2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Efficient and Real-Time Reinforcement Learning for Linear Quadratic Systems with Application to H-infinity Control

Ali Aalipour and Alireza Khani

***Abstract*— This paper presents a model-free, real-time, data-efficient Q-learning-based algorithm to solve the H_∞ control of linear discrete-time systems. The computational complexity is shown to reduce from $\mathcal{O}(\underline{q}^3)$ in the literature to $\mathcal{O}(\underline{q}^2)$ in the proposed algorithm, where $\underline{q}$ is quadratic in the sum of the size of state variables, control inputs, and disturbance. An adaptive optimal controller is designed and the parameters of the action and critic networks are learned online without the knowledge of the system dynamics, making the proposed algorithm completely model-free. Also, a sufficient probing noise is only needed in the first iteration and does not affect the proposed algorithm. With no need for an initial stabilizing policy, the algorithm converges to the closed-form solution obtained by solving the Riccati equation. A simulation study is performed by applying the proposed algorithm to real-time control of an autonomous mobility-on-demand (AMoD) system for a real-world case study to evaluate the effectiveness of the proposed algorithm.**

I. Introduction

Reinforcement learning (RL) is one of the three fundamental machine learning paradigms, alongside supervised learning and unsupervised learning, which has a long history [1]. The dynamical system's model is typically unknown in RL settings, and the ideal controller is discovered by engagement with the environment. It is fundamental for the RL algorithms to deliver assured stability and performance as the range of RL extends to more difficult tasks. Due to deep networks' inherent complexity and the intricacy of the tasks, we are still a long way from being able to analyze RL algorithms. In addition, the majority of them are neither theoretically tractable nor can their convergence be investigated.

H_∞ problem is a classical control problem where the dynamical system follows linear dynamics and the cost function to be minimized is quadratic. It is a robust control method that is implemented to attenuate the effects of disturbances on the performance of dynamical systems. It is a great benchmark for studying since the closed-form solution for H_∞ is available. Moreover, it is theoretically tractable in comparison to the RL algorithms.

As a result of the aforementioned factors, the linear quadratic (LQ) problem has received greater attention from the RL community [2], [3], [4], see also [5] for a thorough overview of RL methods and their properties for the LQ problems. In addition, the convergence of policy gradient methods for the linear quadratic regulator (LQR) problem is shown in [6]. RL also has been applied for solving optimal control problems in an uncertain environment,[7], [8], [9]. Inherently, the Q-learning algorithm does not eliminate the impacts of the probing noise, which is employed to excite the system, in the Bellman equation when evaluating the value function. The algorithm's convergence may be impacted, and this may lead to bias. In [9], two separate policies are used to update the algorithm to cancel the effects of probing noise. However, there should be enough generated data for each iteration to estimate the policies.

A. Contributions

In this paper, we propose a RL algorithm to solve the H_∞ control of linear discrete-time systems. It is model-free, real-time, and data-efficient, i.e., using a single data, the parameters of the actor and critic networks are updated. This feature results in reducing the order of computational complexity to square ($\mathcal{O}(\underline{q}^2)$) where $\underline{q}$ is the number of parameters being estimated, compared to the cube order ($\mathcal{O}(\underline{q}^3)$) in the state-of-the-art algorithms in the literature (e.g., [7], [9]). This RL algorithm does not suffer from bias if probing noise is used. Moreover, a sufficient amount of probing noise is only needed in the first iteration, i.e., the policy used to generate data, called the behavior policy is different only in the first iteration than the policy being evaluated and improved, called the estimation policy or target policy. The convergence of the proposed algorithm is shown. Moreover, we apply the proposed algorithm to an autonomous mobility-on-demand (AMoD) system which can be modeled as an H_∞ control of linear discrete-time systems, in particular, to optimize vehicle scheduling and rebalancing in the AMoD system that can be modeled as a linear discrete-time system. To fulfill customer demands, rebalancing is crucial to ensure that vehicles are distributed properly which means dispatching available empty vehicles to areas undersupplied areas. In summary, the contributions of this paper can be expressed as follows:

1) Proposed a model-free, real-time, and data-efficient algorithm to solve the H_∞ control of linear discrete-time systems.
2) Reduced the order of computational complexity form cube ($\mathcal{O}(\underline{q}^3)$) in the state-of-the-art algorithms in the literature to square ($\mathcal{O}(\underline{q}^2)$).
3) Discussed the properties of the proposed algorithm and proved its convergence.

Ali Aalipour and Alireza Khani are with the Department of Electrical and Computer Engineering and Department of Civil, Environmental, and Geo-Engineering, University of Minnesota, MN, USA {aalip002@umn.edu, akhani@umn.edu}.

*The study has been conducted in the University of Minnesota Transit Lab, through the support of various grants and sponsored projects.

4) Applied the proposed algorithm to an AMoD system which can be modeled as an H_∞ control of linear discrete-time systems.

B. Organization

The remainder of the paper is organized as follows. In Section II, the discrete-time H_∞ control problem is formulated. This section is concluded by implementing the value iteration algorithm. In Section III, the online implementation of the proposed algorithm and its properties are analyzed. Besides, the convergence of the proposed algorithm is proved. Section IV presents the problem formulation and model for the AMoD system. We present the results of the numerical case study example in Section V. Finally, the paper is concluded in Section VI.

