Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

REINFORCEMENT LEARNING WITH AN ABRUPT MODEL CHANGE

Wuxia Chen
Taposh Banerjee

Department of Industrial Engineering
University of Pittsburgh
3700 O'Hara Street
Pittsburgh, PA 15261, USA

Jemin George
Carl Busart

DEVCOM Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783, USA

ABSTRACT

The problem of reinforcement learning is considered where the environment or the model undergoes a change. An algorithm is proposed that an agent can apply in such a problem to achieve the optimal long-time discounted reward. The algorithm is model-free and learns the optimal policy by interacting with the environment. It is shown that the proposed algorithm has strong optimality properties. The effectiveness of the algorithm is also demonstrated using simulation results. The proposed algorithm exploits a fundamental reward-detection trade-off present in these problems and uses an algorithm for the quickest detection of the model change. Recommendations are provided for faster detection of model changes and for smart initialization strategies.

1 INTRODUCTION

We study the problem of reinforcement learning (RL) with model changes in this paper. In an RL problem, an agent interacts with an environment by taking a sequence of actions to learn the optimal way to interact and optimize a long-term reward criterion (Sutton and Barto 2018; Bertsekas 2012; Bertsekas and Tsitsiklis 1996; Meyn 2022). In many applications, the statistical or physical properties of the environment may change over time. It then becomes necessary for the agent to adapt its strategy to the changes. For example, in an inventory control problem, the decision-maker has to consider the time-varying distribution of the demands to achieve the maximum possible profit. In an autonomous driving system, an autonomous car has to derive the driving policy considering the position and velocity of other vehicles (Guan et al. 2018) and also adapt to changing weather conditions. In a recommendation system, the agent must adapt its recommendations based on changing user preferences. In the framework of Markov decision processes (MDP), a change in the environment may correspond to a change in the transition probabilities of the Markov process being controlled or a change in the reward process.

The problem of RL in a nonstationary environment has been extensively studied in the literature. We refer the readers to (Banerjee et al. 2017) and the references therein to review the literature. Some more recent references are discussed below. In the MDP context, if the transition probabilities of the model are known and the distribution of the change points (the times at which the model changes) are also known, then the problem can be reformulated in a Partially Observable MDP (POMDP) framework where a hidden state can be used to represent the true model. However, such model information is rarely known in practice.

In this paper, we provide a model-free solution to this problem and demonstrate its performance through examples. The proposed solution is based on strong theoretical arguments. Specifically, our contributions are as follows:

1. We first argue that under reasonable assumptions it is ϵ-optimal to execute the optimal policy for the learned model, use a quickest change detection (QCD) algorithm (Veeravalli and Banerjee 2014) to detect the model changes, and switch to learning a new model after a model change is detected. The ϵ-optimality is established by comparing the performance with that of an oracle that knows the location of change points, see Section 2.

979-8-3503-6966-3/23/$31.00 ©2023 IEEE

2. Next, we show that the policy that is optimal for optimizing rewards may not be optimal to detect the model change. Thus, there exists a trade-off between detection and immediate reward optimization that can be exploited to optimize the overall reward. Our proposed algorithm exploits this trade-off, see Section 4.
3. In the above context, we show that in problems like inventory control, there exists a universal policy that helps detect the model change fastest. The universal policy there corresponds to the policy that keeps the inventory full at all times, see Section 6.3.
4. We also show that we can use the structural results from the MDP literature to initialize the system after a change is detected, see Section 6. We demonstrate through simulation results that this leads to faster convergence and better overall reward, see Section 6.2.

The existence of the reward-detection trade-off was first reported in (Banerjee et al. 2017), where a model-based solution is provided. We show in this paper that the benefit of this trade-off can be exploited even in the model-free setting by carefully designing the algorithm. In addition, in this paper, we show the existence of universal change detection policies and also discuss smart initialization strategies.

We note that it is sometimes possible for the agent to detect the model or environment change using an external sensor. For example, a change in driving conditions (e.g., weather, friction, or traffic conditions) for autonomous cars can often be detected using external sensors. The more challenging problem is when the change in the model can only be observed through the state of the system. For example, a change in user preference or demand may not always be detected using an external sensor. In this paper, we focus on the latter problem.

While the problem has been extensively researched in the literature, its natural analytical complexity has made it challenging to solve directly. A lot of previous works have focused on developing approximate solutions. For instance, (Hadoux et al. 2014) and (Dayan and Sejnowski 1996) have reformulated the problem as a Partially Observable MDP (POMDP) and utilized approximate POMDP to solve the problem. In (Da Silva et al. 2006) and (Doya et al. 2002), they keep the estimates of the current MDP parameters and use the next state or reward to evaluate whether the parameters of the current MDP have changed. Other approaches, such as hidden mode MDPs and mixed observable MDPs (MOMDP), have been employed to effectively capture the transition between distinct MDPs, and obtain an approximation solution, see (Chades et al. 2012) and (Choi et al. 2001). The changes in the properties of MDPs, such as transition kernels or rewards, will lead to alterations in the state-action sequence. In (Allamaraju et al. 2014) and (Hadoux et al. 2014), sequential detection methods were employed where the optimal policy for each MDP was executed, and a change detection algorithm was utilized to detect model changes, but they have not paid attention to the detection-reward trade-off. Other papers where a QCD approach is considered are (Dahlin et al. 2023; Chen et al. 2022). A formal and extensive comparison with other proposed solutions is part of our future work. We see our method as another candidate for an off-the-shelf algorithm with theoretical guarantees that a user can try in their RL problem.

