A discount factor $0 \leq \gamma \leq 1$ indicates the importance of future rewards compared to the immediate reward.

A. Q - learning

Q-learning is an algorithm which inherits the fundamental of trial and error learning and makes a decision from the past experience. The Q-learning [6] is an off-policy learning technique. An off-policy learning method means that Q-learning can learn the optimal Q-function and the final policy will differ from exploring policy. When agent learns the optimal Q-function, the selecting policy is to select the action with the highest Q-value in the state. The Q-value denotes a value of performing an action a in state s. The Q-learning tries several available actions and continue to updating the Q-value. On transitioning from state s_t, to s_{t+1}, under action a_t and receive a reward r_t, the Q-value is updated using the following Q-function [9]:

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha[r_t + \gamma \, maxQ_t(s_{t+1},a_t) - Q_t(s_t,a_t)] \quad (1)$$

where $0 \leq \alpha \leq 1$ is learning rate of the update rule.

In case of agent reach the terminated state, update rule of Q-value of t+1 state is as follow:

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha[r_t - Q_t(s_t,a_t)] \quad (2)$$

The Q-learning estimates the Q-value based on both current state and its expected values in the future, which is $r_t + \gamma \, maxQ_t(s_t+1,a_t)$. The Q-function gives us better and better approximations by continuously updating the Q-values which is stored in a table called Q-Table. Once the optimal Q-Table is known, the agent starts to exploit the environment and can select the optimal action with the highest Q-value in state s.

Structure of a Q-table is a matrix of size $n \times m$, where n is number of possible actions agent can take and m is a number of states in an environment. For example, describing a maze of size 3×3 as in Fig. 1. There're 9 states in total. Available actions are assigned as Up, Down, Left, Right or *4* actions in total.

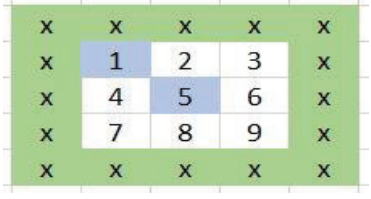

Fig. 1 A maze of size 3 × 3

In Fig. 1, state s_1 and s_5 are obstacle state which agent can't transit into. Assuming agent start from s_2 and goal state is s_4. In state s_2 the only possible action agent can take is to transit to the right state. Transit to right action is labeled as a_4. The Q-value for this situation denotes by $Q(s_2, a_4)$. Fig. 2 illustrates structure of the obtained Q-table of the maze. For the first iteration, the Q-table is initialized with '0'.

$$Q = \begin{array}{|cccc|} \uparrow & \downarrow & \leftarrow & \rightarrow \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Q(s_2,a_4) \\ 0 & Q(s_3,a_2) & Q(s_3,a_3) & 0 \\ 0 & Q(s_4,a_2) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ Q(s_6,a_1) & Q(s_6,a_2) & 0 & 0 \\ Q(s_7,a_1) & 0 & 0 & Q(s_7,a_4) \\ 0 & 0 & Q(s_8,a_3) & Q(s_8,a_4) \\ Q(s_9,a_1) & 0 & Q(s_9,a_3) & 0 \end{array} \quad (1)$$

Fig. 2 The Q-table of the maze of 3 × 3

B. ε-greedy

The important part for reinforcement learning is tradeoff between exploration and exploitation. If we let agent explores more, the better understanding of the whole maze is realized. The greater amount of exploitations, the greater information about the surrounding area. Ideally, with unlimited explorations and exploitations yield perfect information about environment. However, an unlimited number is impossible to achieve. Thus, the exploration and exploitation is a trade-off. To balance between the exploration and exploitation, the ε-greedy [6] is used as an exploration policy.

The ε-greedy is the most used exploration policy. The value of $0 < \varepsilon < 1$ is a tuning parameter of exploration to decide which action to perform in state s_t. The agent chooses an action with the highest Q-value in the current state s_t with probability of $1 - \varepsilon$. If no action satisfies this condition, agent randomly chooses an action from all possible actions pool. The higher the ε-value, the higher chance to random an action. This refers to the higher exploration rate. The higher the exploration rate, the better the understanding of environment. However, taking too many explorations results in lower accumulated reward. To prevent this, each time agent select action, ε-value is reduced by the following equation.

$$\varepsilon = \varepsilon - \frac{1-\varepsilon\_min}{AnnealingStep} \quad (3)$$

where ε_min denotes the minimum probability to do an exploration, *AnnealingStep* is the amount of maximum steps used till ε reaches ε_min.