C. Notation

$\text{vecs}(P) = \left[p_{11}, ..., p_{1n}, p_{22}, ..., p_{2n}, ..., p_{nn}\right]^T$ is the vectorization of the upper-triangular part of a symmetric matrix $P \in \mathbb{R}^{n\times n}$, and $\text{vecv}(v) = \left[v_1^2, 2v_1v_2, ..., 2v_1v_n, v_2^2, ..., 2v_2v_n, ...v_n^2\right]^T$ is the quadratic vector of the vector $v \in \mathbb{R}^n$.

II. Discrete-time (DT) H_∞ Control Problem

Consider the following linear discrete-time system

$$x_{t+1} = \mathcal{A}x_t + \mathcal{B}v_t + \mathcal{L}d_t, \tag{1}$$

where $x_t \in \mathbb{R}^{m_1}$ is the system state, $v_t \in \mathbb{R}^{m_2}$ is the control input, and $d_t \in \mathbb{R}^{m_3}$ is the external disturbance input.

Assumption 1: The pair $(\mathcal{A}, \mathcal{B})$ is stabilizable, i.e., all uncontrollable modes are asymptotically stable.

We consider the standard Q-learning algorithm and discuss its properties. Since system identification is not going to be performed to estimate the parameters of systems, we use the following objective function

$$\mathcal{J}(x_t, v_t, d_t) = \sum_{i=t}^{\infty} r(x_i, v_i, d_i) \tag{2}$$

where

$$r(x_i, v_i, d_i) = x_i^T R_x x_i + v_i^T R_v v_i - \gamma^2 d_i^T d_i,$$

for a prescribed fixed value of γ. Matrices R_x and R_v are positive semidefinite (PSD) and positive definite (PD), respectively. In the H_∞ control problem, γ is an upper bound on the desired L_2 gain disturbance attenuation [10]. Note that the formulation we used is similar to min-max LQ in [11] and [12]. In the zero-sum game LQ problem, it is desired to find the optimal control $v_t^\star$ and the worst-case disturbance $d_t^\star$. Note that functions in $L_2\,[0, \infty)$ represent the signals having finite energy over infinite interval $[0, \infty)$. That is, $\sum_{t=0}^{\infty} d_t^T d_t < \infty$. Moreover, using (2) and given some fixed policy for an admissible control policy $v_t = K_v x_t$ and a disturbance policy $d_t = K_d x_t$ the value function is defined as

$$V(x_t, K_v, K_d) = \sum_{i=t}^{\infty} r(x_i, K_v x_i, K_d x_i), \tag{3}$$

Since $V(x_t, K_v, K_d) = Q(x_t, K_v x_t, K_d d_t)$, the Bellman equation under the policy gains K_v and K_d can be rewritten as follows:

$$Q(x_t, v_t, d_t) = r(x_t, v_t, d_t) + V(x_{t+1}, K_v, K_d), \tag{4}$$

and the Bellman optimality equation for the Q-function under the optimal policy gains $K_v^\star$ and $K_d^\star$ is

$$Q^\star(x_t, v_t, d_t) = r(x_t, v_t, d_t) + Q^\star(x_{t+1}, K_v^\star x_{t+1}, K_v^\star x_{t+1}). \tag{5}$$

A. Derivation of Q-learning Algorithm

We use the Q-function to develop a Q-learning algorithm ([1], [13]) to solve for the DT H_∞ Control Problem using the Bellman equation (4). The learning process starts with an initial Q-function $Q^0(x, v, d) = 0$ in the Q-learning that is not necessarily optimal, and then derives $Q^1(x, v, d)$ by solving Eq. (6) with $i = 0$.

1) Policy evaluation: We evaluate the policy by using Q-function in (6).

$$Q^{i+1}(x_t, v_t, d_t) = r(x_t, v_t, d_t) + Q^i(x_{t+1}, K_v^i x_{t+1}, K_v^i x_{t+1}). \tag{6}$$

2) Policy improvement: The control and disturbance policies will be improved as follows:

$$K_v^{i+1} = \arg\min_{K_v} Q^{i+1}(x_t, v_t, d_t)$$
$$K_d^{i+1} = \arg\max_{K_d} Q^{i+1}(x_t, v_t, d_t).$$

Let $z_t = [x_t^T, v_t^T, d_t^T]^T$ and

$$P^i = \begin{bmatrix} I & K_v^{iT} & K_d^{iT} \end{bmatrix} S^i \begin{bmatrix} I & K_v^{iT} & K_d^{iT} \end{bmatrix}^T.$$

Given a linear system, linear policies, and quadratic cost, we can assume the quality function (Q-function) is quadratic in the state, control, and disturbance so that

$$Q^{i+1}(z_t) = z_t^T S^{i+1} z_t. \tag{7}$$

Applying (7) in (6), the Lyapunov equation yields

$$z_t^T S^{i+1} z_t = r(x_t, v_t, d_t) + x_{t+1}^T P^i x_{t+1}. \tag{8}$$

Replacing the dynamics (1) in (8), we have:

$$\begin{aligned} z_t^T S^{i+1} z_t &= x_t^T R_x x_t + v_t^T R_v v_t - \gamma^2 d_t^T d_t \\ &\quad + (\mathcal{A}x_t + \mathcal{B}v_t + \mathcal{L}d_t)^T P^i (\mathcal{A}x_t + \mathcal{B}v_t + \mathcal{L}d_t) \\ &= \begin{bmatrix} x_t^T & v_t^T & d_t^T \end{bmatrix} \\ &\quad \begin{bmatrix} R_x + \mathcal{A}^T P^i \mathcal{A} & \mathcal{A}^T P^i \mathcal{B} & \mathcal{A}^T P^i \mathcal{L} \\ \mathcal{B}^T P^i \mathcal{A} & R_v + \mathcal{B}^T P^i \mathcal{B} & \mathcal{B}^T P^i \mathcal{L} \\ \mathcal{L}^T P^i \mathcal{A} & \mathcal{L}^T P^i \mathcal{B} & \mathcal{L}^T P^i \mathcal{L} - \gamma^2 I \end{bmatrix} \begin{bmatrix} x_t \\ v_t \\ d_t \end{bmatrix} \\ &= z_t^T \begin{bmatrix} R_x + \mathcal{A}^T P^i \mathcal{A} & \mathcal{A}^T P^i \mathcal{B} & \mathcal{A}^T P^i \mathcal{L} \\ \mathcal{B}^T P^i \mathcal{A} & R_v + \mathcal{B}^T P^i \mathcal{B} & \mathcal{B}^T P^i \mathcal{L} \\ \mathcal{L}^T P^i \mathcal{A} & \mathcal{L}^T P^i \mathcal{B} & \mathcal{L}^T P^i \mathcal{L} - \gamma^2 I \end{bmatrix} z_t. \end{aligned}$$

Let us partition matrix S^{i+1} as

$$S^{i+1} = \begin{bmatrix} S_{xx}^{i+1} & S_{xv}^{i+1} & S_{xd}^{i+1} \\ S_{vx}^{i+1} & S_{vv}^{i+1} & S_{vd}^{i+1} \\ S_{dx}^{i+1} & S_{dv}^{i+1} & S_{dd}^{i+1} \end{bmatrix}. \tag{9}$$

Optimizing $Q^{i+1}(z_t)$ over v_t and d_t results in

$$v_t = -{S_{vv}^{i+1}}^{-1}(S_{vd}^{i+1}d_t + S_{vx}^{i+1}x_t),$$
$$d_t = -{S_{dd}^{i+1}}^{-1}(S_{dv}^{i+1}v_t + S_{dx}^{i+1}x_t).$$

Substituting v_t in d_t and vice versa yields the equations $v_t^{i+1} = K_v^{i+1}x_t$ and $d_t^{i+1} = K_d^{i+1}x_t$ where

$$K_v^{i+1} = \left(S_{vv}^{i+1} - S_{vd}^{i+1}{S_{dd}^{i+1}}^{-1}S_{dv}^{i+1}\right)^{-1} \times \left(S_{vd}^{i+1}{S_{dd}^{i+1}}^{-1}S_{dx}^{i+1} - S_{vx}^{i+1}\right), \quad (10a)$$

$$K_d^{i+1} = \left(S_{dd}^{i+1} - S_{dv}^{i+1}{S_{vv}^{i+1}}^{-1}S_{vd}^{i+1}\right)^{-1} \times \left(S_{dv}^{i+1}{S_{vv}^{i+1}}^{-1}S_{vx}^{i+1} - S_{dx}^{i+1}\right). \quad (10b)$$

Using (7) and applying the above result in (8), the following recursion can be concluded:

$$S^{i+1} = \underbrace{\begin{bmatrix} R_x & 0 & 0 \\ 0 & R_v & 0 \\ 0 & 0 & -\gamma^2 I \end{bmatrix}}_{G} + \begin{bmatrix} \mathcal{A}^T \\ \mathcal{B}^T \\ \mathcal{L}^T \end{bmatrix} \begin{bmatrix} I & {K_v^i}^T & {K_d^i}^T \end{bmatrix} S^i \begin{bmatrix} I \\ K_v^i \\ K_d^i \end{bmatrix} \begin{bmatrix} \mathcal{A} & \mathcal{B} & \mathcal{L} \end{bmatrix}. \quad (11)$$

Given

$$P^i = \begin{bmatrix} I & {K_v^i}^T & {K_d^i}^T \end{bmatrix} S^i \begin{bmatrix} I & {K_v^i}^T & {K_d^i}^T \end{bmatrix}^T,$$

the following equation can be concluded:

$$P^{i+1} = \begin{bmatrix} I & {K_v^{i+1}}^T & {K_d^{i+1}}^T \end{bmatrix} S^{i+1} \begin{bmatrix} I & {K_v^{i+1}}^T & {K_d^{i+1}}^T \end{bmatrix}^T.$$

Substituting (11), (10a), and (10b), one can obtain:

$$P^{i+1} = R_x + \mathcal{A}^T P^i \mathcal{A} - \begin{bmatrix} \mathcal{A}^T P^i \mathcal{B} & \mathcal{A}^T P^i \mathcal{L} \end{bmatrix} \begin{bmatrix} R_v + B^T P^i \mathcal{B} & \mathcal{B}^T P^i \mathcal{L} \\ \mathcal{L}^T P^i \mathcal{B} & \mathcal{L}^T P^i \mathcal{L} - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} \mathcal{B}^T P^i \mathcal{A} \\ \mathcal{L}^T P^i \mathcal{A} \end{bmatrix}. \quad (12)$$

Equation (12) is called Lyapunov Recursion.