2 PROBLEM FORMULATION AND ϵ-OPTIMAL POLICIES

Suppose we have a family of Markov Decision Processes $\{\mathbf{M}_\theta\}$, where θ takes value in some index set Θ. For each θ, one MDP $\mathbf{M}_\theta = (S, A, T_\theta, R_\theta)$ is defined by a tuple with four components: state space S, action space A, transition kernel T_θ, and reward function R_θ (Banerjee et al. 2017). We observe a sequence of states $\{S_t\}$, and for each observed state S_t, we make a decision A_t. For each state-action pair (S_t, A_t), the next state S_{t+1} is acquired according to the transition kernel T_θ, where

$$T_\theta(s,a,s') = \mathbf{P}(S_{t+1} = s' | A_t = a, S_t = s).$$

The reward $R_\theta(S_t, A_t, S_{t+1})$ is acquired after observing the next state S_{t+1}. When the context is clear, we simply refer to the reward at time t by R_t.

In a non-stationary environment, the transition kernel and reward structure change over time. For simplicity and ease of exposition, in this paper, we restrict our attention to only one change point. At some time γ, the MDP parameter changes from $\theta = \theta_0$ to $\theta = \theta_1$:

$$\mathbf{P}(S_{t+1} = s' | A_t = a, S_t = s) = \begin{cases} T_{\theta_0}(s,a,s'), & t < \gamma, \quad \text{Model } \mathbf{M_0} \\ T_{\theta_1}(s,a,s'), & t \geq \gamma, \quad \text{Model } \mathbf{M_1}. \end{cases}$$

A policy is defined as the potentially infinite-length vector of Markov maps:

$$\Pi = [\mu_0, \mu_1, \dots],$$

where each μ_t is a map from state S_t to action A_t. If the model is stationary and the parameter θ remains the same, then one of the classical ways to solve the MDP problem is to seek a policy to maximize the long-term discounted reward:

$$J^*_\theta(s_0) = \max_\Pi \mathbf{E}_\theta \left[\sum_{t=0}^{\infty} \beta^t R_t \,\middle|\, S_0 = s_0 \right].$$

where $\beta \in (0,1)$ is a discount factor and the expectation is with respect to the true θ. We will use $\Pi^*_\theta = [\mu^*_\theta, \mu^*_\theta, \dots]$ to denote the optimal stationary policy for this problem when the true model parameter is θ. In a non-stationary environment where the MDP changes from $\mathbf{M_0}$ to $\mathbf{M_1}$ (θ_0 to θ_1) at change point γ (γ is unknown to the agent), we modify the discounted cost problem as

$$J^*_{\theta_0,\theta_1}(s_0) = \max_\Pi \mathbf{E}_{\theta_0,\theta_1} \left[\sum_{t=0}^{\gamma-1} \beta^t R_t + \sum_{t=\gamma}^{\infty} \beta^{t-\gamma} R_t \,\middle|\, S_0 = s_0 \right]. \tag{1}$$

This way of resetting the discounting gives equal weight to the performance of a policy before and after the change. We now define the concept of an oracle:

Definition 1 (Oracle Policy) A policy is called an oracle policy if it has knowledge of the change point γ, and executes the policy $\Pi^*_{\theta_0}$ before the change and the policy $\Pi^*_{\theta_1}$ after the change.

It is clear that if the change point γ is large enough, the discounted reward for an oracle policy is approximately equal to the value $J^*_{\theta_0,\theta_1}(s_0)$ for any initial state s_0. We now show that using a quickest change detection algorithm (Veeravalli and Banerjee 2014) one can achieve the performance of an oracle under modest assumptions on the problem.

Definition 2 (Quickest Change Detection (QCD) Algorithm and QCD-based policy) By a QCD algorithm we mean a stopping time τ whose value is decided based on the sequence of states $S_0, S_1, \dots,$ actions $A_0, A_1, \dots,$ and rewards $R_0, R_1, \dots.$ At time τ we declare that a change in the model has occurred. A policy that employs a QCD stopping rule to detect change is called a QCD-based policy.

We define the information number (Banerjee et al. 2017; Lai 1998)

$$I_{\theta_0,\theta_1} = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} \log \frac{T_{\theta_1}(S_k, A_k, S_{k+1})}{T_{\theta_0}(S_k, A_k, S_{k+1})}, \quad \text{when } \gamma = 1. \tag{2}$$

Theorem 1 (ϵ-optimality of QCD-based policy) When the transition functions are known, then to achieve ϵ optimality in the problem in (1), it is enough to search over QCD-based policies.

Proof. We only provide a sketch of the proof. We construct a policy and show that under certain conditions it is ϵ-optimal for the problem in (1). Consider the policy that initially employs $\Pi^*_{\theta_0}$ before the change, detects the change using a QCD stopping rule, and then switches to the policy $\Pi^*_{\theta_1}$ after the change is detected. If the rewards are bounded by M and the optimal QCD algorithm is used, then the performance of this algorithm will be within $\mathbf{E}(\tau - \gamma)M + \delta$ of the oracle. Here δ is a small positive constant. Also, the expectation is taken with respect to the distribution of the underlying Markov process. This distribution depends on the policy chosen. It is well-known that the detection delay of the optimal algorithm is inversely proportional to the information number I_{θ_0,θ_1}. Thus, if this number is large enough, the term $\mathbf{E}(\tau - \gamma)M$ will be small enough. Since the Oracle policy is ϵ-optimal, so is the proposed policy. □

In practice, we do not know the models and hence cannot directly use the optimal policy and also cannot employ the optimal QCD algorithm. However, we can use algorithms like Q-learning to learn the optimal policy and use nonparametric methods in QCD to achieve the oracle performance. In the rest of the paper, we make the assumption that the change point γ is large enough so that the Q-learning algorithm has a reasonable amount of time to converge to the optimal policy. In other words, we assume that the stationarity is slowly changing.

