A Comparative Study of Reinforcement Learning Agents on Non-Stationary Bus Schedule

Hankyu Jang
Indiana University Bloomington
School of Informatics and Computing
hankjang@indiana.edu

Abstract

We are living in a highly non-stationary world where we learn to adapt to the changes in dynamics. Besides human, there are robots on the streets interacting with the world. In this paper, I introduce Bus Gridworld, composed of bus route and walking path which a novel way of creating a non-stationary environment in tabular setting, that can be used to test the adaptability of the agents on the non-stationary environment. The result shows that Dyna-Q and Dyna-Q+ agents adapt well to the non-stationary environment, and Sarsa agent also works well when promoted to explore at the time the environment changed.

Introduction

Everyday life, we face changes of the dynamics in the world. Assume we are on our way to the workplace. The weather was beautiful when we started walking, but soon it starts raining hard. We decided to take public transportation, but the bus doesn't arrive on time due to the heavy traffic caused by the rain. Then we changed our plan and start walking to our destination, where we see the bus passing by our way. By the time we get to the workplace, we were late for work. Hence, we failed to adapt to the changing world. If it rains hard tomorrow, we may decide to wait for the bus according to what we learned from today's experience.

Like the human, there are robots on the streets facing the changes in the environment. For instance, delivery robots are on test in Redwood City, California, and Washington D.C. Those robots deliver shopping bags of goods within walking distance. Sometimes the shortcut road where the robot was trained may be under construction, that leads the robot to explore other routes to reach the destination. Or sometimes a shortcut road can be constructed while not interfering the previous road where the robot was trained. If this is the case, it would be hard for the robot to find the shorter path since it has no trouble on the current road.

In this paper, I introduce Bus Gridworld, composed of bus route and walking path which a novel way of creating a non-stationary environment in tabular setting, that can be used to test the adaptability of the agents on the non-stationary environment. Several changes can be made in this simple domain to generate the non-stationary world: (1) the starting position, (2) the goal state, (3) the bus stop location, (4) the bus speed, and (5) the obstacles that agents cannot pass through. If there is a bus stop near the agent that leads to the terminal state, the agent should decide to take the bus. Otherwise, the agent should decide to walk.

I conducted experiments on the two different Bus Gridworld to compare how well the agents adapt to the changes in the environment. Five different agents were tested in the analysis: (1)Sarsa, (2)Expected Sarsa, (3)Q-learning, (4)Dyna-Q, and (5)Dyna-Q+. All the agents decided to explore when the shortest path, the path that includes riding the bus, was hindered by deceleration of the bus speed. In this case, planning agents adapted faster to the environment compared to the non-planning agents. However, many agents could not adjust to the world to find the shortest path when the shorter route was created without interfering the current shortest path. Only Sarsa agent adapted to the environment when it was forced to explore more when the environment changed.

Background

Markov Decision Process

When conducting a reinforcement learning experiment, one needs to make sure that the task has Markov property. The problem that satisfies this condition is called a Markov decision process (Sutton, Precup, and Singh 1999) where a state s, action a, and following one-step dynamics is in existence. For an agent given (s, a), the probability for the agent to be at the next state s' and reward r can be expressed by the following equation:

$$p(s', r|s, a) \doteq Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \quad (1)$$

Value Functions

The agents explore the environment to maximize the expected return. Hence, the agents need to figure out the state-action pairs that maximize the expected return. This information is collected as value functions where the value denotes the goodness of the action taken by the state. This value function is deterministic for each policy π, mapping from (s, a) to the probability of taking action a when in state

s, the agent follows. Let $q_\pi(s,a)$ denote the value of taking action a at state s following the policy π. $q_\pi(s,a)$ is represented by the following equation:

$$q_\pi(s,a) \doteq \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle| S_t = s, A_t = a\right] \quad (2)$$

Here, $\mathbb{E}_\pi$ stands for the expected value of the action taken at state s at time step t following the policy π. $q_\pi(s,a)$ is called action-value function for policy π.

Policy

The agents learn action-value function following the given policy. There are two types of policies: on-policy and off-policy. For on-policy methods, the agents estimate $q_\pi(s,a)$ for all state-action pairs for the behavior policy π. For off-policy methods, the learned $q_\pi(s,a)$ directly approximates the optimal action-value function q_* which is independent of its behavior policy.