In summary, we evaluate the policy gains K_v and K_d by finding the quadratic kernel S of the Q-function using (11) and then improved policy gains are given by (10a) and (10b).

III. Online Implementation of the Proposed Algorithm

In this section, we discuss the online implementation of the proposed algorithm and prove its convergence. Algorithm 1 summarizes the steps of the proposed algorithm for the H_∞ problem (1).

We will parameterize the Q-function in (6) so that we can separate the unknown matrix S. Using parameterization and defining $s = \text{vecs}(S)$, $p = \text{vecs}(P)$, $z_t = [x_t^T, v_t^T, d_t^T]^T$, $\phi_t(K_v^i) = [x_t^T, (K_v^i x_t)^T, d_t^T]^T$, and $\phi_t(K_v^i, K_d^i) = [x_t^T, (K_v^i x_t)^T, (K_d^i x_t)^T]^T$, we have the below equation:

$$\text{vecv}(z_t)s_{i+1} = r(x_t, v_t, d_t) + \text{vecv}(\phi_{t+1}(K_v^i, K_d^i))s_i. \quad (13)$$

Algorithm 1

1: **Initialization:** $i = 0$, Any **arbitrary** policy gain K_v^0, K_d^0, and $S = \mathbf{0}$
2: **for** $\tau = -(q-1), \cdots, 0$ **do**
3: Sample $\lambda \sim \mathcal{N}(0, W_\lambda)$ and set $v = K_v^0 x + \lambda$.
4: Take v and d and observe x_+.
5: **end for**
6: Estimate S^1 by (14)
7: Improve the policies K_v^1 and K_d^1 by (10a) and (10b).
8: **while** $\|S^{i+1} - S^i\|_2 > \epsilon$ **do**
9: Take K_v^i and observe x_{i+1}.
10: Estimate S^{i+1} by (14).
11: Improve the policies by (10a) and (10b).
12: $i = i + 1$.
13: **end while**

To find the optimal policy in each iteration, we need to solve the following least square (LS) problem:

$$s_{i+1} = \min_s \quad \|\Psi_i s - \Gamma_i\|_2^2, \quad (14)$$

where:
$\xi = \text{vecs}(G)$.
$\Gamma_i = \Psi_i \xi + \Phi_i s_i = \left[\Gamma_{i-1}^T, \gamma_i^T\right]^T$ where
$\gamma_i = \text{vecv}(\phi_i(K_v^i))\xi + \text{vecv}(\phi_{i+1}(K_v^i, K_d^i))s_i$.
$\Phi_i s_i$ can be written as $X_i^+ p_i$ where
$X_i^+ = \left[{X_{i-1}^+}^T, \text{vecv}(x_{i+1})^T\right]^T$.
$\Psi_i = \left[{\Psi_{i-1}}^T, \text{vecv}(\phi_i(K_v^i))^T\right]^T$ and
$\Phi_i = \left[\Phi_{i-1}^T, \text{vecv}(\phi_{i+1}(K_v^i, K_d^i))^T\right]^T$, for $i = 1, 2, \cdots$.
The initial values are given as
$\Psi_0 = \left[\text{vecv}(z_{-(q-1)})^T, \text{vecv}(z_{-(q-2)})^T, \cdots, \text{vecv}(z_0)^T\right]^T$,
$\Phi_0 = \begin{bmatrix} \text{vecv}(\phi_{-(q-2)}(K_v^0, K_d^0)) \\ \cdots \\ \text{vecv}(\phi_1(K_v^0, K_d^0)) \end{bmatrix}$, $\Gamma_0 = \Psi_0 \xi + \Phi_0 s_0$, and
$X_0^+ = \left[\text{vecv}(x_{-(q-2)}))^T, \quad \cdots \quad, \text{vecv}(x_1)^T\right]^T$.

Equation (13) is used in the policy evaluation step to solve for the unknown vector s in the least-squares sense by collecting $q \geq \underline{q}$ data samples of x, v, and d, where $\underline{q} = (m_1 + m_2 + m_3)(m_1 + m_2 + m_3 + 1)/2$. It should be noted that v_t and d_t are linearly dependent on x_t which means that $\Psi^T\Psi$ is not invertible. To resolve this issue, excitation noise is added in v_t and d_t in only the first iteration such that a unique solution to (14) is guaranteed. On the other hand, $\text{rank}(\Psi) = \underline{q}$. In Algorithm (1), instead of getting q samples in each iteration and updating matrix S, we update the algorithms using only a single data. Another advantage is that persistent excitation is needed only in the initial iteration. Note that we only have one index since we use only a single data in each iteration.

A. Recursive Least Square (RLS)

LS estimation is used when one has an overdetermined system of equations. If data is coming in sequentially, we do not have to recompute everything each time a new data

point comes in. Moreover, we can write our new, updated estimate in terms of our old estimate [14].