In this paper, the number of epochs for each run is fixed. So *AnnealingStep* determines how well agent can achieve in the different explore and exploit time.

III. Proposed Method

By investigating the conventional Q-learning under environment with sub-tasks, the Q-learning takes too long time in exploring paths to reach the specified goal. The agent likely decides to walk back into the treasure state or the previous state due to the received positive reward when agent reach sub-task or treasure state. However, the treasure appears only once per run. This causes the agent stuck in the loop, called reward loop. It takes sometimes before penalty from walking into blank state terminates all rewards from this treasure state.

A. Multi Q-Table Q-Leraning(MQQ)

To address the problem of reward loop, Agent need to forget the collected reward. The hierarchy for the proposed method is to not letting value received from sub-reward interfering with decision making method. But agent must maintain the memory of the position of this sub-reward. Multiple Q-table acts as memory after receiving sub-reward. Because Q-learning is an algorithm to find the shortest path to reward state. Agent will choose the path to reach the nearest reward state. After reaching reward state, which is a sub-reward, agent has to go to the next nearest sub-reward. Changing Q-table after reaching sub-reward results in agent starts doing its task again from the beginning. But this time, starting point is the state where it was a reward state. Agent will commit this series of finding shortest path to next sub-reward until collect all sub-reward in the maze then the last reward state is a goal state. This will conclude the whole method of collecting all reward and navigate through the terminated state.

The proposed methods modifies Q-Learning method. Structure of Q-table remain the same as conventional method. The proposed algorithm is as follows:

- Agent uses Q-table Q_0 at the start of every epoch.
- Reaching treasure state s_x for the first time in the run, Q-table Q_x will be created. Agent then changes current Q-table to newly create Q-table Q_x.
- New Q-table has the same structure as Q_0 but initial all values with 0.
- Collecting treasure in state s_x in the future epoch in the same run, resulting in changing current Q-table to Q_x.
- Note that treasure can be collected only once per epoch, thus collecting treasure in state s_x and changing Q-table to Q_x can occur only once per epoch.
- Q-table can be created up to the number of treasures in the maze.

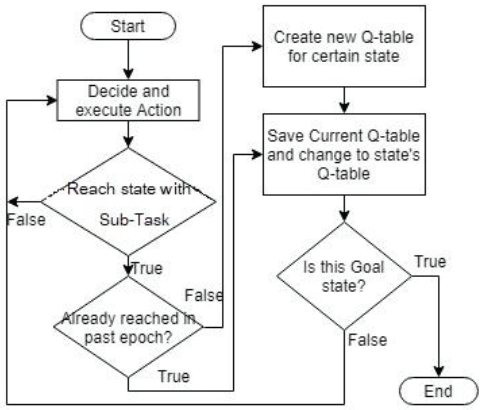

Fig. 3 Multi Q-table Q-learning Algorithm

IV. Experimental Step

A. Premade Maze

In this work, a maze is generated once. The maze is static during training and testing procedure. The maze's rules are defined as follows:

- Sub-tasks are Treasures and Pits. The Treasure gives reward = +1. The pit gets penalty that is reward = -1.
- After collecting an item, the item will disappear for the entire epoch and will come back after the end of epoch.
- Agent always starts at bottom left and exit always at top right.
- Agent cannot walk out maze's boundaries or walk through obstacles. Choosing the mentioned action keeps the agent stay in the previous state.
- Every time agent chooses and executes an action that reach blank state, the agent will get penalty of -0.05 reward.

B. Experimental Setup

An experiment contains 100 runs and total number of epochs is set to 200. For each experiment, different annealing steps which are 2,000 5,000 and 7,000 are investigated with a maze of size 8 × 10. Additionally, both algorithms are tested using the larger maze of size 16 × 16 with annealing step of 10,000 and a total number of epochs is set to 1,000. Other parameters for Q-learning are $\varepsilon = 1$, $\alpha = 0.5$ and $\gamma = 0.99$ $\varepsilon min = 0.1$.