Temporal-Difference Learning

Temporal-Difference Learning (Sutton 1988) agents update $q_\pi(s,a)$ each step on the current prediction. Three TD Learning agents have utilized in the experiment in this paper: Sarsa, Q-learning, and Expected Sarsa.

Sarsa agent utilizes on-policy TD control method. Sarsa agent continually estimate q_π for the behavior policy π and update π according to q_π. Let $Q_\pi(S_t, A_t)$ denote the array estimates of action-value function q_π. Then $Q_\pi(S_t, A_t)$ is updated each step by adding the following term:

$$\alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big] \quad (3)$$

Q-learning (Watkins and Dayan 1992) agent utilizes off-policy TD control method . This method enables early convergence. Here is the update term added at each step to update $Q_\pi(S_t, A_t)$:

$$\alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big] \quad (4)$$

Expected Sarsa agent is similar to Q-learning except that it utilizes the expected value of the next state-action pairs other than taking the maximum of them. Expected Sarsa could be classified to either on-policy or off-policy according to the usage of the target policy π to generate the behavior policy. $Q_\pi(S_t, A_t)$ is updated at each step by adding the following term:

$$\alpha\Big[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1})|S_{t+1}] - Q(S_t, A_t)\Big] \quad (5)$$

Planning

Apart from learning directly from interacting with the environment like TD method, agents can model the environment from the previous experience. Given the state-action pair, the model outputs the next state and the reward. Given the model, the agent can do planning which improves the policy by interacting the modeled world from experience.

```
Initialize Q(s, a) and Model(s, a) $\forall s \in S$ and $\forall a \in A(s)$
while forever do
    (a) $S \leftarrow$ current (nonterminal) state
    (b) $A \leftarrow \epsilon$-greedy($S$, $Q$)
    (c) Execute action $A$; observe resultant reward, $R$,
        and state, $S'$
    (d) $Q(S, A) \leftarrow$
        $Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
    (e) $Model(S, A) \leftarrow R, S'$ (assuming deterministic
        environment)
    for n times do
        end
        S $\leftarrow$ random previously observed state
        $A \leftarrow$ random action previously taken in $S$
        $R, S' \leftarrow Model(S, A)$
        $Q(S, A) \leftarrow$
          $Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
end
```

Algorithm 1: Tabular Dyna-Q

The planning agents utilize the learning method and the planning method together to improve the model, the value function, and the policy. Two planning agents are used in this experiment: DynaQ and DynaQ+ (Sutton 1990) agents.

Dyna-Q agent has the following two sections, (1) direct reinforcement learning and (2) model-learning. Direct reinforcement learning uses the same update rule with that of Q-learning shown in Algorithm 1 part (d). Model-learning is done by querying the previously observed state-action pair with the following next state and reward shown in Algorithm 1 part (e). Dyna-Q+ agent adds the bonus reward term to promote the exploration.

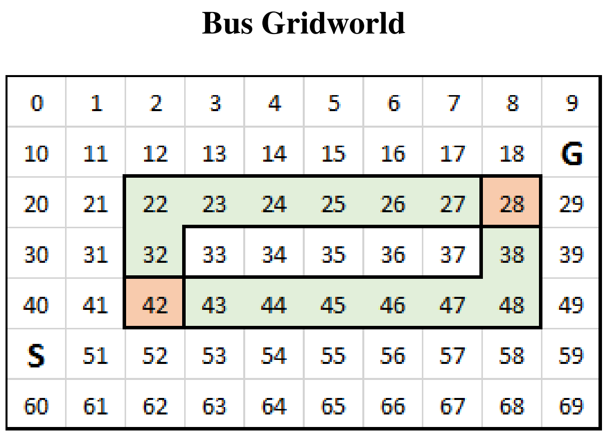

Figure 1: Bus Gridworld1

Here I introduce Bus Gridworld1 and Bus Gridworld2 of Figure 1 and Figure 2. Bus Gridworld is undiscounted, episodic task, with start and goal states denoted as S and G. The two bus stops are colored in orange, whereas the road for the bus is colored in green. The reward is -1 for every step to encourage the agent to explore. The states and actions are finite, where the agent can take one of five actions

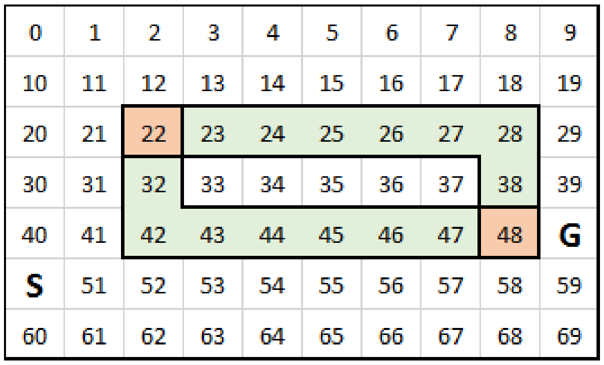

Figure 2: Bus Gridworld2

at each step: up, right, down, left, and stay.