Consider Eq. (14). The solution can thus be written as

$$\Psi_i{}^T \Psi_i s_{i+1} = \Psi_i{}^T \Gamma_i. \tag{15}$$

By defining $\Xi_i = \Psi_i^T \Psi_i$, we have

$$\begin{aligned}\Xi_i = \Psi_i^T \Psi_i =& \Psi_{i-1}^T \Psi_{i-1} + \text{vecv}(\phi_i(K_v^i))^T \text{vecv}(\phi_i(K_v^i)) \\ =& \Xi_{i-1} + \text{vecv}(\phi_i(K_v^i))^T \text{vecv}(\phi_i(K_v^i)).\end{aligned} \tag{16}$$

Rearranging Eq. (15), we get

$$\begin{aligned}\Xi_i s_{i+1} =& \Psi_{i-1}^T \Gamma_{i-1} + \text{vecv}(\phi_i(K_v^i))^T \gamma_i \\ =& \Xi_{i-1} s_i + \text{vecv}(\phi_i(K_v^i))^T \gamma_i.\end{aligned}$$

By denoting $M_i = \Xi_i^{-1}$,

$$s_{i+1} = M_i \left(\Xi_{i-1} s_i + \text{vecv}(\phi_i(K_v^i))^T \gamma_i \right).$$

Plug the above equation into (16), it yields

$$\begin{aligned}s_{i+1} &= s_i - M_i \\ &\times \left(\text{vecv}(\phi_i(K_v^i))^T \text{vecv}(\phi_i(K_v^i)) s_i - \text{vecv}(\phi_i(K_v^i))^T \gamma_i \right) \\ &= s_i + M_i \text{vecv}(\phi_i(K_v^i))^T \left(\gamma_i - \text{vecv}(\phi_i(K_v^i)) s_i \right),\end{aligned}$$

where M_i can be updated in each iteration using Sherman-Morrison formula ([15]) as follows:

$$M_i = M_{i-1} - \frac{M_{i-1} \text{vecv}(\phi_i(K_v^i))^T \text{vecv}(\phi_i(K_v^i)) M_{i-1}}{1 + \text{vecv}(\phi_i(K_v^i)) M_{i-1} \text{vecv}(\phi_i(K_v^i))^T}. \tag{17}$$

The quantity $M_i \text{vecv}(\phi_i(K_v^i))^T$ is called the "Kalman Filter Gain", and $\gamma_i - \text{vecv}(\phi_i(K_v^i)) s_i$ is called 'innovations' since it compares the difference between a data update and the action given the last estimate. If the dimension of Ξ_i is very large, computation of its inverse can be computationally expensive, so one would like to have a recursion for the M_{i+1} as in (17).

Theorem 1 (Convergence of Algorithm 1): Assume that the linear quadratic problem (1)-(3) is solvable and has a value under the state feedback information structure or equivalently assume there exists a solution to the game's algebraic Riccati recursion (12). Then, iterating on (11) (equivalent to iterating on (12)) with $S^0 = 0$, $K_v^0 = 0$, and $K_d^0 = 0$ converges with $S^i \to S^\star$ and equivalently $P^i \to P^\star$ where the matrix $P^\star$ satisfies the following Recatti equation:

$$\begin{aligned}P^\star = R_x + \mathcal{A}^T P^\star \mathcal{A} - \begin{bmatrix}\mathcal{A}^T P^\star \mathcal{B} & \mathcal{A}^T P^\star \mathcal{L}\end{bmatrix} \\ \begin{bmatrix}R_v + B^T P^\star \mathcal{B} & \mathcal{B}^T P^\star \mathcal{L} \\ \mathcal{L}^T P^\star \mathcal{B} & \mathcal{L}^T P^\star \mathcal{L} - \gamma^2 I\end{bmatrix}^{-1} \begin{bmatrix}\mathcal{B}^T P^\star \mathcal{A} \\ \mathcal{L}^T P^\star \mathcal{A}\end{bmatrix}.\end{aligned} \tag{18}$$

Proof: The proof can be found in [16]. The key idea of the proof is to show Algorithm 1 follows Equation (12) in each iteration, and then by using *Lemma 4.1 and Theorem 4.2* in [17], it is shown that iterating on (12) with $P_0 = 0$ converges to $P^\star$. ■

B. Computational Complexity Analysis

Recall $\underline{q} = (m_1 + m_2 + m_3)(m_1 + m_2 + m_3 + 1)/2$ as the number of parameters to be estimated. In both classical Q-learning and the proposed algorithm, the number of parameters being estimated is similar. For the sake of comparison, assume $q = \underline{q}$. for the initial iteration both of the algorithms have a computational complexity of order $\mathcal{O}(\underline{q}^3)$ while in the rest of the iterations, Algorithm 1 has a computational complexity of order $\mathcal{O}(\underline{q}^2)$, unlike classical Q-learning that has $\mathcal{O}(\underline{q}^3)$ order of computational complexity. In [7], [9]), to update the parameters of the critic network, at least $\underline{q}$ data is required, and because there is a batch of data in each iteration, a pseudo-inverse (with the computational complexity of $\mathcal{O}(\underline{q}^3)$) in each iteration must be computed. In contrast, we emphasize sample complexity and use only a *single data* to update the parameters of the critic network. It is a huge advantage for systems that have long time steps or when acquiring data is not trivial. On the order of computational complexity, using the key equation $s_{i+1} = s_i + M_i \text{vecv}(\phi_i(K_v^i))^T \left(\gamma_i - \text{vecv}(\phi_i(K_v^i)) s_i \right)$, the computational complexity of $M_{i_{\underline{q}\times\underline{q}}} \text{vecv}(\phi_i(K_v^i))_{\underline{q}\times 1}^T$ is $\mathcal{O}(\underline{q}^2)$ (considering $\gamma_i - \text{vecv}(\phi_i(K_v^i)) s_i$ is a scalar). The computational complexity of the key equation reduces to the computational complexity of calculating M_i in (17). The computational complexity of calculating the column vector $M_{i-1_{\underline{q}\times\underline{q}}} \text{vecv}(\phi_i(K_v^i))_{\underline{q}\times 1}^T$ and the row vector $\text{vecv}(\phi_i(K_v^i))_{1\times\underline{q}} M_{i-1_{\underline{q}\times\underline{q}}}$ are $\mathcal{O}(\underline{q}^2)$. Considering the computational complexity of the scalar $\text{vecv}(\phi_i(K_v^i))_{1\times\underline{q}} M_{i-1_{\underline{q}\times\underline{q}}} \text{vecv}(\phi_i(K_v^i))_{\underline{q}\times 1}^T$ is $\mathcal{O}(\underline{q}^2)$, therefore, the computational complexity of calculating M_i is $\mathcal{O}(\underline{q}^2)$, and consequently, the computational complexity of calculating s_{i+1} is $\mathcal{O}(\underline{q}^2)$.