3 Q-LEARNING ALGORITHM WITH DECREASING EPSILON GREEDY ACTION SELECTION

We use a modified version of the classical Q-learning algorithm (Watkins and Dayan 1992; Bertsekas and Tsitsiklis 1996) to learn the optimal policy for each model, before and after the change. We provide a brief overview of the modified Q-learning algorithm here.

It is well-known that the optimal reward function J^*_θ satisfies the Bellman equation.

$$J^*_\theta(s) = \max_a \sum_{s'} T_\theta(s,a,s')\Big[R_\theta(s,a,s') + \beta J^*_\theta(s')\Big].$$

The Q-function is defined as

$$Q^*_\theta(s,a) = \sum_{s'} T_\theta(s,a,s')\Big[R_\theta(s,a,s') + \beta J^*_\theta(s')\Big].$$

With the definition, the Q-function also satisfies a fixed-point equation given by

$$Q^*_\theta(s,a) = \sum_{s'} T_\theta(s,a,s')\Big[R_\theta(s,a,s') + \beta \max_{a'} Q^*_\theta(s',a')\Big].$$

The problem of Q-learning is to estimate, for any fixed θ, the optimal Q-function $Q^*_\theta(s,a)$ without knowing the transition function $T_\theta(s,a,s')$. In the Q-learning algorithm this estimation is done using a stochastic approximation algorithm (Borkar 2022; Harold, Kushner, and Yin 1997).

For a sequence of states and actions $S_0, A_0, S_1, A_1, S_2, A_2, \ldots$, the Q-learning algorithm estimates the Q-function for each state-action pair using the updates

$$TD \leftarrow R_t + \beta \max_a Q(S_{t+1}, a) - Q(S_t, A_t)$$
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha TD,$$

where $Q(S_t, A_t)$ is the current Q value of state-action pair (s,a), $\max_a Q(S_{t+1}, a)$ is the estimation of optimal future value, R_t is the received reward when taking action A_t at state S_t, β is the discount factor ($0 \le \beta \le 1$), and α is the learning rate ($0 < \alpha \le 1$). For the Q-learning to converge to the optimal Q^*, we must visit each state-action pair infinitely often (Bertsekas and Tsitsiklis 1996). This is achieved by using an ϵ-greedy strategy where a random action is chosen with probability ϵ and the optimal action (based on current estimates of Q) is chosen with probability $1-\epsilon$. To ensure faster convergence in a nonstationary environment, we use a variant of Q-learning in which the learning rate and the exploration rate are reduced over time. The entire algorithm is given in Algorithm 1.

4 QUICKEST CHANGE DETECTION ALGORITHMS AND EFFECT OF POLICY

To detect a model change using the state and reward processes, we use a quickest change detection algorithm (Banerjee et al. 2017; Veeravalli and Banerjee 2014; Tartakovsky et al. 2014) . The basic ideas are from (Wald 2004) and (Shiryaev 1963). If the transition kernels $T_{\theta_0}(s,a,s')$ and $T_{\theta_1}(s,a,s')$ are known and it is also known that the model will change from parameter θ_0 to θ_1, then we can use the generalized cumulative sum (CUSUM) algorithm from (Lai 1998). In the following algorithm, referred to as the Shiryaev algorithm (Tartakovsky and Veeravalli 2005), we compute a sequence of statistics

$$W_n = \max_{1\le k\le n} \sum_{i=k}^{n} \log \frac{T_{\theta_1}(S_{i-1}, A_{i-1}, S_i)}{T_{\theta_0}(S_{i-1}, A_{i-1}, S_i)},$$

Algorithm 1: Q-learning algorithm with decreasing epsilon greedy action selection

```
Data  : initialized Q-table, initial learning rate $\alpha_0$, initial exploration rate $\epsilon$, discount factor $\beta$,
        cut-off greedy probability $\epsilon_c$, cut-off learning rate $\alpha_c$.
Result: a trained Q-table that gives optimal (or near-optimal) policy
initialization: set $S_0 = 0$, learning rate $\alpha = \alpha_0$, $\epsilon = \epsilon_0$
for $k$ = 1 : time-horizon do
    $c \leftarrow random(0,1)$
    if $c < \epsilon$ then
        $a \leftarrow random.choice$(action space A)
    else
        $a \leftarrow \arg\max_a(Q(s,\cdot))$
    $TD \leftarrow R_k + \beta \max_{a'} Q(s',a') - Q(s,a)$
    $Q(s,a) \leftarrow Q(s,a) + \alpha TD$
    $s \leftarrow s'$
    if $\epsilon > \epsilon_c$ then
        $\epsilon \leftarrow \epsilon - \Delta$          /* decrease exploration rate */
    if $\alpha > \alpha_c$ then
        $\alpha \leftarrow \alpha - \Delta$          /* decrease learning rate */
return a new Q-table whose $\arg\max_a(Q(s,\cdot)$ gives an optimal or near-optimal policy
```

and stop the first time this statistic is above a pre-defined threshold:

$$\tau_c = \min\{n \geq 1 : W_n > A\}.$$

If γ is a constraint on the meantime to a false alarm, then it has been shown in (Lai 1998) that under mild conditions, the delay of the generalized CUSUM algorithm is given by

$$\mathbf{E}_1[\tau_c] = \frac{\log \gamma}{I_{\theta_0,\theta_1}}, \quad \textbf{as } \gamma \to \infty, \tag{3}$$

where I_{θ_0,θ_1} is defined in (2). Note that the delay is inversely proportional to the information number I_{θ_0,θ_1}. This number depends on the policy through the choice of action sequence $\{A_t\}$. Thus, different policies will lead to different values of I_{θ_0,θ_1} and hence different detection delays. In general, this characteristic and dependence on information number is shown by almost all popular QCD algorithms.

