Learning-aided Stochastic Network Optimization with Imperfect State Prediction

Longbo Huang
IIIS@Tsinghua University
longbohuang@tsinghua.edu.cn

Minghua Chen
IE@CUHK
minghua@ie.cuhk.edu.hk

Yunxin Liu
Microsoft Research Asia
yunxin.liu@microsoft.com

ABSTRACT

We investigate the problem of stochastic network optimization in the presence of imperfect state prediction and non-stationarity. Based on a novel distribution-accuracy curve prediction model, we develop the *predictive learning-aided control* (PLC) algorithm, which jointly utilizes historic and predicted network state information for decision making. PLC is an online algorithm that requires *zero* a-prior system statistical information, and consists of three key components, namely sequential distribution estimation and change detection, dual learning, and online queue-based control.

Specifically, we show that PLC simultaneously achieves good long-term performance, short-term queue size reduction, accurate change detection, and fast algorithm convergence. In particular, for stationary networks, PLC achieves a near-optimal $[O(\epsilon), O(\log(1/\epsilon)^2)]$ utility-delay tradeoff. For non-stationary networks, PLC obtains an $[O(\epsilon), O(\log^2(1/\epsilon) + \min(\epsilon^{c/2-1}, e_w/\epsilon))]$ utility-backlog tradeoff for distributions that last $\Theta(\frac{\max(\epsilon^{-c}, e_w^{-2})}{\epsilon^{1+a}})$ time, where e_w is the prediction accuracy and $a = \Theta(1) > 0$ is a constant (the Backpressue algorithm [1] requires an $O(\epsilon^{-2})$ length for the same utility performance with a larger backlog). Moreover, PLC detects distribution change $O(w)$ slots faster with high probability (w is the prediction size) and achieves an $O(\min(\epsilon^{-1+c/2}, e_w/\epsilon) + \log^2(1/\epsilon))$ convergence time, which is faster than Backpressure and other algorithms. Our results demonstrate that state prediction (even imperfect) can help (i) achieve faster detection and convergence, and (ii) obtain better utility-delay tradeoffs. They also quantify the benefits of prediction in four important performance metrics, i.e., utility (efficiency), delay (quality-of-service), detection (robustness), and convergence (adaptability), and provide new insight for joint prediction, learning and optimization in stochastic networks.

CCS CONCEPTS

•Networks →Network resources allocation;

ACM Reference format:
Longbo Huang, Minghua Chen, and Yunxin Liu. 2017. Learning-aided Stochastic Network Optimization with Imperfect State Prediction. In *Proceedings of Mobihoc '17, Chennai, India, July 10-14, 2017,* 10 pages.
DOI: http://dx.doi.org/10.1145/3084041.3084054

Mobihoc '17, Chennai, India

DOI: http://dx.doi.org/10.1145/3084041.3084054

1 INTRODUCTION

Enabled by recent developments in sensing, monitoring, and machine learning methods, utilizing prediction for performance improvement in networked systems has received a growing attention in both industry and research. For instance, recent research works [2], [3], and [4] investigate the benefits of utilizing prediction in energy saving, job migration in cloud computing, and video streaming in cellular networks. On the industry side, various companies have implemented different ways to take advantage of prediction, e.g., Amazon utilizes prediction for better package delivery [5] and Facebook enables prefetching for faster webpage loading [6]. However, despite the continuing success in these attempts, most existing results in network control and analysis do not investigate the impact of prediction. Therefore, we still lack a thorough theoretical understanding about the *value-of-prediction* in stochastic network control. Fundamental questions regarding how prediction should be integrated in network algorithms, the ultimate prediction gains, and how prediction error impacts performance, remain largely unanswered.

To contribute to developing a theoretical foundation for utilizing prediction in networks, in this paper, we consider a general constrained stochastic network optimization formulation, and aim to rigorously quantify the benefits of system state prediction and the impact of prediction error. Specifically, we are given a discrete-time stochastic network with a dynamic state that evolves according to some potentially non-stationary probability law. Under each system state, a control action is chosen and implemented. The action generates traffic into network queues but also serves workload from them. The action also results in a system utility (cost) due to service completion (resource expenditure). The traffic, service, and cost are jointly determined by the action and the system state. The objective is to maximize the expected utility (or equivalently, minimize the cost) subject to traffic/service constraints, given imperfect system state prediction information.

This is a general framework that models various practical scenarios, for instance, mobile networks, computer networks, supply chains, and smart grids. However, understanding the impact of prediction in this framework is challenging. First, statistical information of network dynamics is often unknown a-priori. Hence, in order to achieve good performance, algorithms must be able to quickly learn certain sufficient statistics of the dynamics, and make efficient use of prediction while carefully handling prediction error. Second, system states appear randomly in every time slot. Thus, algorithms must perform well under such incremental realizations of the randomness. Third, quantifying system service quality often involves handling queueing in the system. As a result, explicit connections between control actions and queues must be established.

There has been a recent effort in developing algorithms that can achieve good utility and delay performance for this general problem without prediction in various settings, for instance, wireless networks, [7], [8], [9], [10], processing networks, [11], [12], cognitive radio, [13], and the smart grid, [14], [15]. However, existing results mostly focus on networks with stationary distributions. They either assume full system statistical information beforehand, or rely on stochastic approximation techniques to avoid the need of such information. Works [16] and [17] propose schemes to incorporate historic system information into control, but they do not consider prediction. Recent results in [18], [19], [20], [21] and [22] consider problems with traffic demand prediction, and [23] jointly considers demand and channel prediction. However, they focus either on $M/M/1$-type models, or do not consider queueing, or do not consider the impact of prediction error. In a different line of work, [24], [25], [26] and [27] investigate the benefit of prediction from the online algorithm design perspective. Although the results provide novel understanding about the effect of prediction, they do not apply to the general constrained network optimization problems, where action outcomes are general functions of time-varying network states, queues evolve in a controlled manner, i.e., arrival and departure rates depend on the control policy, and prediction can contain error.

In this paper, we develop a novel control algorithm for the general framework called *predictive learning-aided control* (PLC). PLC is an online algorithm that consists of three components, sequential distribution estimation and change detection, dual learning, and online control (see Fig. 1).

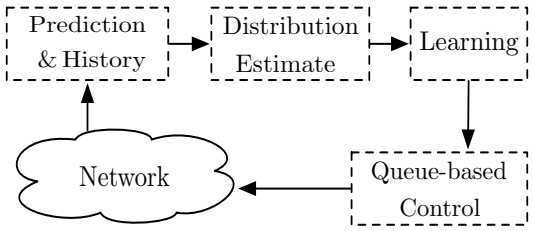

Figure 1: The PLC algorithm contains (i) a distribution estimator that utilizes historic and predicted information to simultaneously form a distribution estimate and detect distribution change, (ii) a learning component that computes an empirical Lagrange multiplier based on the distribution estimate, and (iii) a queue-based controller whose decision-making information is augmented with the multiplier.

The distribution estimator conducts sequential statistical comparisons based on prediction and historic network state records. Doing so efficiently detects changes of the underlying probability distribution and guides us in selecting the right state samples to form distribution estimates. The estimated distribution is then fed into the dual learning component to compute an empirical multiplier of an underlying optimization formulation. This multiplier is further incorporated into the Backpressure (BP) network controller [1] to perform realtime network operation. Compared to the commonly adopted receding-horizon-control approach (RHC), e.g., [28], PLC provides another way to utilize future state information, which focuses on using the predicted distribution for guiding action selection in the present slot and can be viewed as performing steady-state control under the predicted future distribution.