Remark 1: In section III, we only have one index, i, since in each iteration we use only a single data. Therefore, we do not require the use of both subscript i and superscript t and only use index i.

IV. Autonomous Mobility-on-Demand (AMoD) Model

In this section, a discrete-time linear dynamic model is formulated for the AMoD system. We relax the model in [18] by considering origin-destination demand. The linear discrete-time time-delay dynamic system is as follows:

$$w^{rs}(t+1) = w^{rs}(t) + d^{rs}(t) - U^{rs}(t) \tag{19a}$$

$$p_r(t+1) = p_r(t) - \sum_{s\in N} \left(U^{rs}(t) + R^{rs}(t) \right) + \sum_{q\in N} \left(\frac{g^{qr}(t)}{T_{qr}} \right) \tag{19b}$$

$$g^{rs}(t+1) = \left(1 - \frac{1}{T_{rs}} \right) g^{rs}(t) + U^{rs}(t) + R^{rs}(t), \tag{19c}$$

for $\forall\, r, s \in N$ where state variable w^{rs} denotes the waiting customers at r aiming to go to s. State variable p_r characterizes the waiting or available vehicles at station r. State

variable g^{rs} denotes vehicles moving along the link $\{r,s\}$, including both customer-carrying and rebalancing vehicles. Control input U^{rs} is the number of available vehicles at station r with a customer that will be dispatched to link $\{r,s\}$. R^{rs} is the number of available vehicles at station r that will be dispatched to link $\{r,s\}$ for rebalancing. The term $d^{rs}(t)$ represents the arrival of customers in a time step given by the realization of a Poisson process of parameter λ^{rs}. Model (19) is derived using a first-order lag approximation of the time delays. It is assumed that the number of vehicles exiting a link is proportional to the number of vehicles on that link. In other word, at each time instant t, the quantity $g^{rs}(t)/T_{rs}$ leaves the link $\{r,s\}$. Therefore, $U^{rs}(t-T_{rs})+R^{rs}(t-T_{rs})$ can be replaced by $g^{rs}(t)/T_{rs}$.

This AMoD system is subject to some constraints that enforce the non-negativity of state and control input variables. The global system associated with graph G is represented as

$$x_{t+1} = \mathcal{A}x_t + \mathcal{B}v_t + \mathcal{L}d_t, \tag{20}$$

where the vector of all state variables $x_t \in \mathbb{R}^{2n^2-n}$ is $\left[w(t)^T, p(t)^T, g(t)^T\right]^T$ and the vector of all control input variables $v \in \mathbb{R}^{2n(n-1)}$ is defined as $v_t = \left[U(t)^T, R(t)^T\right]^T$. $d_t \in \mathbb{R}^{n(n-1)}$ represents arriving customers. Matrices $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{L}$ can be written as below:

$$\mathcal{A} = \begin{bmatrix} I_{n(n-1)} & 0 & 0 \\ 0 & I_n & E_{\text{in}}\tilde{T}^{-1} \\ 0 & 0 & I_{n(n-1)} - \tilde{T}^{-1} \end{bmatrix}$$
$$\mathcal{B} = \begin{bmatrix} -I_{n(n-1)} & 0 \\ -E_{\text{out}} & -E_{\text{out}} \\ I_{n(n-1)} & I_{n(n-1)} \end{bmatrix}, \quad \mathcal{L} = \begin{bmatrix} I_{n(n-1)} \\ 0 \\ 0 \end{bmatrix}. \tag{21}$$

where E_{in} and $E_{\text{out}} \in \{0,1\}^{n\times m}$ are the in-neighbors and out-neighbors matrices. If graph G is strongly connected and $d^{rs} = \lambda^{rs}$ for $\forall\{r,s\} \in \hat{A}$, where λ^{rs} represents the Poisson arrival rate for the link $\{r,s\}$, then equilibrium points of system (20) are given by $\bar{x} = (\bar{w}, \bar{p}, \bar{g})$, where $\bar{w}$ and $\bar{p}$ can be any arbitrary positive vector, $\bar{g} = \tilde{T}\left(\lambda + \bar{R}\right)$, $\bar{U} = \lambda$, and $\bar{R}$ satisfies $E\left(\bar{R} + \lambda\right) = 0$. If the number of nodes, n, is greater than 2, there will be an infinite number of equilibrium points. Also, the desired equilibrium point that minimizes the number of rebalancing, $\bar{R}^\star$, can be found by solving the following optimization problem:

$$\min_{\bar{R}} \quad \left\|\tilde{T}^{\frac{1}{2}}\bar{R}\right\|_2^2 \tag{22a}$$
$$\text{s.t.} \quad E\left(\bar{R} + \lambda\right) = 0, \quad \bar{R} \geq 0. \tag{22b}$$

where $E = E_{\text{in}} - E_{\text{out}}$ is the incidence matrix. By changing the coordinates of (20), we aim to regulate the AMoD system around the desired equilibrium points.