If the state and action spaces are finite, there must exist an optimal policy for quickest change detection. We note that the best policy may depend on the algorithm used for QCD.

Definition 3 (Best QCD Policy) A policy is called the best QCD policy if when applied to the system leads to the fastest detection of a model change.

Since we do not have access to the transition kernels, we cannot use the above CUSUM algorithm. If the state space is high-dimensional (or even moderate-dimensional), then tracking changes in the state-space model becomes intractable. As a result, we use the nonparametric CUSUM algorithm (Basseville and Nikiforov 1993) applied to the reward process $\{R_k\}$. The stopping rule remains the same, but we compute the statistic W_n using (for example)

$$W_n = \max\left\{0, W_{n-1} + R_n - \mu_0 - \eta\sigma_0\right\}.$$

The algorithm works as follows. The parameters μ_0, σ_0 are the (estimated) mean and standard deviation of R_n before the change, and η is a control parameter. Before the change, R_n and μ_0 cancel each other giving the reflected random walk W_n a negative drift. After the change, if the average reward increases more than $\eta\sigma_0$, then the drift becomes positive and can be detected using a large positive threshold A. Thus, η controls the amount of change in the average reward that we would like to tolerate before sounding an alarm. We note that the notion of the Best QCD policy is well-defined even when we use the nonparametric CUSUM algorithm: it is the policy that leads to the fastest delay when using the algorithm.

5 ϵ-OPTIMAL POLICIES AND EXPLOITING REWARD-DETECTION TRADE-OFF

Based on the result in Theorem 1 on the ϵ-optimality of QCD-based policies, we can argue that it is enough to restrict our search to this class of policies. In addition, the discussion in the previous section suggests that the reward-detection trade-off should be exploited and the Best QCD policy should be used to achieve better performance. In (Banerjee, Liu, and How 2017) it was shown that in the model-based setting, this exploitation is possible and leads to better rewards. It is not clear *a priori* that this trade-off can be used even when the model parameters are not known. In addition to the fact that using the Best QCD policy is not optimal for rewards, we also learn the best policy locally using Q-learning.

We show in this paper that exploitation is possible even in the model-free or RL setting. To demonstrate this, we compare two basic algorithms, one in which the Best QCD policy is used, and another, in which it is not used.

5.1 Single-Threshold Change Detection: A Policy without Using Best QCD Policy

In this section, we propose an end-to-end algorithm for RL with model changes. We call the algorithm the Single-Threshold Adaptive Q-Learning (STAQL) algorithm. In STAQL, we first learn the optimal policy

Algorithm 2: Single-Threshold Adaptive Q-Learning (STAQL) algorithm

```
Presets: threshold A, and stabilizer $\eta$
Result : a detected change point $\hat{\gamma}$, discounted reward
initialization $S_0 = 0, w = 0$
found=False
set a smart initial Q-table according to the demand
for $t$ = 1 : time-horizon do
    do one step Q-learning, updating the Q-table according to the transition kernel
    document each step reward $R_t$
    if $t == \delta$ then
        take the single step reward $R[\tau : \delta]$ as the bench mark, and compute the mean and
          standard deviation $\mu_0 = mean(R[\tau : \delta])$, $\sigma_0 = sd(R[\tau : \delta])$
    if $t > \delta$ & not found then
        compute the change detector $w$
        $w \leftarrow \max(0, w + R_t - \mu_0 - \eta sd_0)$ (if average reward changes from low to high)
        or $w \leftarrow \min(0, w + R_t - \mu_0 + \eta sd_0)$ (if average reward changes from high to low)
        if $|w| > A$ then
            found=True
            /* change is detected!                                        */
            document the detected change $\hat{\gamma} \leftarrow t$
            reset Q-table according to the next stage demand
            reset learning parameters $\epsilon$, $\alpha$
return detected change time $\hat{\gamma}$, discounted reward
```

for $\mathbf{M}_0$ using our Q-learning algorithm discussed in Algorithm 1. We initialize the Q-matrix using numbers that can help with achieving faster convergence and more rewards. In the next section, we discuss how to smartly initialize an inventory control system. The system starts at time 0. Let τ be the time at which the Q-learning converges and learns the optimal policy for model $\mathbf{M}_0$. We can learn the time τ through simulations and experience with the system. From time τ to another time δ we learn the baseline reward statistics and estimate

$$\mu_0 = mean(R[\tau:\delta]), \qquad \sigma_0 = sd(R[\tau:\delta]).$$

Here $R[\tau:\delta]$ denotes the vector of rewards collected from time τ to δ. Starting time δ, we apply the nonparametric CUSUM algorithm to detect the model change. Here, we are assuming that γ, the change point, satisfies $\gamma \gg \delta$. If the average reward is expected to change from low to high, we use the statistical update

$$W_n = \max\left\{0, W_{n-1} + R_n - \mu_0 - \eta\sigma_0\right\}. \tag{4}$$

or if the average reward is expected to change from high to low, we use the following instead,

$$W_n = \min\left\{0, W_{n-1} + R_n - \mu_0 + \eta\sigma_0\right\}. \tag{5}$$

If the average reward can change in any direction, we can use both statistics in parallel. The change is declared at

$$\hat{\gamma} = \min\{n \geq \delta : W_n > A\}.$$

After the change is detected at $\hat{\gamma}$, we reinitialize the Q-matrix to smart values and start the Q-learning again (Algorithm 1) to learn the optimal policy for model $\mathbf{M}_1$. The STAQL algorithm is written in algorithmic form in Algorithm 2 and can also be represented using the following equation:

$$\Pi_{\text{STAQL}} = (\underbrace{\tilde{\pi}, \ldots, \hat{\pi}_0, \hat{\pi}_0, \ldots}_{|w| \leq A, \hat{\gamma}-1}, \underbrace{\tilde{\pi}, \ldots, \hat{\pi}_1, \hat{\pi}_1, \ldots}_{|w| > A, \hat{\gamma} \text{ onward}}),$$

where $\tilde{\pi}$ is the Markov map generated from the initial Q-table, $\hat{\pi}_0$ (respectively, $\hat{\pi}_1$) is the optimal policy for model $\mathbf{M}_0$ (respectively, $\mathbf{M}_1$) learned using Q-learning.