We summarize our main contributions as follows.

i. We propose a general state prediction model featured with a distribution-accuracy curve. Our model captures key factors of several existing prediction models, including window-based [22], distribution-based [29], and filter-based [26] models.

ii. We propose a general constrained network control algorithm called *predictive learning-aided control* (PLC), which is an online algorithm that requires *zero* a-prior system statistical information. PLC jointly performs sequential distribution estimation and change detection, dual learning, and queue-based online control.

iii. We show that for stationary networks, PLC achieves an $[O(\epsilon), O(\log^2(1/\epsilon))]$ utility-delay tradeoff. For non-stationary networks, PLC obtains an $[O(\epsilon), O(\log^2(1/\epsilon) + \min(\epsilon^{c/2-1}, e_w/\epsilon))]$ utility-backlog tradeoff for distributions that last $\Theta(\frac{\max(\epsilon^{-c}, e_w^{-2})}{\epsilon^{1+a}})$ time, where e_w is the prediction accuracy, $c \in (0,1)$ and $a > 0$ is an $\Theta(1)$ constant (the Backpressue algorithm [1] requires an $O(\epsilon^{-2})$ length for the same utility performance with a larger backlog).[1]

iv. We show that for both stationary and non-stationary system dynamics, PLC detects distribution change $O(w)$ slots (w is prediction window size) faster with high probability and achieves a fast $O(\min(\epsilon^{-1+c/2}, e_w/\epsilon) + \log^2(1/\epsilon))$ convergence time, which is faster than the $O(\epsilon^{-1+c/2} + \epsilon^{-c})$ time of the OLAC scheme [16], and the $O(1/\epsilon)$ time of Backpressure.

v. Our results show that state prediction (even imperfect) can help performance in two ways (a) achieve faster detection, i.e., detect change w slots faster, and (b) obtain a better utility-delay tradeoff, i.e., reduce delay to $O(e_w/\epsilon + \log^2(1/\epsilon))$ for the same utility. They rigorously quantify the benefits of prediction in four important performance metrics, i.e., utility (efficiency), delay (quality-of-service), detection (robustness), and convergence (adaptability).

The rest of the paper is organized as follows. In Section 2, we discuss a few motivating examples in different application scenarios. We set up the notations in Section 3, and present the problem formulation in Section 4. Background information is provided in Section 5. Then, we present PLC in Section 6, and prove its performance in Section 7. Simulation results are presented in Section 8, followed by conclusions in Section 9. To facilitate reading, all the proofs are placed in the appendices.

2 MOTIVATING EXAMPLES

In this section, we present a few interesting practical scenarios that fall into our general framework.

Matching in sharing platforms: Consider a Uber-like company that provides ride service to customers. At every time, customer requests enter the system and available cars join to provide service. Depending on the environment condition (state), e.g., traffic condition or customer status, matching customers to drivers can result in different user satisfaction, and affect the revenue of the company (utility). The company gets access to future customer demand and car availability, and system condition information (prediction), e.g., through reservation or machine learning tools. The

[1]Note that when there is no prediction, i.e., $w = 0$ and $e_w = \infty$, we recover previous results of OLAC [16].

objective is to optimally match customers to cars so that the utility is maximized, e.g., [30] and [31].

Energy optimization in mobile networks: Consider a base-station (BS) sending traffic to a set of mobile users. The channel conditions (state) between users and the BS are time-varying. Thus, the BS needs different amounts of power for packet transmission (cost) at different times. Due to higher layer application requirements, the BS is required to deliver packets to users at pre-specified rates. On the other hand, the BS can predict future user locations in some short period of time, from which it can estimate future channel conditions (prediction). The objective of the BS is to jointly optimize power allocation and scheduling among users, so as to minimize energy consumption, while meeting the rate requirements, e.g., [8], [13]. Other factors such as energy harvesting, e.g., [32], can also be incorporated in the formulation.

Resource allocation in cloud computing: Consider an operator, e.g., a dispatcher, assigning computing jobs to servers for processing. The job arrival process is time-varying (state), and available processing capacities at servers are also dynamic (state), e.g., due to background processing. Completing users' job requests brings the operator reward (utility). The operator may also have information regarding future job arrivals and service capacities (prediction). The goal is to allocate resources and balance the loads properly, so as to maximize system utility. This example can be extended to capture other factors such as rate scaling [33] and data locality constraints [34].

In these examples and related works, not only can the state statistics be potentially non-stationary, but the system often gets access to certain (possibly imperfect) future state information through various prediction techniques. These features make the problems different from existing settings considered, e.g., [8] and [15], and require different approaches for both algorithm design and analysis.

3 NOTATIONS

$\mathbb{R}^n$ denotes the n-dimensional Euclidean space. $\mathbb{R}^n_+$ ($\mathbb{R}^n_-$) denotes the non-negative (non-positive) orthant. Bold symbols $\boldsymbol{x} = (x_1, ..., x_n)$ denote vectors in $\mathbb{R}^n$. $w.p.1$ denotes "with probability 1." $\|\cdot\|$ denotes the Euclidean norm. For a sequence $\{y(t)\}_{t=0}^{\infty}$, $\overline{y} = \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{y(\tau)\}$ denotes its average (when exists). $\boldsymbol{x} \geq \boldsymbol{y}$ means $x_j \geq y_j$ for all j. For distributions $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$, $\|\boldsymbol{\pi}_1 - \boldsymbol{\pi}_2\|_{tv} = \sum_i |\pi_{1i} - \pi_{2i}|$ denotes the total variation distance.

4 SYSTEM MODEL

Consider a controller that operates a network with the goal of minimizing the time average cost, subject to the queue stability constraint. The network is assumed to operate in slotted time, i.e., $t \in \{0, 1, 2, ...\}$, and there are $r \geq 1$ queues in the network.

4.1 Network state

In every slot t, we use $S(t)$ to denote the current network state, which indicates the current network parameters, such as a vector of conditions for each network link, or a collection of other relevant information about the current network channels and arrivals. $S(t)$ is independently distributed across time, and each realization is drawn from a state space of M distinct states denoted as $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$.[2] We denote $\pi_i(t) = \Pr\{S(t) = s_i\}$ the probability of being in state s_i at time t and denote $\boldsymbol{\pi}(t) = (\pi_1(t), ..., \pi_M(t))$ the state distribution at time t. The network controller can observe $S(t)$ at the beginning of every slot t, but the $\pi_i(t)$ probabilities are unknown. To simplify notations, we divide time into intervals that have the same distributions and denote $\{t_k, k = 0, 1, ...\}$ the starting point of the k-th interval $\mathcal{I}_k$, i.e., $\boldsymbol{\pi}(t) = \boldsymbol{\pi}_k$ for all $t \in \mathcal{I}_k \triangleq \{t_k, t_{k+1} - 1\}$. The length of $\mathcal{I}_k$ is denoted by $d_k \triangleq t_{k+1} - t_k$.