V. Result

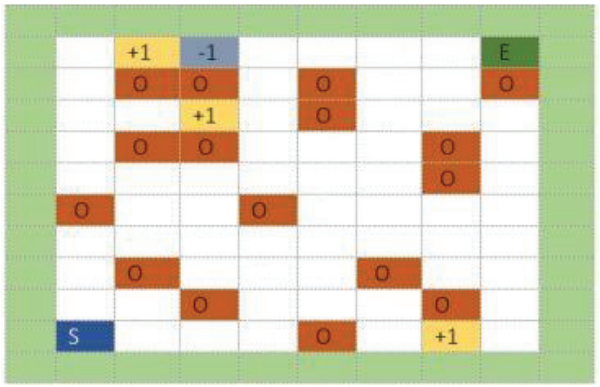

Fig. 4 A maze of size 8 × 10

The maze consist of state with +1 as a Treasure, Treasure grant agent +1 reward, state with -1 as a Pit which will penalty -1 reward to agent and 0 as obstacles, Choosing to transit into this state will keep agent to stay in the previous state.

Agent's task is to collect all treasures before navigate to exit state with the shortest path.

For each experiment, we calculate the average cumulative reward and the average amount of time used to reach the end state.

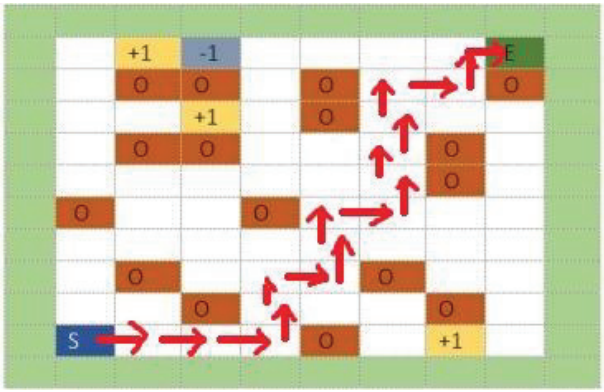

Fig. 5 Conventional Q–learning result

Fig. 5 shows the result from using conventional Q-learning to navigate through the maze with sub-task. From this result we can say that this algorithm work perfectly to navigate through the exit state but for maze with sub-task, this algorithm completely ignore the sub-task resulting in the low cumulative reward in the final result.

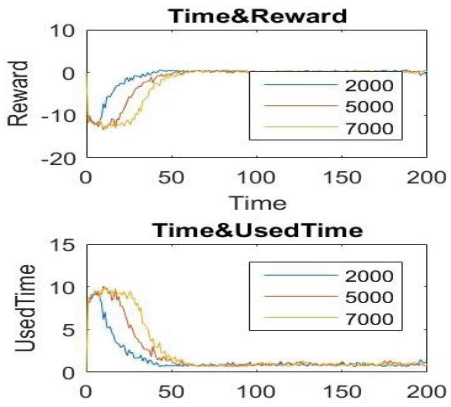

Fig. 6 Results of the Conventional Q-Learning

Fig. 6 shows the result of the experiment for conventional Q-learning. The number of reward agent can achieve for each epoch is increasing for the later epoch in the run. For annealing = 2,000 is the fastest value to reach the highest reward point for this algorithm.

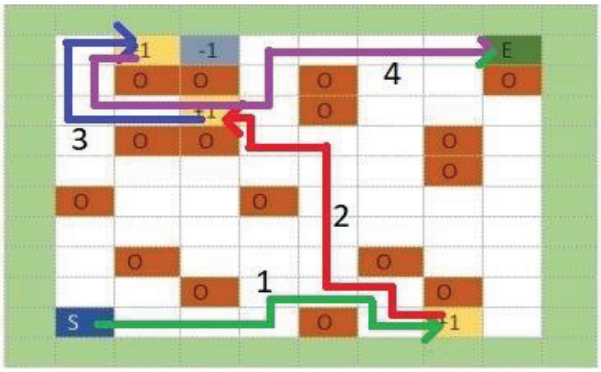

Fig. 7 Multi Q-table Q-learning results

Fig. 7 shows results obtained optimal path using the proposed method. The experimental results show that agent can achieve the sub-task and can also avoid walking into pit states before navigating itself to the end state of the maze.