The agent cannot leave the gridworld and cannot enter the bus road. Assume the agent's state-action pair is (s_{54}, a_{up}). Then, (s', r') = (s_{54}, -1). In Bus Gridworld1, if the agent is at the bus stop and decides to stay, then the agent can take the bus if the bus is currently at the bus stop. The (s, a) for the agent to get into the bus stop are (s_{41}, a_{right}) or (s_{52}, a_{up}).

Bus Gridworld1 of Figure 1 is the easier version of the environment for the agent to explore since one bus stop is near the start state, and the other bus stop is near the goal state. However, Bus Gridworld2 of Figure 2 is a more tricky version since the bus stop is far from the start state. In Bus Gridworld2, if the agent learned the short walking path from the start state to the goal state, then it would be hard for the agent to discover the bus stop, even for the agent using ϵ-greedy policy.

Experiments

The goal of the experiment is to understand what makes the agent in general deal well with non-stationarity. Five agents were tested in the experiment: Sarsa, Expected Sarsa, Q-Learning, Dyna-Q, and Dyna-Q+. Dyna-Q architectures are assumed to adapt well to the changing environment (Sutton 1990). The blocking experiment conducted on the navigation task (Sutton 1990) is similar to the decelerating the bus speed. Hence Dyna systems in this setting are expected to adapt to the change quickly. Likewise, the shortcut experiment conducted on the navigation task is similar to accelerating the bus speed. Thus Dyna-Q+ agent in this setting is the only agent expected to adapt to the change.

I used RL-Glue (Tanner and White 2009) which provides a standard interface that is used in experimenting reinforcement learning problems. Two experiments were conducted for each of Bus Gridworld1 and Bus Gridworld2 to test how robust the agents are to the changes in the bus speed. ϵ was set to 0 for the experiments held on Bus Gridworld1, since the bus stop at s_{42} is close to the walking path. Which means, small ϵ may affect the performance of the agents in finding the shortest path on Bus Gridworld1. Whereas, ϵ was set to 0.05 for the experiments held on Bus Gridworld2, since the bus stop is far from the shortest walking path from start to goal state. Even agents with ϵ-greedy policy would be hard to find the shortest path on Bus Gridworld2 if the bus speed accelerated.

I experimented with various parameter sets for α, n, and κ to find the right parameters for each agent. α is used for all the five agents, n for Dyna-Q and Dyna-Q+ agents, and κ for Dyna-Q+ agent only. Here are the different parameter sets utilized in the experiment:

- α = {0.05, 0.1, 0.2, 0.4, 0.8, 1.0}
- n = {0, 1, 2, 4, 8, 16, 32}
- κ = {e-02, 3e-03, e-03, 3e-04, e-04, 3e-05, e-05}