4.2 State prediction

At every time slot, the operator gets access to a prediction module, e.g., a machine learning algorithm, which provides prediction of future network states. Different from recent works, e.g., [25], [26] and [35], which assume prediction models on individual states, we assume that the prediction module outputs a sequence of predicted distributions $\mathcal{W}_w(t) \triangleq \{\hat{\boldsymbol{\pi}}(t), \hat{\boldsymbol{\pi}}(t+1), ..., \hat{\boldsymbol{\pi}}(t+w)\}$, where $w+1$ is the prediction window size. Moreover, the prediction quality is characterized by a distribution-accuracy curve $\{e(0), ..., e(w)\}$ as follows. For every $0 \leq k \leq w$, $\hat{\boldsymbol{\pi}}(t+k)$ satisfies:

$$||\hat{\boldsymbol{\pi}}(t+k) - \boldsymbol{\pi}(t+k)||_{tv} \leq e(k), \ \forall\ k. \tag{1}$$

That is, the predicted distribution at time k has a total-variation error bounded by some $e(k) \geq 0$.[3] Note that $e(k) = 0$ for all $0 \leq k \leq w$ corresponds to a perfect predictor, in that it predicts the exact distribution in every slot. We assume the $\{e(0), ..., e(w)\}$ curve is known to the operator and denote $e_w \triangleq \frac{1}{w+1}\sum_{k=0}^{w} e(k)$ the average prediction error.

Our prediction model (1) is general and captures key characteristics of several existing prediction models. For instance, it captures the exact demand statistics prediction model in [29], where the future demand distribution is known ($e(k) = 0$ for all $0 \leq k \leq w$). It can also capture the window-based predictor model, e.g., [22], if each $\hat{\boldsymbol{\pi}}(t+k)$ corresponds to the indicator value for the true state. Moreover, our model captures the error-convolution prediction model proposed in [35], [25] and [26], which captures features of the Wiener filter and Kalman filter. Specifically, under the convolution model, the predicted state $\hat{S}(t+k)$ at time t satisfies:[4]

$$\|\hat{S}(t+k) - S(t+k)\| = \sum_{s=t+1}^{t+k} g(t+k-s)a(s), \tag{2}$$

where $g(s)$ is the impulse function that captures how error propagates over time in prediction, and $a(s)$ is assumed to be a zero mean i.i.d. random variable [25]. Thus, we can compute the corresponding $e(k)$ once $g(s)$ and $a(s)$ are given.

4.3 The cost, traffic, and service

At each time t, after observing $S(t) = s_i$, the controller chooses an action $x(t)$ from a set $\mathcal{X}_i$, i.e., $x(t) = x_i$ for some $x_i \in \mathcal{X}_i$. The set $\mathcal{X}_i$ is called the feasible action set for network state s_i and is assumed to be time-invariant and compact for all $s_i \in \mathcal{S}$. The cost, traffic, and service generated by the action $x(t) = x_i$ are as follows:

[2] The independent assumption is made to facilitate presentation and understanding. The results in this paper can likely be generalized to systems where $S(t)$ evolves according to general time inhomogeneous Markovian dynamics.

[3] It makes sense to assume a deterministic upper bound of the difference here because we are dealing with distributions.

[4] In [25] and [26], the state space is a metric space.

(a) The chosen action has an associated cost given by the cost function $f(t) = f(s_i, x_i) : \mathcal{X}_i \mapsto \mathbb{R}_+$ (or $\mathcal{X}_i \mapsto \mathbb{R}_-$ in reward maximization problems).[5]
(b) The amount of traffic generated by the action to queue j is determined by the traffic function $A_j(t) = A_j(s_i, x_i) : \mathcal{X}_i \mapsto \mathbb{R}_+$, in units of packets.
(c) The amount of service allocated to queue j is given by the rate function $\mu_j(t) = \mu_j(s_i, x_i) : \mathcal{X}_i \mapsto \mathbb{R}_+$, in units of packets.

Here $A_j(t)$ can include both exogenous arrivals from outside the network to queue j, and endogenous arrivals from other queues, i.e., transmitted packets from other queues to queue j. We assume the functions $-f(s_i, \cdot)$, $\mu_j(s_i, \cdot)$ and $A_j(s_i, \cdot)$ are time-invariant, their magnitudes are uniformly upper bounded by some constant $\delta_{\max} \in (0, \infty)$ for all s_i, j, and they are known to the operator. Note that this formulation is general and models many network problems, e.g., [8], [15], and [36].

4.4 Problem formulation

Let $\boldsymbol{q}(t) = (q_1(t), ..., q_r(t))^T \in \mathbb{R}_+^r$, $t = 0, 1, 2, ...$ be the queue backlog vector process of the network, in units of packets. We assume the following queueing dynamics:

$$q_j(t+1) = \max\Big[q_j(t) - \mu_j(t) + A_j(t), 0\Big], \quad \forall j, \tag{3}$$

and $\boldsymbol{q}(0) = \boldsymbol{0}$. By using (3), we assume that when a queue does not have enough packets to send, null packets are transmitted, so that the number of packets entering $q_j(t)$ is equal to $A_j(t)$. We adopt the following notion of queue stability [1]:

$$\overline{q}_{\text{av}} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{j=1}^{r} \mathbb{E}\Big\{q_j(\tau)\Big\} < \infty. \tag{4}$$

We use Π to denote an action-choosing policy, and use f_{av}^{Π} to denote its time average cost, i.e.,

$$f_{\text{av}}^{\Pi} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\Big\{f^{\Pi}(\tau)\Big\}, \tag{5}$$

where $f^{\Pi}(\tau)$ is the cost incurred at time τ under policy Π. We call an action-choosing policy *feasible* if at every time slot t it only chooses actions from the feasible action set $\mathcal{X}_i$ when $S(t) = s_i$. We then call a feasible action-choosing policy under which (4) holds a *stable* policy.

In every slot, the network controller observes the current network state and prediction, and chooses a control action, with the goal of minimizing the time average cost subject to network stability. This goal can be mathematically stated as:[6]

$$\textbf{(P1)} \ \ \min: \ f_{\text{av}}^{\Pi}, \ \text{s.t. } \textbf{(4)}.$$

In the following, we call **(P1)** *the stochastic problem*, and we use $f_{\text{av}}^{\boldsymbol{\pi}}$ to denote its optimal solution given a fixed distribution $\boldsymbol{\pi}$. It can be seen that the examples in Section 2 can all be modeled by our stochastic problem framework.

Throughout our paper, we make the following assumption.

[5] We use cost and utility interchangeably in this paper.
[6] When $\boldsymbol{\pi}(t)$ is time-varying, the optimal system utility needs to be defined carefully. We will specify it when discussing the corresponding results.

Assumption 1. *For every system distribution $\boldsymbol{\pi}_k$, there exists a constant $\epsilon_k = \Theta(1) > 0$ such that for any valid state distribution $\boldsymbol{\pi}' = (\pi'_1, ..., \pi'_M)$ with $\|\boldsymbol{\pi}' - \boldsymbol{\pi}_k\|_{tv} \le \epsilon_k$, there exist a set of actions $\{x_z^{(s_i)}\}_{i=1,...,M}^{z=1,2,...,\infty}$ with $x_z^{(s_i)} \in \mathcal{X}_i$ and variables $\vartheta_z^{(s_i)} \ge 0$ for all s_i and z with $\sum_z \vartheta_z^{(s_i)} = 1$ for all s_i (possibly depending on $\boldsymbol{\pi}'$), such that:*

$$\sum_{s_i} \pi'_i \Big\{ \sum_z \vartheta_z^{(s_i)} [A_j(s_i, x_z^{(s_i)}) - \mu_j(s_i, x_z^{(s_i)})] \Big\} \le -\eta_0, \ \forall j, \tag{6}$$

where $\eta_0 = \Theta(1) > 0$ is independent of $\boldsymbol{\pi}'$. $\diamond$

Assumption 1 corresponds to the "slack" condition commonly assumed in the literature with $\epsilon_k = 0$, e.g., [36] and [37].[7] With $\epsilon_k > 0$, we assume that when two systems are relatively close to each other (in terms of $\boldsymbol{\pi}$), they can both be stabilized by some (possibly different) randomized control policy that results in the same slack.