Fig. 8 shows the results of experiment for MQQ algorithm. The flow of reward graph is similar to conventional Q-learning. But the used times are different. The used time of the proposed method is steadily decreased compared to the conventional Q-learning. This is due to the conventional needs the great amount of time to solve a reward loop problem at the beginning of the run.

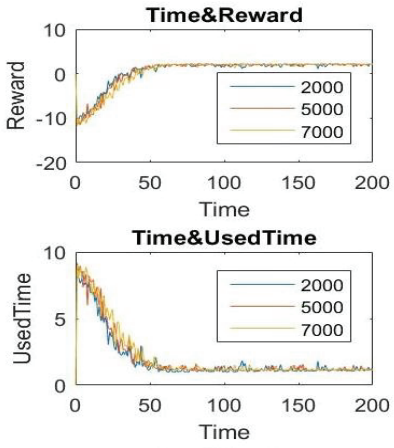

Fig. 8 Results of Multi Q-Table Q-Learning

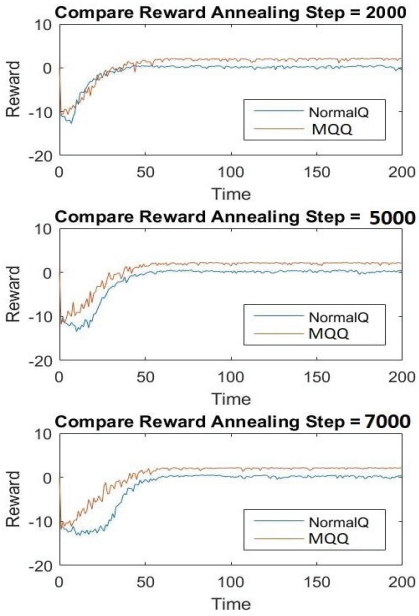

Fig. 9 Comparisons of the average obtained rewards of both algorithms

Fig. 9 shows the comparison reward between both algorithm and all *AnnealingStep* value. Result shows that MQQ yield the better amount of reward for all *AnnealingStep*. Because MQQ learn not only navigate through exit point but can also collect treasure throughout the maze too. Apparently, to collect the reward from around the map, agent also need more step and time to take as shown in Fig. 10

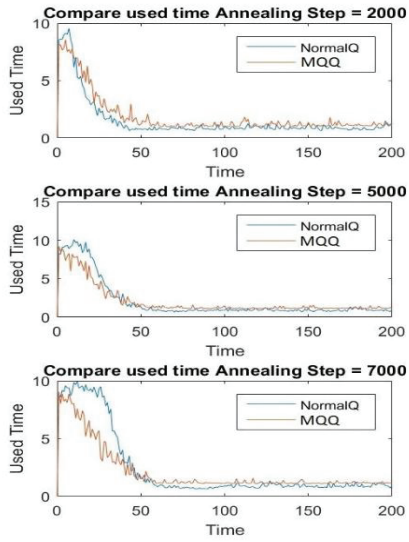

Fig. 10 The used time of both algorithms

Next we perform an experiment in the larger maze of size 16 x 16

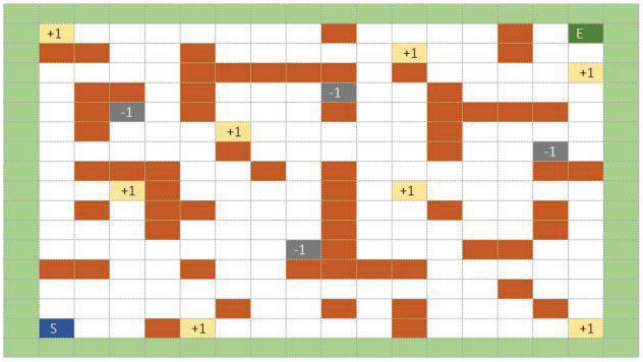

Fig. 11 A maze of size 16 × 16

In this experiment we use only one Annealing step value as mentioned in experimental setup section.