V. Simulation Study

We first introduce a network for the test we perform. Then, we apply Algorithm 1 developed in Section III to obtain optimal control, disturbance actions, and the value function parameters in time.

A. Studied Network

The University of Minnesota-Twin Cities (UMN) campus network is considered as the site on which to perform the test. The network we consider is partitioned into six zones. Consequently, a digraph with $n = 6$ vertices and $m = 30$ links is produced by partitioning.

B. Case Study

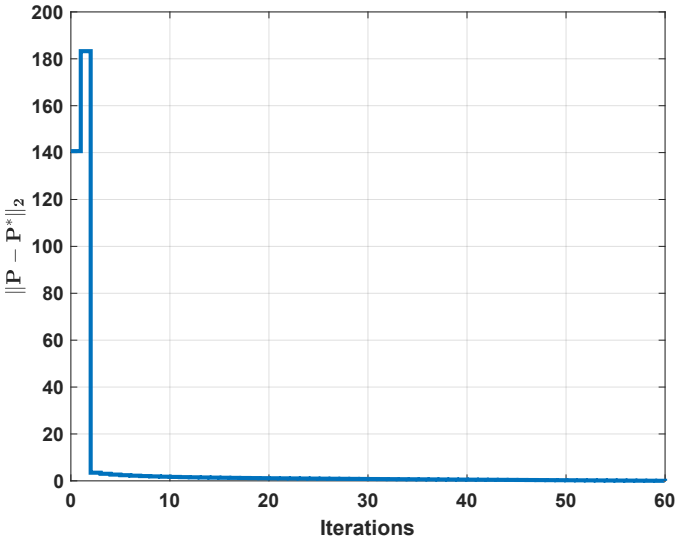
A 12-hour historical trip dataset is considered for the case study. We consider each time step equal to two minutes. So, the number of iterations is 360. Some origin-destination pairs are used more frequently than others, which implies a significant imbalance in demand. The number of vehicles is constant at each time step (including equilibrium) and is equal to $\mathbf{1}_n^T p(t) + \mathbf{1}_{n(n-1)}^T g(t)$ ([19], [18]). Therefore, $\underline{M} = \mathbf{1}_{n(n-1)}^T \bar{g} = T^T\left(\lambda + \bar{R}\right)$ can be considered a lower bound for the fleet size. Initial conditions for the AMoD model are $x_0 = \left[0_{\frac{n(n-1)}{2}}^T \quad \frac{M}{n}1_n^T \quad 0_{\frac{n(n-1)}{2}}^T\right]^T$. The average queue length, the average number of rebalancing vehicles, and the average number of customer-carrying vehicles are the metrics that we are interested in investigating using Algorithm 1. The disturbance attenuation γ is selected to be 0.1. Let $W_\lambda = 0.01I$. Weights matrices R_x and R_v are chosen as $R_x = \begin{bmatrix} \tilde{\lambda} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ and $R_v = \begin{bmatrix} \rho\tilde{T} & 0 \\ 0 & \rho\tilde{T} \end{bmatrix}$, where $\rho = 0.05$. The reference being tracked (λ, $\bar{R}^\star$) is recomputed via Problem (22) every 2 hours (60 iterations). Therefore, R_x will change every 60 iterations. The recursive least-squares algorithm is used to tune the parameters of the critic network online. The parameters of the action networks are updated according to (10a) and (10b).

The parameters of the critic and the action networks are initialized to identity and zero, respectively. Based on this initialization step, the system dynamics move forward in time, and tuning the parameter structures is done by observing the states online. In the RLS problems, the persistency of the excitation condition required to converge the recursive least-squares tuning, i.e., avoiding the parameter drift problem, will hold. However, In Algorithm 1, the persistency of the excitation condition is only required for the initial iteration.

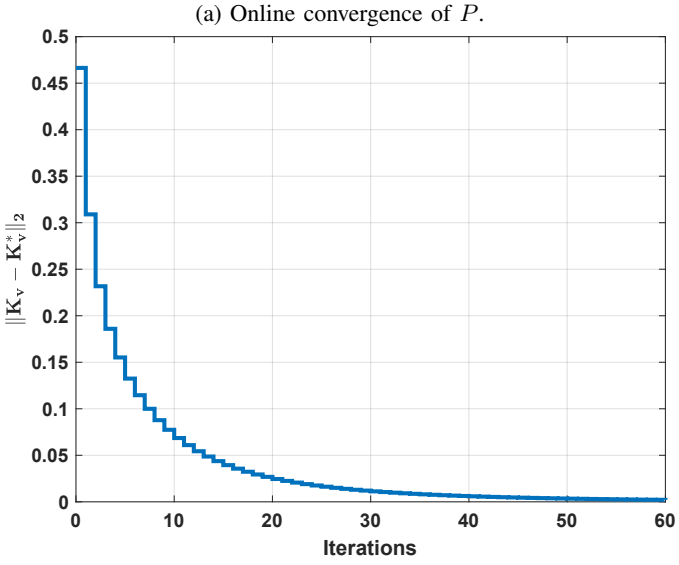
In Fig. 1a, the convergence of the critic network is illustrated. Fig. 1b shows the convergence of the control action network, while Fig. 1c depicts the convergence of the disturbance action network.