4.5 Discussion of the model

Two key differences between our model and previous ones include (i) $\boldsymbol{\pi}(t)$ itself can be time-varying and (ii) the operator gets access to a prediction window $\mathcal{W}_w(t)$ that contains imperfect prediction. These two extensions are important to the current network control literature. First, practical systems are often non-stationary. Thus, system dynamics can have time-varying distributions. Thus, it is important to have efficient algorithms to automatically adapt to the changing environment. Second, prediction has recently been made increasingly accurate in various contexts, e.g., user mobility in cellular network and harvestable energy availability in wireless systems, by data collection and machine learning tools. Thus, it is critical to understand the fundamental benefits and limits of prediction, and its optimal usage.

5 THE DETERMINISTIC PROBLEM

For our algorithm design and analysis, we define the *deterministic problem* and its dual problem [38]. Specifically, the deterministic problem for a given distribution $\boldsymbol{\pi}$ is defined as follows [38]:

$$\begin{aligned} \min: \ & V \sum_{s_i} \pi_i f(s_i, x^{(s_i)}) \\ \text{s.t.} \ & \sum_{s_i} \pi_i [A_j(s_i, x^{(s_i)}) - \mu_j(s_i, x^{(s_i)})] \le 0, \ \forall j, \\ & x^{(s_i)} \in \mathcal{X}_i \quad \forall i = 1, 2, ..., M. \end{aligned} \tag{7}$$

Here the minimization is taken over $\boldsymbol{x} \in \prod_i \mathcal{X}_i$, where $\boldsymbol{x} = (x^{(s_1)}, ..., x^{(s_M)})^T$, and $V \ge 1$ is a positive constant introduced for later analysis. The dual problem of (7) can be obtained as follows:

$$\max: \ \ g(\boldsymbol{\gamma}, \boldsymbol{\pi}), \quad \text{s.t. } \boldsymbol{\gamma} \ge \boldsymbol{0}, \tag{8}$$

where $g(\boldsymbol{\gamma}, \boldsymbol{\pi})$ is the dual function for problem (7) and is defined as:

$$\begin{aligned} g(\boldsymbol{\gamma}, \boldsymbol{\pi}) = \inf_{x^{(s_i)} \in \mathcal{X}_i} \sum_{s_i} \pi_i \Big\{ & V f(s_i, x^{(s_i)}) \\ & + \sum_j \gamma_j \Big[A_j(s_i, x^{(s_i)}) - \mu_j(s_i, x^{(s_i)})\Big] \Big\}. \end{aligned} \tag{9}$$

[7] Note that $\eta_0 \ge 0$ is a necessary condition for network stability [1].

$\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_r)^T$ is the Lagrange multiplier of (7). It is well known that $g(\boldsymbol{\gamma}, \boldsymbol{\pi})$ in (9) is concave in the vector $\boldsymbol{\gamma}$ for all $\boldsymbol{\gamma} \in \mathbb{R}^r$. Hence, the problem (8) can usually be solved efficiently, particularly when the cost functions and rate functions are separable over different network components [39]. We use $\boldsymbol{\gamma}^*_{\boldsymbol{\pi}}$ to denote the optimal multiplier corresponding to a given $\boldsymbol{\pi}$ and sometimes omit the subscript when it is clear. Denote $g^*_{\boldsymbol{\pi}}$ the optimal value of (8) under a fixed distribution $\boldsymbol{\pi}$. It was shown in [40] that:

$$f^{\boldsymbol{\pi}}_{\text{av}} = g^*_{\boldsymbol{\pi}}. \tag{10}$$

That is, $g^*_{\boldsymbol{\pi}}$ characterizes the optimal time average cost of the stochastic problem. For our analysis, we make the following assumption on the $g(\boldsymbol{\gamma}, \boldsymbol{\pi}_k)$ function.

Assumption 2. *For every system distribution $\boldsymbol{\pi}_k$, $g(\boldsymbol{\gamma}, \boldsymbol{\pi}_k)$ has a unique optimal solution $\boldsymbol{\gamma}^*_{\boldsymbol{\pi}_k} \neq \mathbf{0}$ in $\mathbb{R}^r$.* $\diamond$

Assumption 2 is also commonly assumed and holds for many network utility optimization problems, e.g., [8] and [38].

6 PREDICTIVE LEARNING-AIDED CONTROL

In this section, we present the *predictive learning-aided control* algorithm (PLC). PLC contains three main components: a distribution estimator, a learning component, and an online queue-based controller. Below, we first present the estimation part. Then, we present the PLC algorithm.

6.1 Distribution estimation and change detection

Here we specify the distribution estimator. The idea is to first combine the prediction in $\mathcal{W}_w(t)$ with historic state information to form an *average* distribution, and then perform statistical comparisons for change detection. We call the module the *average distribution estimate* (ADE).

Specifically, ADE maintains two windows $\mathcal{W}_m(t)$ and $\mathcal{W}_d(t)$ to store network state samples, i.e.,

$$\mathcal{W}_d(t) = \{b^s_d(t), ..., b^e_d(t)\}, \tag{11}$$

$$\mathcal{W}_m(t) = \{b_m(t), ..., \min[b^s_d(t), b_m(t) + T_l]\}. \tag{12}$$

Here $b^s_d(t)$ and $b_m(t)$ mark the beginning slots of $\mathcal{W}_d(t)$ and $\mathcal{W}_m(t)$, respectively, and $b^e_d(t)$ marks the end of $\mathcal{W}_d(t)$. Ideally, $\mathcal{W}_d(t)$ contains the most recent d samples (including the prediction) and $\mathcal{W}_m(t)$ contains T_l subsequent samples (where T_l is a pre-specified number). We denote $W_m(t) = |\mathcal{W}_m(t)|$ and $W_d(t) = |\mathcal{W}_d(t)|$. Without loss of generality, we assume that $d \geq w + 1$. This is a reasonable assumption, as we see later that d grows with our control parameter V while prediction power is often limited in practice.

We use $\hat{\boldsymbol{\pi}}^d(t)$ and $\hat{\boldsymbol{\pi}}^m(t)$ to denote the empirical distributions of $\mathcal{W}_d(t)$ and $\mathcal{W}_m(t)$, i.e.,[8]

$$\hat{\pi}^d_i(t) = \frac{1}{d}\left(\sum_{\tau=(t+w-d)_+}^{t-1} 1_{[S(\tau)=s_i]} + \sum_{\tau \in \mathcal{W}_w(t)} \hat{\pi}_i(\tau)\right)$$

$$\hat{\pi}^m_i(t) = \frac{1}{W_m(t)} \sum_{\tau \in \mathcal{W}_m(t)} 1_{[S(\tau)=s_i]}.$$

[8]Note that this is only one way to utilize the samples. Other methods such as EWMA can also be applied when appropriate.