5.2 Two-Threshold Change Detection: Policy Exploiting Reward-Detection Trade-Off

In this section, we propose another end-to-end algorithm for RL with model changes. We call the algorithm the Two-Threshold Adaptive Q-Learning (TTAQL) algorithm. The TTAQL exploits the reward-detection trade-off and uses the Best QCD policy to optimize the overall reward.

Similar to STAQL, the TTAQL algorithm also uses a smart initialization followed by Q-learning to learn the optimal policy for $\mathbf{M}_0$. Again, similar to the STAQL algorithm, the TTAQL algorithm learns the baseline reward statistics using

$$\mu_0 = mean(R[\tau:\delta]), \qquad \sigma_0 = sd(R[\tau:\delta]).$$

Starting time δ, however, a two-threshold version of the nonparametric CUSUM algorithm is applied to detect the model change. To clarify concepts, we assume that the average reward is expected to decrease. We will then use the statistic

$$W_n = \min\left\{0, W_{n-1} + R_n - \mu_0 + \eta\sigma_0\right\}.$$

The change is declared at

$$\hat{\gamma} = \min\{n \geq \delta : W_n > A\}.$$

Algorithm 3: Two-Threshold Adaptive Q-Learning (TTAQL) algorithm

```
Presets: quick change detection policy $\pi_{qcd}$, threshold B, threshold $\tilde{A}$ and stabilizer $\eta$
Result : a detected change point $\hat{\gamma}$, discounted reward
initialization $S_0 = 0, w = 0$
found=False
suspect=False
set a smart initial Q-table according to the demand
for  $t$ = 1 : time-horizon do
    if not suspect then
        do one step Q-learning, updating the Q-table according to the transition kernel
    else
        apply $\pi_{qcd}$ policy
        update the state $S_t$ according to the transition kernel
        /* do not update the Q-table while applying QCD policy            */
    document each step reward $R_t$
    if $t == \delta$ then
        take the single step reward $R[\tau : \delta]$ as the benchmark, and compute the mean and
          standard deviation $\mu_0 = mean(R[\tau : \delta])$, $\sigma_0 = sd(R[\tau : \delta])$
    if $t > \delta$ & not found then
        compute the change detector $w$, using (4) or (5)
        if $|w| > B$ then
            suspect=True
        else
            suspect=False
        if suspect then
            if $|w| > \tilde{A}$ then
                found=True
                /* change is detected!                                    */
                suspect=False
                document the detected change $\hat{\gamma} \leftarrow t$
                reset Q-table, and learning parameters $\epsilon$, $\alpha$
return  detected change time $\hat{\gamma}$, discounted reward
```

However, the algorithm uses another threshold $B < A$ to choose which policy to use at any time after time δ. Specifically, if $\hat{\pi}_0$ denotes the Markov map for model $\mathbf{M}_0$ learned using Q-learning and π_{qcd} denotes the Markov map for the Best QCD policy, then we use the following strategy:

$$\text{If} \quad 0 \le |W_n| \le B, \quad \text{use map } \hat{\pi}_0 \text{ at time } n+1$$

$$\text{If} \quad B < |W_n| < A, \quad \text{use map } \pi_{qcd} \text{ at time } n+1.$$

After the change is detected at $\hat{\gamma}$, we reinitialize the Q-matrix to smart values and start the Q-learning again (Algorithm 1) to learn the optimal policy for model $\mathbf{M}_1$. The TTAQL algorithm is written in algorithm form in Algorithm 3, and can also be represented using the following equation:

$$\Pi_{\text{TTAQL}} = (\underbrace{\tilde{\pi}, ..., \hat{\pi}_0,}_{|w|<B} \underbrace{\pi_{qcd}, ...\pi_{qcd},}_{B<|w|<\tilde{A}} \underbrace{\hat{\pi}_0, \hat{\pi}_0, ...,}_{|w|<B} \underbrace{\pi_{qcd}, ...\pi_{qcd},}_{B<|w|<\tilde{A},\ \text{before } \hat{\gamma}-1} \underbrace{\tilde{\pi}, ..., \hat{\pi}_1, \hat{\pi}_1, ...}_{|w|>\tilde{A},\ \hat{\gamma} \text{ onward}}).$$

6 SIMULATION RESULTS: APPLICATION TO AN INVENTORY CONTROL PROBLEM

In this section, we apply the STAQL and TTAQL algorithms to an inventory control problem and show that the TTAQL algorithm can outperform the STAQL algorithm. We also show that the Best QCD policy for this problem is universal: there exists an interpretable policy that can detect the change fastest for any realization of the inventory control problem. We also discuss the convergence rate for smart initializations.