Bus Gridworld1: Bus Speed 8 $\rightarrow$ 1

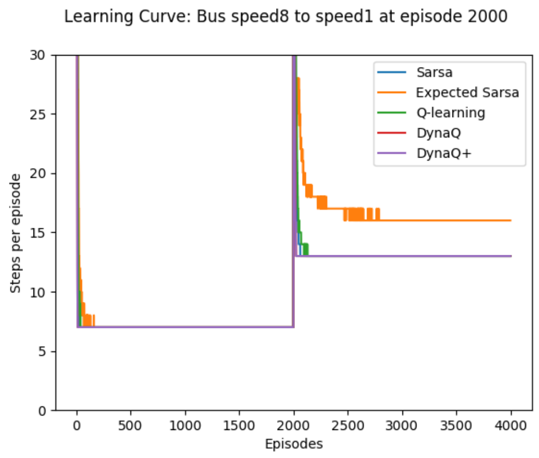

Figure 3: Performance on the Bus Speed 8 $\rightarrow$ 1 task

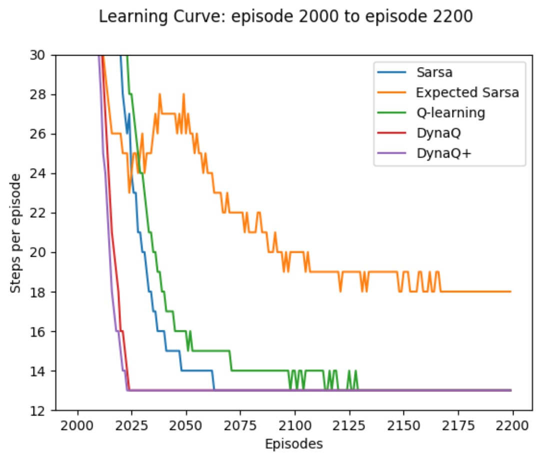

Figure 4: Performance on the Bus Speed 8 $\rightarrow$ 1 Task After Change in the Bus Speed

Bus Speed 8 $\rightarrow$ 1 experiment was conducted on the Bus Gridworld1 of Figure 1. Initially, when the bus speed was 8, all the agents found the shortest path by riding the bus.

DynaQ and DynaQ+ found the shortest path in 10 steps, followed by Sarsa, Q-learning, and Expected Sarsa. After 2,000 episodes, by the time when all the agents greedily take the bus to reach the terminal state, the bus speed was decelerated to 1.

Figure 3 and Figure 4 shows the averaged results over 300 runs for the five agents. The graph demonstrates the number of steps per episode. In the first 2,000 episodes, all the five agents found the shortest route by taking the bus. After the bus speed had decelerated to 1, the graph for Expected Sarsa was three steps above other graphs near the 4,000 episodes, showing that it was unable to find the shortest walking path. DynaQ and DynaQ+ agents precisely solved the Bus Speed $8 \rightarrow 1$ experiment in less than 25 episodes, followed by Sarsa and Q-learning agents.

Bus Gridworld1: Bus Speed $1 \rightarrow 8$

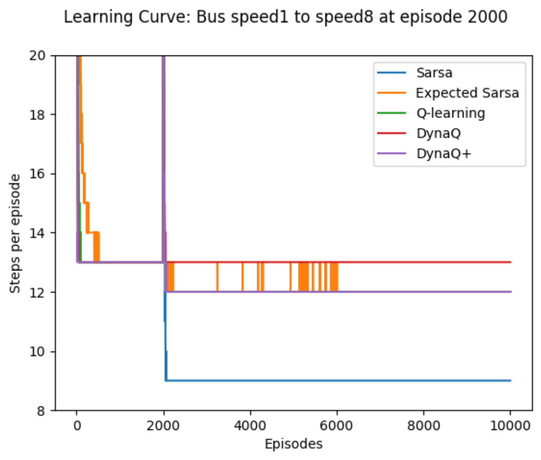

Figure 5: Performance on the Bus Speed $1 \rightarrow 8$ task

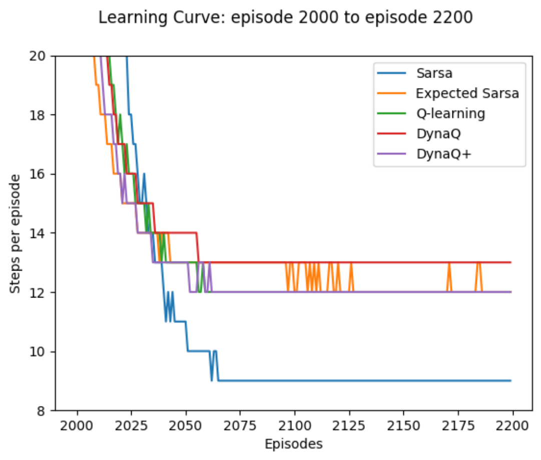

Figure 6: Performance on the Bus Speed $1 \rightarrow 8$ Task After Change in the Bus Speed

Bus Speed $1 \rightarrow 8$ experiment started with the bus speed of 1. After 2,000 steps all the agents had learned to walk the shortest path to the terminal state. Then the bus speed accelerated to 8 creating the shorter path without interfering with the previously found shortest path. ϵ was increased artificially to 0.5 to promote the five agents to explore more when bus speed had accelerated. From then, ϵ was decreased gradually by multiplying 0.95 per episode.