That is, $\hat{\boldsymbol{\pi}}^d(t)$ is the average of the empirical distribution of the "observed" samples in $\mathcal{W}_d(t)$ and the predicted distribution, whereas $\hat{\boldsymbol{\pi}}^m(t)$ is the empirical distribution.

The formal procedure of ADE is as follows (parameters T_l, d, ϵ_d will be specified later).

Average Distribution Estimate (ADE(T_l, d, ϵ_d)): Initialize $b^s_d(0) = 0$, $b^e_d(0) = t + w$ and $b_m(0) = 0$, i.e., $\mathcal{W}_d(t) = \{0, ..., t + w\}$ and $\mathcal{W}_m(t) = \phi$. At every time t, update $b^s_d(t)$, $b^e_d(t)$ and $b_m(t)$ as follows:

(i) If $W_m(t) \geq d$ and $||\hat{\boldsymbol{\pi}}^d(t) - \hat{\boldsymbol{\pi}}^m(t)||_{tv} > \epsilon_d$, set $b_m(t) = t + w + 1$ and $b^s_d(t) = b^e_d(t) = t + w + 1$.

(ii) If $W_m(t) = T_l$ and there exists k such that $||\hat{\boldsymbol{\pi}}(t + k) - \hat{\boldsymbol{\pi}}^m(t)||_{tv} > e(k) + \frac{2M\log(T_l)}{\sqrt{T_l}}$, set $b_m(t) = t + w + 1$ and $b^s_d(t) = b^e_d(t) = t + w + 1$. Mark $t + w + 1$ a reset point.

(iii) Else if $t \leq b^s_d(t - 1)$, $b_m(t) = b_m(t - 1)$, $b^s_d(t) = b^s_d(t - 1)$, and $b^e_d(t) = b^e_d(t - 1)$.[9]

(iv) Else set $b_m(t) = b_m(t - 1)$, $b^s_d(t) = (t + w - d)_+$ and $b^e_d(t) = t + w$.

Output an estimate at time t as follow:

$$\boldsymbol{\pi}_a(t) = \begin{cases} \hat{\boldsymbol{\pi}}^m(t) & \text{if } W_m(t) \geq T_l \\ \frac{1}{w+1}\sum_{k=0}^{w} \hat{\boldsymbol{\pi}}(t + k) & \text{else} \end{cases} \quad \diamond \tag{13}$$

The idea of ADE is shown in Fig. 4.

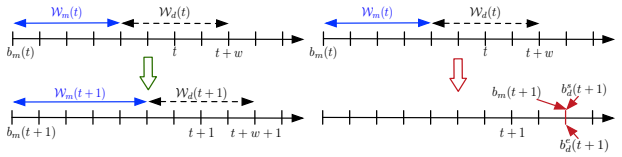

Figure 2: Evolution of $\mathcal{W}_m(t)$ and $\mathcal{W}_d(t)$. (Left) No change detected: $\mathcal{W}_d(t)$ advances by one slot and $\mathcal{W}_m(t)$ increases its size by one. (Right) Change detected: both windows set their start and end points to $t + w + 1$.

The intuition of ADE is that if the environment is changing over time, we should rely on prediction for control. Else if the environment is stationary, then one should use the average distribution learned over time to combat the potential prediction error that may affect performance. T_l is introduced to ensure the accuracy of the empirical distribution and can be regarded as the confidence-level given to the distribution stationarity. A couple of technical remarks are also ready. (a) The term $2M\log(T_l)/\sqrt{T_l}$ is to compensate the inevitable deviation of $\hat{\boldsymbol{\pi}}^m(t)$ from the true value due to randomness. (b) In $\mathcal{W}_m(t)$, we only use the first T_l historic samples. Doing so avoids random oscillation in estimation and facilitates analysis.

Note that prediction is used in two ways in ADE. First, it is used in step (i) to decide whether the empirical distributions match (average prediction). Second, it is used to check whether prediction is consistent with the history (individual prediction). The reason for having this two-way utilization is to accommodate general prediction types. For example, suppose each $\hat{\boldsymbol{\pi}}(t + k)$ denotes the indicator for state $S(t + k)$, e.g., as in the look-ahead window model

[9]This step is evoked after we set $b_m(t') = b^s_d(t') = t' + w + 1 \geq t$ for some time t', in which case we the two windows remain unchanged until t is larger than $t' + w + 1$.

[22]. Then, step (ii) is loose since $e(k)$ is large, but step (i) will be useful. On the other hand, when $\hat{\boldsymbol{\pi}}(t+k)$ gets closer to the true distribution, both steps will be useful.

6.2 Predictive learning-aided control

We are now ready to present the PLC algorithm. Our algorithm is shown in Fig. 1, and the formal description is given below.

Predictive Learning-aided Control (PLC): At time t, do:

- (Estimation) Update $\boldsymbol{\pi}_a(t)$ with $\texttt{ADE}(T_l, d, \epsilon_d)$.
- (Learning) Solve the following empirical problem and compute the optimal Lagrange multiplier $\boldsymbol{\gamma}^*(t)$, i.e.,

$$\max: \quad g(\boldsymbol{\gamma}, \boldsymbol{\pi}_a(t)), \quad \text{s.t. } \boldsymbol{\gamma} \geq \mathbf{0}, \tag{14}$$

  If $\boldsymbol{\gamma}^*(t) = \infty$, set $\boldsymbol{\gamma}^*(t) = V\log(V)\cdot\mathbf{1}$. If $W_m(t-1) = T_l$ and $\boldsymbol{\pi}_a(t) \neq \boldsymbol{\pi}_a(t-1)$, set $\boldsymbol{q}(t+w+1) = \mathbf{0}$.
- (Control) At every time slot t, observe the current network state $S(t)$ and the backlog $\boldsymbol{q}(t)$. If $S(t) = s_i$, choose $x^{(s_i)} \in \mathcal{X}_i$ that solves the following:

$$\max: \quad -Vf(s_i,x) + \sum_{j=1}^{r} Q_j(t)\left[\mu_j(s_i,x) - A_j(s_i,x)\right]$$
$$\text{s.t.} \quad x \in \mathcal{X}_i, \tag{15}$$

  where $Q_j(t) \triangleq q_j(t) + (\gamma_j^*(t) - \theta)^+$. Then, update the queues according to (3) with Last-In-First-Out. ◇

For readers who are familiar with the Backpressure (BP) algorithm, e.g., [1] and [41], the control component of PLC is the BP algorithm with its queue vector augmented by the empirical multiplier $\boldsymbol{\gamma}^*(t)$. Also note that packet dropping is introduced to enable quick adaptation to new dynamics if there is a distribution change. It occurs only when a long-lasting distribution ends, which avoids dropping packets frequently in a fast-changing environment.