6.1 Inventory Control Problem

Consider the inventory control problem (Szepesvári 2010) with inventory level or state S_t, $S_t \in \{0, 1, ...N\}$, and N is the maximum inventory size of the warehouse. Let action A_t be the number of new orders in the morning of the day t, $A_t \in \{0, 1, ...N\}$. During the day, customers come with a stochastic demand D_t, where D_t is an independent and identically distributed sequence of Poisson random variables with some rate λ:

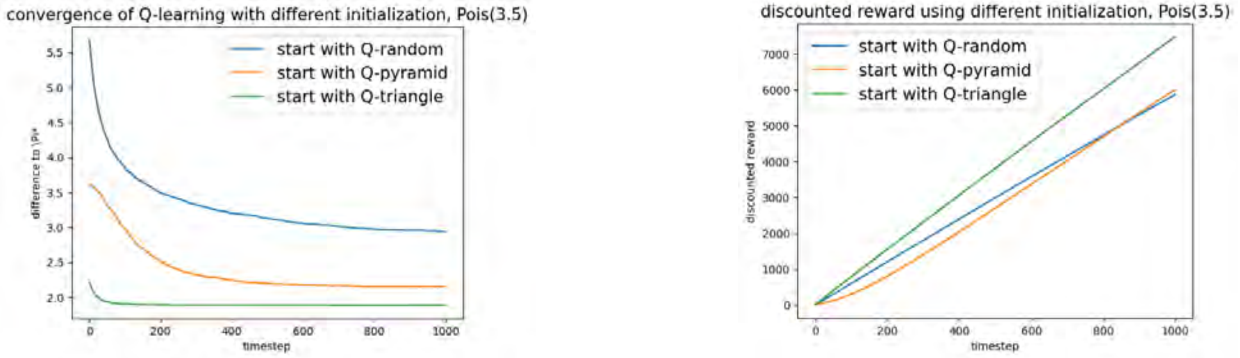
$$S_{t+1} = max(min(S_t + A_t, N) - D_t, 0).$$

The reward or income on the day t is

$$R_t = -k\mathbb{I}(A_t > 0) - c(min(A_t, N - S_t)) - hS_{t+1} + p(min(S_t + A_t, N) - S_{t+1}) - \text{rent}.$$

The income on the day t is determined as follows: there is a fixed entry cost k of ordering nonzero items and each item must be purchased at a fixed price c, so the cost associated with purchasing A_t items is $k\mathbb{I}(A_t > 0) + cA_t$. In addition, there is a cost of h for holding an unsold item. If there are x leftovers at the end of the day t, the manager will pay hx for holding the items the next morning. Finally, upon selling z items the manager receives a payment of pz. To make the warehouse running, we must have $p > h$, otherwise, there is no incentive to order new items.

We use the following set of parameters for the simulations. The parameters we use here are: warehouse capacity $N = 5$, discounted factor $\beta = 0.9999$, change point $\gamma = 1000$, $timehorizon = 5000$, stabilizer $\eta = 0.92$, $A = 6sd_0$, $B = 3.35sd_0$, $\tilde{A} = 6.67sd_0$, fixed cost $k = 0.5$, per unit cost $c = 3$, holding cost $h = 2$, profit per sale $p = 8$, and rent=4.8, initial learning rate $\alpha_0 = 0.2$, cut-off learning rate $\alpha_c = 0.05$, initial exploration rate $\epsilon_0 = 0.2$, cut-off exploration rate $\epsilon_c = 0.05$, step decent $\Delta = 0.001$. We use the $R_t, \forall t \in [500, 600]$ as the bench mark, where $\mu_0 = mean(R[500:600])$, $sd_0 = sd(R[500:600])$. All results are averaged over 10,000 iterations.



(a) (b)

Figure 1: Q-learning with different initialization.

6.2 Using Smart Initialization in Q-Learning

It is well known that the optimal Markov map in the inventory control problem is linear and nonincreasing (Bertsekas 2012). In Figure 1, we show that if we initialize the Q-tables with values that correspond to a monotonically decreasing policy that it leads to faster convergence (Figure 1a) and better overall rewards (Figure 1b). In the figure, Q-random corresponds to a randomly initialized Q-table, and Q-pyramid corresponds to a policy that is unimodal, increasing first, and then decreasing after the mode. We see this pattern as long as the demand is high. If the demand is low, we have observed that initializing with Q-random leads to the best overall reward.

6.3 A Universal Change Detection Policy

In general, for every realization of the problem, one may have to search for the Best QCD policy through simulations. In the inventory control problem, however, we show that there is a universal Best QCD policy that is Best QCD policy for every realization of the inventory problem. This policy corresponds to the one that keeps the inventory full at all times. This is intuitive since if the demand is low, it is optimal to keep the inventory low. If there is a sudden increase in demand, items may appear out of stock and a user may never place an order. The system will fail to detect a sudden increase in demand from low to high. However, if we keep the inventory full at all times, we can always capture the fluctuations in demand.

Table 1 compares the detection performances of the Best QCD policy and the learned optimal policy in the situation where the demand for $\mathbf{M}_0$ is high and then it switches to a lower demand after the change point $\gamma = 1000$. We note that in the table, the learned optimal policy $\hat{\pi}_0$ changes with the choice of demand rates. In Table 2, we show similar results when the demand of $\mathbf{M}_0$ is low and the demand for $\mathbf{M}_1$ is high.