The Figure 5 and Figure 6 shows the results averaged over 300 runs. In the first 2,000 episodes, all the five agents found the shortest route by walking to the terminal state, though DynaQ and DynaQ+ agents took around 30 steps which are considerably faster than other three non-planning agents. After the bus speed was accelerated to 8, Figure 6 shows that Sarsa agent started exploring more than 40 steps per episode, taking much more steps compared to the other four agents. As a result, Sarsa agent found the path to ride the bus that ended up in 9 steps per episode. Expected Sarsa, Q-learning, DynaQPlus agents also found a way to ride the bus, but the result was not promising since the agents ended up in 12 steps. DynaQ agent failed to locate the path to ride the bus.

Bus Gridworld2: Bus Speed $8 \rightarrow 1$

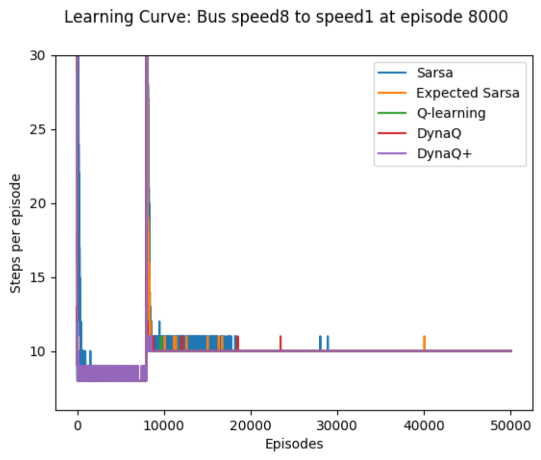

Figure 7: Performance on the Bus Speed $8 \rightarrow 1$ task

Bus Speed $8 \rightarrow 1$ experiment was conducted on the Bus Gridworld2 of Figure 2. Initially, when the bus speed was 8, all the agents found the shortest path by riding the bus. In the first 8,000 episodes, a thick line represents that the agents are exploring since the ϵ of the agents is set to 0.05. DynaQ, DynaQ+, and Q-learning found the shortest path in around 40 steps, followed by Sarsa, Q-learning, and Expected Sarsa. After 8,000 episodes, by the time where all the agents were relatively stable to take the bus to reach the terminal state, the bus speed was decelerated to 1.

Figure 7 and Figure 8 illustrates averaged results over 300 runs for the five agents. The graph shows the number of steps per episode. In the first 8,000 episodes, all the five agents found the shortest route by taking the bus. After the bus speed had decelerated to 1, DynaQ+ solved the Bus Speed $8 \rightarrow 1$ experiment in around than 30 episodes, followed by

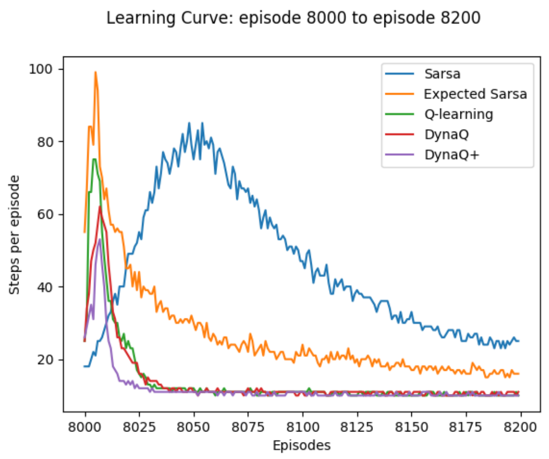

Figure 8: Performance on the Bus Speed 8 $\rightarrow$ 1 Task After Change in the Bus Speed

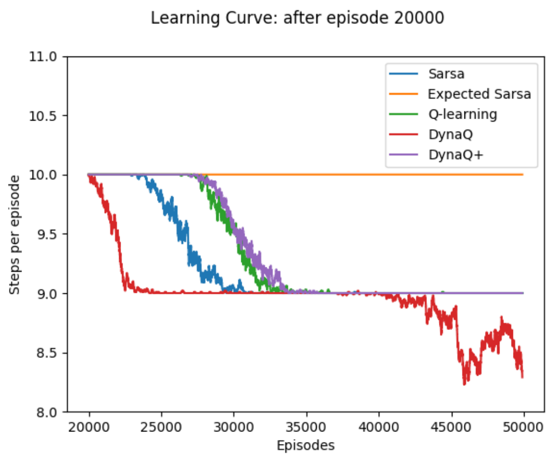

Figure 10: Performance on the Bus Speed 1 $\rightarrow$ 8 Task After Change in the Bus Speed

Q-learning, DynaQ agents. Sarsa and Expected Sarsa were slow in finding the shortest walking path here since they explored around 10,000 episodes.