We have the following remarks. **(i) Prediction usage:** Prediction is explicitly incorporated into control by forming an average distribution and converting the distribution estimate into a Lagrange multiplier. The intuition for having $T_l = \max(V^c, e_w^{-2})$ is that when e_w is small, we should rely on prediction as much as possible, and only switch to learned statistics when it is sufficiently accurate. **(ii) Connection with RHC:** It is interesting to see that when $W_m(t) < T_l$, PLC mimics the commonly adopted receding-horizon-control method (RHC), e.g., [28]. The main difference is that, in RHC, future *states* are predicted and are directly fed into a predictive optimization formulation for computing the current action. Under PLC, *distribution prediction* is combined with *historic state information* to compute an empirical multiplier for augmenting the controller. In this regard, PLC can be viewed as exploring the benefits of statistics whenever it finds the system stationary (and does it automatically). **(iii) Parameter selection:** The parameters in PLC can be conveniently chosen as follows. First, fix a detection error probability $\delta = V^{-\log(V)}$. Then, choose a small ϵ_d and a d that satisfies $d \geq 4\log(V)^2/\epsilon_d^2 + w + 1$. Finally, choose $T_l = \max(V^c, e_w^{-2})$ and θ according to (17).

While recent works [16] and [17] also design learning-based algorithms that utilize historic information, they do not consider prediction and do not provide insight on its benefits and the impact of prediction error. Moreover, [16] focuses on stationary systems and [17] adopts a frame-based scheme.

7 PERFORMANCE ANALYSIS

This section presents the performance results of PLC. We focus on four metrics, detection efficiency, network utility, service delay, and algorithm convergence. The metrics are chosen to represent robustness, resource utilization efficiency, quality-of-service, and adaptability, respectively.

7.1 Detection and estimation

We first look at the detection and estimation part. The following lemma summarizes the performance of ADE, which is affected by the prediction accuracy as expected.

Lemma 7.1. *Under* $\texttt{ADE}(T_l, d, \epsilon_d)$*, we have:*

(a) Suppose at a time t*,* $\boldsymbol{\pi}(\tau_1) = \boldsymbol{\pi}_1$ *for* $\tau_1 \in \mathcal{W}_d(t)$ *and* $\boldsymbol{\pi}(\tau_2) = \boldsymbol{\pi}_2 \neq \boldsymbol{\pi}_1$ *for all* $\tau_2 \in \mathcal{W}_m(t)$ *and* $\max|\pi_{1i} - \pi_{2i}| > 4(w+1)e_w/d$*. Then, by choosing* $\epsilon_d < \epsilon_0 \triangleq \max|\pi_{1i} - \pi_{2i}|/2 - (w+1)e_w/d$ *and* $d > \ln\frac{4}{\delta}\cdot\frac{1}{2\epsilon_d^2} + w + 1$*, if* $W_m(t) \geq W_d(t) = d$*, with probability at least* $1-\delta$*,* $b_m(t+1) = t+w+1$ *and* $\mathcal{W}_m(t+1) = \phi$*, i.e.,* $W_m(t+1) = 0$.

(b) Suppose $\boldsymbol{\pi}(t) = \boldsymbol{\pi}\ \forall t$*. Then, if* $W_m(t) \geq W_d(t) = d$*, under* $\texttt{ADE}(T_l, d, \epsilon_d)$ *with* $d \geq \ln\frac{4}{\delta}\cdot\frac{2}{\epsilon_d^2} + w + 1$*, with probability at least* $1 - \delta - (w+1)MT_l^{-2\log(T_l)}$*,* $b_m(t+1) = b_m(t)$. ◇

Proof. See Appendix A. □

Lemma 7.1 shows that for a stationary system, i.e., $\boldsymbol{\pi}(t) = \boldsymbol{\pi}$, $W_m(t)$ will likely grow to a large value (Part (b)), in which case $\boldsymbol{\pi}_a(t)$ will stay close to $\boldsymbol{\pi}$ most of the time. If instead $\mathcal{W}_m(t)$ and $\mathcal{W}_d(t)$ contain samples from different distributions, ADE will reset $\mathcal{W}_m(t)$ with high probability. Note that since the first $w+1$ slots are predicted, this means that PLC detects changes $O(w)$ slots faster compared to that without prediction. The condition $\max|\pi_{1i} - \pi_{2i}| > 4(w+1)e_w/d$ can be understood as follows. If we want to distinguish two different distributions, we want the detection threshold to be no more than half of the distribution distance. Now with prediction, we want the potential prediction error to be no more than half of the threshold, hence the factor 4. Also note that the delay involved in detecting a distribution change is nearly order-optimal, in that it requires only $d = O(1/\min_i|\pi_{1i} - \pi_{2i}|^2)$ time, which is known to be necessary for distinguishing two distributions [42]. Moreover, $d = O(\ln(1/\delta))$ shows that a logarithmic window size is enough to ensure a high detection accuracy.

7.2 Utility and delay

In this section, we look at the utility and delay performance of PLC. To state our results, we first define the following structural property of the system.

Definition 7.2. A system is called polyhedral with parameter $\rho > 0$ under distribution $\boldsymbol{\pi}$ if the dual function $g(\boldsymbol{\gamma}, \boldsymbol{\pi})$ satisfies:

$$g(\boldsymbol{\gamma}^*, \boldsymbol{\pi}) \geq g(\boldsymbol{\gamma}, \boldsymbol{\pi}) + \rho\|\boldsymbol{\gamma}_{\boldsymbol{\pi}}^* - \boldsymbol{\gamma}\|. \ \diamond \tag{16}$$

The polyhedral property typically holds for practical systems, especially when action sets are finite (see [38] for more discussions).

7.2.1 Stationary system. We first consider stationary systems, i.e., $\boldsymbol{\pi}(t) = \boldsymbol{\pi}$. Our theorem shows that PLC achieves the near-optimal utility-delay tradeoff for stationary networks. This result is

important, as any good adaptive algorithm must be able to handle stationary settings well.

Theorem 7.3. *Suppose $\boldsymbol{\pi}(t) = \boldsymbol{\pi}$, the system is polyhedral with $\rho = \Theta(1)$, $e_w > 0$, and $\boldsymbol{q}(0) = \mathbf{0}$. Choose $0 < \epsilon_d < \epsilon_0 \triangleq 2(w+1)e_w/d$, $d = \log(V)^3/\epsilon_d^2$, $T_l = \max(V^c, e_w^{-2})$ for $c \in (0,1)$ and*

$$\theta = 2\log(V)^2(1 + \frac{V}{\sqrt{T_l}}), \tag{17}$$

Then, with a sufficiently large V, PLC achieves the following:

(a) Utility: $f_{av}^{\mathsf{PLC}} = f_{av}^{\boldsymbol{\pi}} + O(1/V)$

(b) Delay: For all but an $O(\frac{1}{V})$ fraction of traffic, the average packet delay is $D = O(\log(V)^2)$

(c) Dropping: The packet dropping rate is $O(V^{-1})$. ◇

Proof. Omitted due to space limitation. Please see [43] for details. □

Choosing $\epsilon = 1/V$, we see that PLC achieves the near-optimal $[O(\epsilon), O(\log(1/\epsilon)^2)]$ utility-delay tradeoff. Moreover, prediction enables PLC to also greatly reduce the queue size (see Part (b) of Theorem 7.4). Our result is different from the results in [20] and [22] for proactive service settings, where delay vanishes as prediction power increases. This is because we only assume observability of future states but not pre-service, and highlights the difference between pre-service and pure prediction. Note that the performance of PLC does not depend heavily on ϵ_d in Theorem 7.3. The value ϵ_d is more crucial for non-stationary systems, where a low false-negative rate is critical for performance. Also note that although packet dropping can occur during operation, the fraction of packets dropped is very small, and the resulting performance guarantee cannot be obtained by simply dropping the same amount of packets, in which case the delay will still be $\Theta(1/\epsilon)$.