Table 1: Best QCD policy vs $\hat{\pi}_0$: demand from high to low.

high to low	η	Best QCD policy delay	FA	$\hat{\pi}_0$ delay	FA
λ_0=4, λ_1=1.8	0.92	96	0.009	228	0.009
	0.7	26	0.008	53	0.008
λ_0=3, λ_1=1	0.9	48.9	0.0091	165	0.0096
	0.7	17.5	0.0089	22	0.0094
λ_0=3.5, λ_1=2.5	0.2	89	0.0069	170	0.0073
	0.1	109	0.0084	160	0.0101

6.4 Comparison of STAQL and TTAQL Policies

In the previous section, we showed that the Best QCD policy can detect changes faster. However, it can also cause a loss of immediate rewards. In Figure 2, we show that the TTAQL algorithm can also achieve a better overall reward. In the figure, oracle policy refers to the oracle policy discussed in Definition 1 except the policies are learned using Q-learning. The ignore policy simply ignores the change and incurs heavy losses due to a high holding cost. The table in the figure shows the expected discounted reward at the end of the horizon with Rwd(mdp1) as the reward collected beginning at the change point. In the figure on the right in Figure 2, we plot the cumulative reward beginning at the change point, averaging over 10000 realizations. The figure shows that the TTAQL performs better almost at each point in the entire horizon. To guarantee a fair evaluation of the two aforementioned methods, we maintain a false alarm rate of roughly 1%. When computing the average delay and average reward, we exclude only the false alarm instances. Additionally, if there are any cases where the agent fails to detect a change, we assume that the agent detects the change at the last possible moment.

In Figure 3, we show the results for the inventory control problem with the maximum warehouse capacity $N = 7$. Most of the parameters are the same to the $N = 5$ case, except for $\lambda_0 = 6$, $\lambda_1 = 2.5$, $\eta = 1.2$, $B = 4sd_0$, $A = 8sd_0$, and $\tilde{A} = 6.9sd_0$.

Table 2: Best QCD policy vs $\hat{\pi}_0$: demand from low to high.

low to high	η	Best QCD policy delay	FA	$\hat{\pi}_0$ delay	FA
λ_0=2,λ_1=4	0.3	19	0.0066	175	0.0101
	0.1	20	0.0088	100	0.0092
λ_0=1.5,λ_1=3.5	0.4	13	0.0074	95	0.0083
	0.3	12	0.0084	35	0.0096
λ_0=2,λ_1=3	0.2	63	0.0074	344	0.0092
	0.05	50	0.01	225	0.01

	TTAQL	STAQL	Ignore	Oracle
Rwd(mdp1)	264	185	-3210	424
Rwd(total)	8376	8310	4928	8601
avg-delay	145	227	∞	0
true-detect%	97.78	96.78	0	1
miss%	1.16	2.09		
F-A %	1.06	1.13		

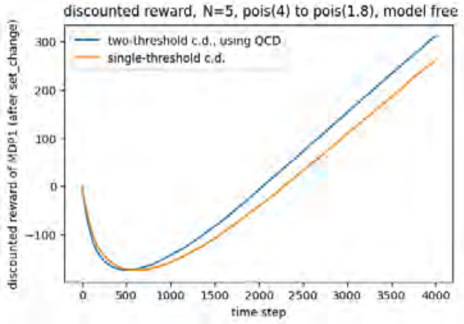


Figure 2: Discounted reward and delay using different change detection policies, $\lambda_0 = 4, \lambda_1 = 1.8$ N=5.

	TTAQL	STAQL	Ignore	Oracle
Rwd(mdp1)	603	521	-3838	718
Rwd(total)	10645	10552	6251	10812
avg-delay	63	139	∞	0
true-detect%	98.77	97.78	0	1
miss%	0.41	1.26		
F-A %	0.82	0.96		

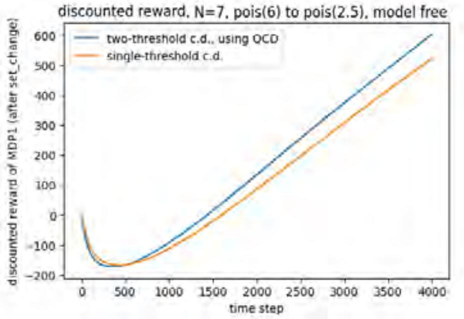


Figure 3: Discounted reward and delay using different change detection policies, $\lambda_0 = 6, \lambda_1 = 2.5$ N=7.

7 CONCLUSION

We proposed an algorithm called the Two-Threshold Adaptive Q-Learning (TTAQL) algorithm that can be used for RL with model changes. This algorithm exploits a fundamental trade-off between detection delay and immediate reward optimization that is present in RL in nonstationary environments. We also showed that this algorithm belongs to a class of policies called QCD-based policies. We argued in Theorem 1 that in the search for optimal policies, it is enough to restrict the search to QCD-based policies because one can achieve ϵ-optimality. We also showed that in some applications like inventory control, there is a universal policy that provides the fastest detection delay. This policy can be used to exploit the reward-detection trade-off. In addition, smart initialization in the Q-learning algorithm can lead to faster convergence and better overall rewards. In the future, we plan to apply the TTAQL algorithm to more complex RL problems, develop better change detection algorithms for this domain, and also develop deeper theoretical insights.

8 ACKNOWLEDGEMENT

The work of Wuxia Chen and Taposh Banerjee was supported in part by the U.S. Army Research Lab under grant W911NF2120295.

REFERENCES

Allamaraju, R., H. Kingravi, A. Axelrod, G. Chowdhary, R. Grande, J. P. How, C. Crick, and W. Sheng. 2014. “Human Aware UAS Path Planning in Urban Environments Using Nonstationary MDPs”. In *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, 1161–1167. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Banerjee, T., M. Liu, and J. P. How. 2017. “Quickest Change Detection Approach to Optimal Control in Markov Decision Processes with Model Changes”. In *Proceedings of the 2017 American control conference (ACC)*, 399–405. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Basseville, M., and I. V. Nikiforov. 1993. *Detection of Abrupt Changes: Theory and Application*. Englewood Cliffs: Prentice Hall.