VI. Conclusion

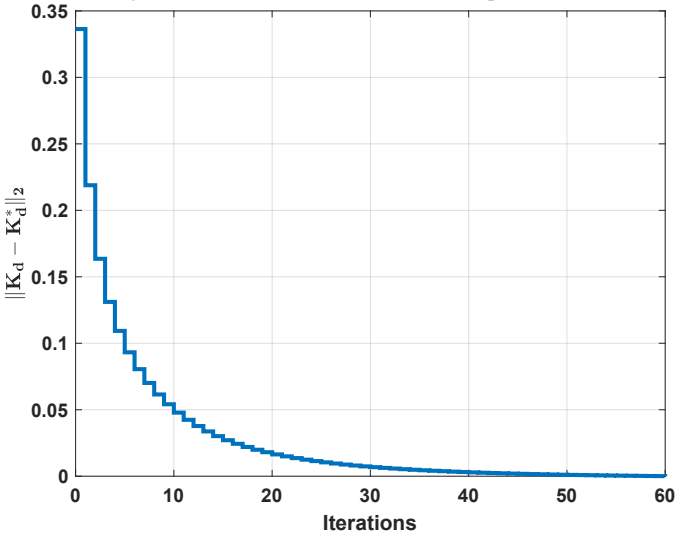
In this paper, we proposed a model-free, real-time, data-efficient Q-learning-based algorithm to solve the H_∞ control of linear discrete-time systems and applied it to an AMoD system modeled as an H_∞ control of the linear discrete-time system. The convergence of the algorithm was proved and it was shown that the parameters of the actions and critic networks converged to the optimal values. Numerical results from an AMoD system control in a real case study showed



(a) Online convergence of P.



(b) Convergence of the control action network parameters (K_v).



(c) Convergence of the disturbance action network parameters (K_d).

Fig. 1: Convergence of the parameters of actions and critic network.

that the proposed algorithm can be implemented in high-dimension systems thanks to the quadratic computational complexity $\mathcal{O}(q^2)$ and using only a single data point for updating the actor and critic networks in each iteration.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Y. Abbasi-Yadkori, N. Lazic, and C. Szepesvári, "Model-free linear quadratic control via reduction to expert prediction," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 3108–3117.

[3] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, "On the sample complexity of the linear quadratic regulator," *Foundations of Computational Mathematics*, vol. 20, no. 4, pp. 633–679, 2020.

[4] S. Tu and B. Recht, "Least-squares temporal difference learning for the linear quadratic regulator," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5005–5014.

[5] N. Matni, A. Proutiere, A. Rantzer, and S. Tu, "From self-tuning regulators to reinforcement learning and back again," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 3724–3740.

[6] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for the linear quadratic regulator," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1467–1476.

[7] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Model-free q-learning designs for linear discrete-time zero-sum games with application to h-infinity control," *Automatica*, vol. 43, no. 3, pp. 473–481, 2007.

[8] H.-N. Wu and B. Luo, "Neural network based online simultaneous policy update algorithm for solving the hji equation in nonlinear h_∞ control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 12, pp. 1884–1895, 2012.

[9] B. Kiumarsi, F. L. Lewis, and Z.-P. Jiang, "H_∞ control of linear discrete-time systems: Off-policy reinforcement learning," *Automatica*, vol. 78, pp. 144–152, 2017.

[10] T. Baar and P. Bernhard, "If'-optimal control and related minimax design problems," *Birkh/iuser;*, 1995.

[11] A. Rantzer, "Minimax adaptive control for a finite set of linear systems," in *Learning for Dynamics and Control*. PMLR, 2021, pp. 893–904.

[12] K. Zhang, Z. Yang, and T. Basar, "Policy optimization provably converges to nash equilibria in zero-sum linear quadratic games," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Systems Magazine*, vol. 32, no. 6, pp. 76–105, 2012.

[14] A. Goel, A. L. Bruce, and D. S. Bernstein, "Recursive least squares with variable-direction forgetting: Compensating for the loss of persistency [lecture notes]," *IEEE Control Systems Magazine*, vol. 40, no. 4, pp. 80–102, 2020.

[15] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *The Annals of Mathematical Statistics*, vol. 21, no. 1, pp. 124–127, 1950.

[16] A. Aalipour and A. Khani, "Data-driven h-infinity control with a real-time and efficient reinforcement learning algorithm: An application to autonomous mobility-on-demand systems," `arXiv:eess.SY/5117049`, 2023.

[17] A. A. Stoorvogel and A. J. Weeren, "The discrete-time riccati equation related to the h/sub/spl infin//control problem," *IEEE Transactions on Automatic Control*, vol. 39, no. 3, pp. 686–691, 1994.

[18] A. Carron, F. Seccamonte, C. Ruch, E. Frazzoli, and M. N. Zeilinger, "Scalable model predictive control for autonomous mobility-on-demand systems," *IEEE Transactions on Control Systems Technology*, 2019.

[19] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus, "Robotic load balancing for mobility-on-demand systems," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, 2012.