Although Theorem 7.3 has a similar form as those in [17] and [16], the analysis is very different, in that (i) prediction error must be taken into account, and (ii) PLC performs sequential detection and decision-making.

7.2.2 Piecewise stationary system. We now turn to the non-stationary case and consider the scenario where $\boldsymbol{\pi}(t)$ changes over time. In this case, we see that prediction is critical as it significantly accelerates convergence and helps to achieve good performance when each distribution only lasts for a finite time. As we know that when the distribution can change arbitrarily, it is hard to even define optimality. Thus, we consider the case when the system is piecewise stationary, i.e., each distribution lasts for a duration of time, and study how the algorithm optimizes the performance for each distribution.

The following theorem summarizes the performance of PLC in this case. In the theorem, we define $D_k \triangleq t_k + d - t^*$, where $t^* \triangleq \sup\{t < t_k + d : t \text{ is a reset point}\}$, i.e., the most recent ending time after having a cycle with size T_l (recall that reset points are marked in step (ii) of ADE and $d \geq w+1$).

Theorem 7.4. *Suppose $d_k \geq 4d$ and the system is polyhedral with $\rho = \Theta(1)$ for all k,. Also, suppose there exists $\epsilon_0^* = \Theta(1) > 0$ such that $\epsilon_0^* \leq \inf_{k,i} |\pi_{ki} - \pi'_{k-1i}|$ and $\boldsymbol{q}(0) = \mathbf{0}$. Choose $\epsilon_d < \epsilon_0^*$ in ADE, and choose d, θ and T_l as in Theorem 7.3. Fix any distribution $\boldsymbol{\pi}_k$ with length $d_k = \Theta(V^{1+a}T_l)$ for some $a = \Theta(1) > 0$. Then, under PLC with a sufficiently large V, if $W_m(t_k)$ only contains samples after t_{k-1}, we achieve the following with probability $1 - O(V^{-3\log(V)/4})$:*

(a) Utility: $f_{av}^{\mathsf{PLC}} = f_{av}^{\boldsymbol{\pi}_k} + O(1/V) + O(\frac{D_k \log(V)}{T_l V^{1+a}})$

(b) Queueing: $\overline{q}_{av} = O((\min(V^{1-c/2}, Ve_w)+1)\log^2(V) + D_k + d)$.

(c) In particular, if $d_{k-1} = \Theta(T_l V^{a_1})$ for $a_1 = \Theta(1) > 0$ and $W_m(t_{k-1})$ only contains samples after t_{k-2}, then with probability $1 - O(V^{-2})$, $D_k = O(d)$, $f_{av}^{\mathsf{PLC}} = f_{av}^{\boldsymbol{\pi}_k} + O(1/V)$ and $\overline{q}_{av} = O(\min(V^{1-c/2}, Ve_w) + \log^2(V))$. ◇

Proof. Omitted due to space limitation. Please see [43] for details. □

A few remarks are in place. (i) Theorem 7.4 shows that, with an increasing prediction power, i.e., a smaller e_w, it is possible to simultaneously reduce network queue size and the time it takes to achieve a desired average performance (even if we do not execute actions ahead of time). The requirement $d_k = \Theta(V^{1+a}T_l)$ can be strictly less than the $O(V^{2-c/2+a})$ requirement for RLC in [17] and the $O(V^2)$ requirement of BP for achieving the same average utility. This implies that PLC finds a good system operating point faster than previous algorithms, a desirable feature for network algorithms. (ii) The dependency on D_k here is necessary. This is because PLC does not perform packet dropping if previous intervals do not exceed length T_l. As a result, the accumulated backlog can affect decision making in the current interval. Fortunately the queues are shown to be small and do not heavily affect performance (also see simulations). (iii) To appreciate the queueing result, note that BP (without learning) under the same setting will result in an $O(V)$ queue size.

Compared to the analysis in [17], one complicating factor in proving Theorem 7.4 is that ADE may not always throw away samples from a previous interval. Instead, ADE ensures that with high probability, only $o(d)$ samples from a previous interval will remain. This ensures high learning accuracy and fast convergence of PLC. One interesting special case not covered in the last two theorems is when $e_w = 0$. In this case, prediction is perfect and $T_l = \infty$, and PLC always runs with $\boldsymbol{\pi}_a(t) = \frac{1}{w+1}\sum_{k=0}^{w}\hat{\boldsymbol{\pi}}(t+k)$, which is the exact average distribution. For this case, we have the following result.

Theorem 7.5. *Suppose $e_w = 0$ and $\boldsymbol{q}(0) = \mathbf{0}$. Then, PLC achieves the following:*

(a) Suppose $\boldsymbol{\pi}(t) = \boldsymbol{\pi}$ and the system is polyhedral with $\rho = \Theta(1)$. Then, under the conditions of Theorem 7.3, PLC achieves the $[O(\epsilon), O(\log(1/\epsilon)^2)]$ utility-delay tradeoff.

(b) Suppose $d_k \geq d\log^2(V)$ and the system is polyhedral with $\rho = \Theta(1)$ under each $\boldsymbol{\pi}_k$. Under the conditions of Theorem 7.4, for an interval $d_k \geq V^{1+\epsilon}$ for any $\epsilon > 0$, PLC achieves that $f_{av}^{\mathsf{PLC}} = f_{av}^{\boldsymbol{\pi}_k} + O(1/V)$ and $\mathbb{E}\{\boldsymbol{q}(t_k)\} = O(\log^4(V))$. ◇

Proof. Omitted due to space limitation. Please see [43] for details. □

The intuition here is that since prediction is perfect, $\boldsymbol{\pi}_a(t) = \boldsymbol{\pi}_k$ during $[t_k + d, t_{k+1} - w]$. Therefore, a better performance can be achieved. The key challenge in this case is that PLC does not perform any packet dropping. Thus, queues can build up and one needs to

show that the queues will be concentrating around $\theta \cdot \mathbf{1}$ even when the distribution changes.

7.3 Convergence time

We now consider the algorithm convergence time, which is an important evaluation metric and measures how long it takes for an algorithm to reach its steady-state. While recent works [17], [16], [44], and [45] also investigate algorithm convergence time, they do not consider utilizing prediction in learning and do not study the impact of prediction error.

To formally state our results, we adopt the following definition of convergence time from [16].

Definition 7.6. Let $\zeta > 0$ be a given constant and let $\boldsymbol{\pi}$ be a system distribution. The ζ-convergence time of a control algorithm, denoted by T_ζ, is the time it takes for the effective queue vector $\boldsymbol{Q}(t)$ to get to within ζ distance of $\boldsymbol{\gamma}^*_{\boldsymbol{\pi}}$, i.e.,

$$T_\zeta \triangleq \inf\{t \,|\, ||\boldsymbol{Q}(t) - \boldsymbol{\gamma}^*_{\boldsymbol{\pi}}|| \leq \zeta\}. \ \diamond \tag{18}$$

We have the following theorem. Recall that $w \leq d = \Theta(\log(V)^2)$.