Bus Gridworld2: Bus Speed 1 $\rightarrow$ 8

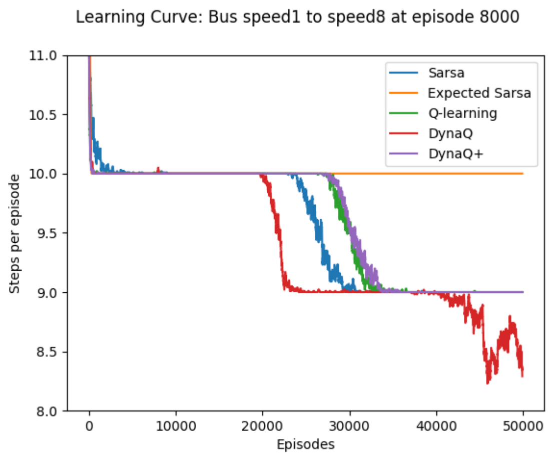

Figure 9: Performance on the Bus Speed 1 $\rightarrow$ 8 task

Bus Speed 1 $\rightarrow$ 8 experiment started with the bus speed of 1. After 8,000 steps all the agents learned to walk the shortest path to the terminal state. Then the bus speed accelerated to 8 creating the shorter path without interfering with the previously found shortest path. Unlike previous Bus Speed 1 $\rightarrow$ 8 held on Bus Gridworld1, the ϵ of the five agents stayed the same. Hence they were not forced to explore more.

The Figure 9 and Figure 10 shows the results averaged over 300 runs and then smoothed by the mean of the following 100 episodes. In the first 8,000 episodes, all the five agents found the shortest route by walking to the terminal state. After the bus speed accelerated to 8, the straight line for Expected Sarsa agent of Figure 10 reveals that it was unable to find the path to ride the bus. At around 20,000 steps, DynaQ agent found the way to ride the bus followed by Sarsa, Q-learning, and DynaQ+ agents.

Result

In the experiment where the environment got worse, Bus Speed 8 $\rightarrow$ 1, Dyna-Q agents quickly adapted to the world as expected. However in the experiment where the environment got better, Bus Speed 1 $\rightarrow$ 8, Dyna systems did not explore much to find the shorter path. However, Figure 6 shows that when the agents were given high ϵ by the time environment changed, Sarsa agent was able to find the shorter path.

From these experiments, I conclude that Dyna systems generally adapt to the non-stationary environment faster compared to other agents. However, if forced to explore more by the time the environment changes, Sarsa adapted to the environment better compared to the DynaQ+ Agent. This result contradicts the feature of Dyna-Q architectures that they are easy to adapt for use in changing environments (Sutton 1990). The discrepancy is perhaps due to the sensitivity issues of Dyna-Q agents.

Related Work

There have been needs for research in nonstationary environment, but many researchers focused on the stationary environment due to the simplicity in the stationary environment as well as it being the prerequisite to handle the nonstationary problem (Sutton, Koop, and Silver 2007). The research in improving elevator performance proved that reinforcement learning has power in handling non-stationary domain (Barto and Crites 1996). Likewise, reinforcement learning approach showed the adaptability in scheduling policies to dynamic workload behavior for self-optimizing memory controllers (Ipek et al. 2008).

Some researchers compared robustness of the agents to the changes on the environment by making one big change in the environment, either make the environment better or worse, which is similar to the experiment I conducted. One research was done on navigation task where two experiments, blocking the shortest path and creating a shortcut, were held in three Dyna systems. All the Dyna systems adapted to the blocking experiment but only the Dyna-Q+ agent successfully adapted to the shortcut experiment. (Sutton 1990). Another comparative study was held on dialogue management where the environment changed either to noise free or noisy environment (Papangelis 2012). In the study, Sarsa(λ) and Q-learning agents adapted well to the changing environment compared to Dyna-Q agent and other agents.

In practice, online learning approach is used to keep track of the changing environment. However, many reinforcement learning algorithms used in robotic platforms were trained in offline such as improving gait of a robot dog (Kohl and Stone 2004), walking robot (Tedrake, Zhang, and Seung 2004), swinging baseball task (Peters and Schaal 2008), and flying autonomous helicopter (Abbeel, Coates, and Ng 2010). Some of the few approaches done in real-time reinforcement learning in robotics are balancing task using actor-critic methods (Benbrahim et al. 1992) and conventional mobile robot (Bowling and Veloso 2003). Tile-coding architecture with adaptive exploration adapted fast in a non-stationary environment that could be used in practice (Degris, Pilarski, and Sutton 2012).