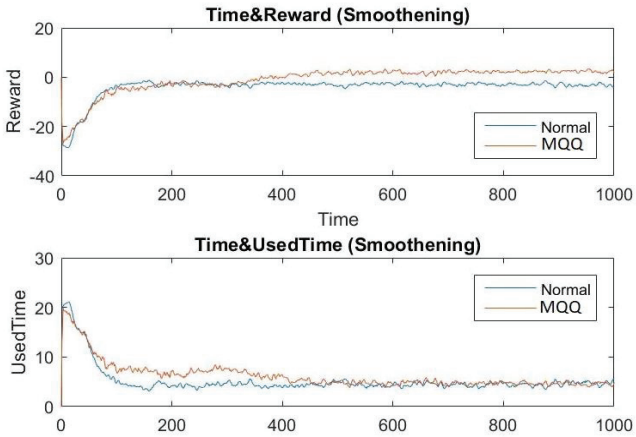

Fig. 12 Results obtained from traning with a maze of size 16 × 16

Fig. 12 shows the result from the experiment in larger maze. The amount of reward MQQ can achieve is greater than conventional Q-learning. The interesting point from this experiment is that, both algorithm use the same amount of time to reach the exit point in the later epoch. Because in this maze there are some sub-tasks with +1 reward along the shortest path. So conventional Q-learning has to deal with reward loop more often than the smaller maze. However, MQQ always creates new Q-table depending on number of sub-tasks in environment. For the problem with high complexity, this algorithm will consume a large number of space.

VI. CONCLUSION

This paper proposes Multi Q-Table Q-Learning to address a problem of finding STP in a maze with sub-tasks. The conventional Q-learning often loops back to sub-task such as treasure state due to its positive reward. In this paper, this problem is called reward loop. To address this problem, agent needs to know that an item will be deleted for the entire epoch once it has been collected. This can be accomplished by changing Q-table every time agent collects a sub-task. The new Q-table makes agent to continuously navigate to the next nearest sub-reward or exit state. From our experimental results, the proposed method can achieve the goal state with highest accumulative sub-reward. The proposed method is sometime stuck at the local maximization due to unsatisfied exploring time.

Our future work is on improving efficiency of the exploration phase without increasing exploring steps.

ACKNOWLEDGMENT

This research is financially supported by Thailand Advance Institute of Science and Technology (TAIST), National science and Technology Development Agency (NSTDA), Tokyo Tech Program, Kasetsart University under TAIST Tokyo Tech Program.

REFERENCES

[1] M. McPartland and M. Gallagher, "Learning to be a Bot: Reinforcement Learning in Shooter Games.," in Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, 2008.

[2] M. Asada and T. Yasutake, "Behavior acquisition by multi-layered reinforcement learning," in IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028), Tokyo, Japan, 1999, pp. 716-721 vol.6., 1999.

[3] W. Toshihiko and S. Toru, "Instruction for Reinforcement Learning Agent Based on Sub-Rewards and Forgetting," in International Conference on Fuzzy Systems, Barcelona, 2010, pp. 1-7., 2010.

[4] C. Holmg˚ard, A. Liapis, J. Togelius and G. Yannakakis, "Personas versus Clones for Player Decision Modeling," in Entertainment Computing – ICEC 2014. ICEC 2014. Lecture Notes in Computer Science, vol 8770 DOI = https://doi.org/10.1007/978-3-662-45212-7_20, 2014.

[5] O. D. and K. S., "Implementation of Q-Learning algorithm for solving maze problem," in 2011 Proceedings of the 34th International Convention MIPRO, Opatija, 2011, pp. 1619-1622., 2011.

[6] C. J. C. H. Watkins, "Learning from Delayed Rewards," in Ph. D dissertation, King's College, 1989.

[7] A. Tijisma, M. Drugen and M. Wiering, "Comparing Exploration Strategies for Q-learning in Random Stochastic Mazes," in 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, 2016, pp. 1-8. doi: 10.1109/SSCI.2016.7849366, 2016.

[8] T. Brys, A. Harutyunyan, P. Vrancx, M. E. Taylor, D. Kudenko and A. Nowe, "Multi-Objectivization of Reinforcement Learning Problems by Reward Shaping," in International Joint Conference on Neural Network (IJCNN), Beijing, China, 2014.

[9] R. Bellman, A Markov decision process, Indiana University Mathematics Department, 1957.

[10] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," in IEEE Transactions on Neural Networks DOI = 10.1109/TNN.1998.712192, 1998.

[11] H. v. Hado, "Double Q-Learning," in advances in Neural Information Processing system 23(NIP2010), VVancouver, British Columbia, Canada, 2010.