Bertsekas, D. 2012. *Dynamic Programming and Optimal Control: Volume I*. Belmont, MA: Athena Scientific.

Bertsekas, D., and J. N. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Belnont, MA: Athena Scientific.

Borkar, V. S. 2022. *Stochastic Approximation: A Dynamical Systems Viewpoint*. 2nd ed. New Delhi: Hindustan Book Agency.

Chades, I., J. Carwardine, T. Martin, S. Nicol, R. Sabbadin, and O. Buffet. 2012. "MOMDPs: A Solution for Modelling Adaptive Management Problems". In *Proceedings of the AAAI Conference on Artificial Intelligence*. July 22th-26th ,Toronto, 267–273.

Chen, H., J. Tang, and A. Gupta. 2022. "Change Detection of Markov Kernels with Unknown Pre and Post Change Kernel". In *Proceedings of the 2022 IEEE 61st Conference on Decision and Control (CDC)*, 4814–4820. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Choi, S. P., D. Y. Yeung, and N. L. Zhang. 2001. "Hidden-Mode Markov Decision Processes for Nonstationary Sequential Decision Making". *Sequence Learning: Paradigms, Algorithms, and Applications* 1828:264–287.

Da Silva, B. C., E. W. Basso, A. L. Bazzan, and P. M. Engel. 2006. "Dealing with Non-Stationary Environments Using Context Detection". In *Proceedings of the 23rd International Conference on Machine Learning*. June 25th-29th, Pittsburgh, 217–224.

Dahlin, N., S. Bose, and V. V. Veeravalli. 2023. "Controlling a Markov Decision Process with an Abrupt Change in the Transition Kernel". In *Proceedings of the 2023 American Control Conference (ACC)*, 3401–3408. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Dayan, P., and T. J. Sejnowski. 1996. "Exploration Bonuses and Dual Control". *Machine Learning* 25:5–22.

Doya, K., K. Samejima, K.-i. Katagiri, and M. Kawato. 2002. "Multiple Model-Based Reinforcement Learning". *Neural Computation* 14(6):1347–1369.

Guan, Y., S. E. Li, J. Duan, W. Wang, and B. Cheng. 2018. "Markov Probabilistic Decision Kaking of Self-Driving Cars in Highway with Random Traffic Flow: A Simulation Study". *Journal of Intelligent and Connected Vehicles* 1(2):77–84.

Hadoux, E., A. Beynier, and P. Weng. 2014. "Sequential Decision-Making Under Non-Stationary Environments via Sequential Change-Point Detection". In *Proceedings of the Learning Over Multiple Contexts (LMEC) 2014*. September 10th-15th, Nancy, France.

Harold, J., G. Kushner, and G. Yin. 1997. *Stochastic Approximation and Recursive Algorithm and Applications*. New York: Springer.

Lai, T. L. 1998. "Information Bounds and Quick Detection of Parameter Changes in Stochastic Systems". *IEEE Transactions on Information theory* 44(7):2917–2929.

Meyn, S. 2022. *Control Systems and Reinforcement Learning*. New York: Cambridge University Press.

Shiryaev, A. N. 1963. "On Optimum Methods in Quickest Detection Problems". *Theory of Probability & Its Applications* 8(1):22–46.

Sutton, R. S., and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Szepesvári, C. 2010. *Algorithms for Reinforcement Learning*. San Rafael, CA: Morgan and Claypool Publishers.

Tartakovsky, A., I. Nikiforov, and M. Basseville. 2014. *Sequential Analysis: Hypothesis Testing and Changepoint Detection*. Boca Raton, FL: CRC Press.

Tartakovsky, A. G., and V. V. Veeravalli. 2005. "General Asymptotic Bayesian Theory of Quickest Change Detection". *Theory of Probability & Its Applications* 49:458–497.

Veeravalli, V. V., and T. Banerjee. 2014. "Quickest Change Detection". *Academic Press Library in Signal Processing* 3:209–255.

Wald, A. 2004. *Sequential Analysis*. North Chelmsford, MA: Courier Corporation.

Watkins, C. J., and P. Dayan. 1992. "Q-Learning". *Machine Learning* 8:279–292.

AUTHOR BIOGRAPHIES

WUXIA CHEN received her M.S. in Electrical and Computer Engineering (ECE) from the University of Texas at San Antonio. She is a Ph.D. student in the Department of Industrial Engineering at the University of Pittsburgh. Email:wuc3@pitt.edu.

TAPOSH BANERJEE received his Ph.D. in ECE in 2014 from the University of Illinois at Urbana-Champaign. He is an Assistant Professor of Industrial Engineering at the University of Pittsburgh (Pitt). Before joining Pitt, he was an Assistant Professor of ECE at the University of Texas at San Antonio. Email:taposh.banerjee@pitt.edu.

JEMIN GEORGE received his M.S. ('07), and Ph.D. ('10) in Aerospace Engineering from the State University of New York at Buffalo. He is a Research Engineer at U.S. Army Research Laboratory. Prior to joining ARL, he worked at the U.S. Air Force Research Laboratory's Space Vehicles Directorate as a Space Scholar and at the National Aeronautics and Space Administration's Langley Aerospace Research Center (NASA LaRC). Email: jemin.george.civ@army.mil.

CARL BUSART received the B.S. and M.S. degrees from Johns Hopkins University, an MBA from the University of Maryland, College Park, and a D.Eng. degree from George Washington University. He is a branch chief at the U.S. Army Research Laboratory and a member of the Institute of Electrical and Electronics Engineers (IEEE) and the Association for Computing Machinery (ACM). His research interests include artificial intelligence/machine learning (AI/ML) and secure design. Email:carl.e.busart.civ@army.mil.