Apart from waiting the agent to find the changes in the environment, options can be given to the agents when the environment changes. Some of the approaches are using Gradient-based approach in semi-Markov options (Comanici and Precup 2010) and on policy gradient methods by treating stopping events as control actions (Levy and Shimkin 2012). There are also some papers utilizing function approximation in option discovery (Daniel et al. 2016) (Vezhnevets et al. 2016).

Conclusion

The experiments performed in the paper are restricted to the two different non-stationary Bus Gridworld. Also the limited number of the state-action pairs as well as the finite number of parameter sets tried on the five agents may have affected the result of the experiments. However, I showed that Bus Gridworld creates non-stationary environment that could be used in testing how well the agents adapt to the changing environment. I believe that more experiments such as using options should be conducted on this domain to compare the robustness of the agents on the non-stationary environment. However, I defer this for the future work.

References

Abbeel, P.; Coates, A.; and Ng, A. Y. 2010. Autonomous Helicopter Aerobatics through Apprenticeship Learning. *The International Journal of Robotics Research* 29(13):1608–1639.

Barto, a., and Crites, R. H. 1996. Improving elevator performance using reinforcement learning. *Advances in neural information processing systems* 8:1017–1023.

Benbrahim, H.; Doleac, J. S.; Franklin, J. A.; and Selfridge, O. G. 1992. Real-time learning: a ball on a beam. *International Joint Conference on Neural Networks, 1992. IJCNN* 1:98–103 vol.1.

Bowling, M., and Veloso, M. 2003. Simultaneous adversarial multi-robot learning. *IJCAI International Joint Conference on Artificial Intelligence* 699–704.

Comanici, G., and Precup, D. 2010. Optimal Policy Switching Algorithms for Reinforcement Learning. *Proc of. 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)* 709–714.

Daniel, C.; van??Hoof, H.; Peters, J.; and Neumann, G. 2016. Probabilistic inference for determining options in reinforcement learning. *Machine Learning* 104(2-3):337–357.

Degris, T.; Pilarski, P.; and Sutton, R. 2012. Model-free reinforcement learning with continuous action in practice. *... Control Conference (ACC), ...* 2177–2182.

Ipek, E.; Mutlu, O.; Martínez, J. F.; and Caruana, R. 2008. Self-optimizing memory controllers: A reinforcement learning approach. *Proceedings - International Symposium on Computer Architecture* (2):39–50.

Kohl, N., and Stone, P. 2004. Policy gradient reinforcement learning for fast quadrupedal locomotion. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004* 3(May):2619–2624.

Levy, K. Y., and Shimkin, N. 2012. Unified inter and intra options learning using policy gradient methods. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7188 LNAI:153–164.

Papangelis, A. 2012. A comparative study of reinforcement learning techniques on dialogue management. (April):22–31.

Peters, J., and Schaal, S. 2008. Natural Actor-Critic. *Neurocomputing* 71(7-9):1180–1190.

Sutton, R. S.; Koop, A.; and Silver, D. 2007. On the Role of Tracking in Stationary Environments. *The 24th International Conference on Machine Learning (ICML 2007)* 871–878.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112(1):181–211.

Sutton, R. S. 1988. Learning to Predict by the Method of Temporal Differences. *Machine Learning* 3(1):9–44.

Sutton, R. S. 1990. Integrated Modeling and Control Based on Reinforcement Learning and Dynamic Programming. *Nips*.

Tanner, B., and White, A. 2009. RL-Glue : Language-Independent Software for Reinforcement-Learning Experiments. 10:2133–2136.

Tedrake, R.; Zhang, T. W.; and Seung, H. S. 2004. Stochastic Policy Gradient Reinforcement Learning on a Simple 3D Biped. *IEEE/RSJ International Conference on Intelligent Robots and Systems* 3:2849–2854.

Vezhnevets, A.; Mnih, V.; Agapiou, J.; Osindero, S.; Graves, A.; Vinyals, O.; and Kavukcuoglu, K. 2016. Strategic Attentive Writer for Learning Macro-Actions. *arXiv* (Nips).

Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. *Machine Learning* 8(3-4):279–292.