Theorem 7.7. *Assuming all conditions in Theorem 7.4, except that $\boldsymbol{\pi}(t) = \boldsymbol{\pi}_k$ for all $t \geq t_k$. If $e_w = 0$, under* PLC*,*

$$\mathbb{E}\{T_G\} = O(\log^4(V)). \tag{19}$$

Else suppose $e_w > 0$. Under the conditions of Theorem 7.4, with probability $1 - O(\frac{1}{VT_l} + \frac{D_k}{V^2 T_l})$,

$$\mathbb{E}\{T_G\} = O(\theta + T_l + D_k + w) \tag{20}$$

$$\mathbb{E}\{T_{G_1}\} = O(d). \tag{21}$$

Here $G = \Theta(1)$ and $G_1 = \Theta(D_k + 2\log(V)^2(1 + Ve_w))$, where D_k is defined in Theorem 7.4 as the most recent reset point prior to t_k. In particular, if $d_{k-1} = \Theta(T_l V^{a_1})$ for some $a_1 = \Theta(1) > 0$ and $\theta = O(\log(V)^2)$, then with probability $1 - O(V^{-2})$, $D_k = O(d)$, and $\mathbb{E}\{T_{G_1}\} = O(\log^2(V))$. $\diamond$

Proof. Omitted. Please see [43] for details. □

Here the assumption $\boldsymbol{\pi}(t) = \boldsymbol{\pi}_k$ for all $t \geq t_k$ is made to avoid the need for specifying the length of the intervals. It is interesting to compare (19), (20) and (21) with the convergence results in [16] and [17] without prediction, where it was shown that the convergence time is $O(V^{1-c/2}\log(V)^2 + V^c)$, with a minimum of $O(V^{2/3})$. Here although we may still need $O(V^{2/3})$ time for getting into an G-neighborhood (depending on e_w), getting to the G_1-neighborhood can take only an $O(\log^2(V))$ time, which is much faster compared to previous results, e.g., when $e_w = o(V^{-2})$ and $D_k = O(w)$, we have $G_1 = O(\log^2(V))$. This confirms our intuition that *prediction accelerates algorithm convergence* and demonstrates the power of (even imperfect) prediction.

8 SIMULATION

In this section, we present simulation results of PLC in a two-queue system shown in Fig. 3. Though being simple, the system models various settings, e.g., a two-user downlink transmission problem in a mobile network, a CPU scheduling problem with two applications,

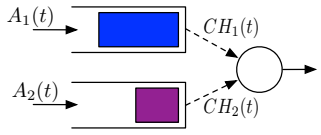

Figure 3: A single-server two-queue system. Each queue receives random arrivals. The server can only serve one queue at a time.

or an inventory control system where two types of orders are being processed.

$A_j(t)$ denotes the number of arriving packets to queue j at time t. We assume $A_j(t)$ is i.i.d. with either 1 or 0 with probabilities p_j and $1 - p_j$, and use $p_1 = 0.3$ and $p_2 = 0.6$. Thus, $\lambda_1 = 0.3$ and $\lambda_2 = 0.6$. Each queue has a time-varying channel condition. We denote $CH_j(t)$ the channel condition of queue j at time t. We assume that $CH_j(t) \in \mathcal{CH}_j$ with $\mathcal{CH}_1 = \{0, 1\}$ and $\mathcal{CH}_2 = \{1, 2\}$. The channel distributions are assumed to be uniform. At each time, the server determines the power allocation to each queue. We use $P_j(t)$ to denote the power allocated to queue j at time t. Then, the instantaneous service rate $q_j(t)$ gets is given by:

$$\mu_j(t) = \log(1 + CH_j(t)P_j(t)). \tag{22}$$

We assume that $P_j(t) \in \mathcal{P} = \{0, 1, 2\}$ for $j = 1, 2$, and at each time only one queue can be served. The objective is to stabilize the system with minimum average power. It can be verified that Assumptions 1 and 2 both hold in this example.

We compare PLC with BP in two cases. The first case is a stationary system where the arrival distributions remain constant. The second case is a non-stationary case, where we change the arrival distributions during the simulation. In both cases we simulate the system for $T = 5 \times 10^4$ slots. We simulate $V \in \{20, 50, 100, 150, 200, 300\}$. We set $w + 1 = 5$ and generate prediction error by adding uniform random noise to distributions with max value $e(k)$ (specified below). We also use $\epsilon_d = 0.1$, $\delta = 0.005$ and $d = 2\ln(4/\delta)/\epsilon^2 + w + 1$. We also simplify the choice of θ and set it to $\theta = \log(V)^2$.

We first examine the long-term performance. Fig. 4 shows the utility-delay performance of PLC compared to BP in the stationary setting. There are two PLC we simulated, one is with $e_w = 0$ (PLC) and the other is with $e_w = 0.04$ (PLC-e). From the plot, we see that both PLCs achieve a similar utility as BP, but guarantee a much smaller delay. The reason PLC-e has a better performance is due to packet dropping. We observe around an average packet dropping rate of 0.06. As noted before, the delay reduction of PLC cannot be achieved by simply dropping this amount of packets.

Next, we take a look at the detection and convergence performance of PLC. Fig. 5 shows the performance of PLC with perfect prediction ($e_w = 0$), PLC with prediction error ($e_w = 0.04$) and BP when the underlying distribution changes. Specifically, we run the simulation for $T = 5000$ slots and start with the arrival rates of $p_1 = 0.2$ and $p_2 = 0.4$. Then, we change them to $p_1 = 0.3$ and $p_2 = 0.6$ at time $T/2$.

We can see from the green and red curves that PLC quickly adapts to the change and modifies the Lagrange multiplier accordingly. By doing so, the actual queues under PLC (the purple and the brown curves) remain largely unaffected. For comparison, we see that BP takes a longer time to adapt to the new distribution and results in a larger queue size. We also see that during the 5000 slots,

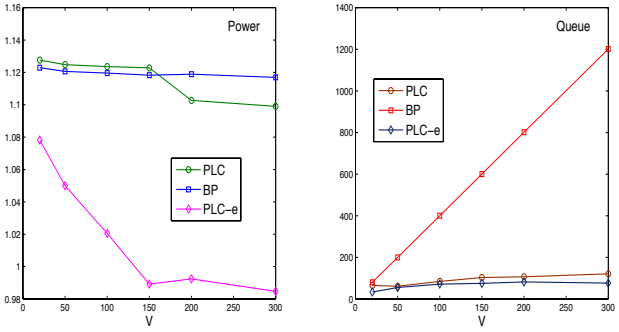

Figure 4: Utility and delay performance comparison between PLC and BP.

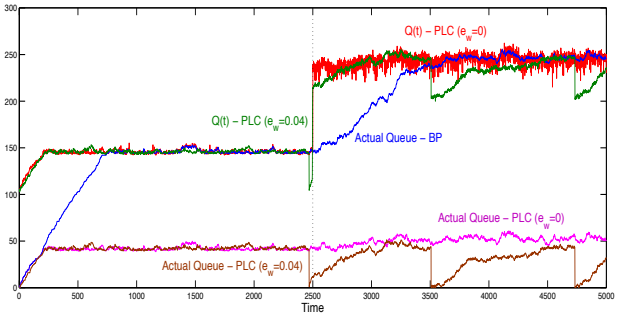

Figure 5: Convergence comparison between PLC and BP for queue 1 under $V = 100$. Here PLC ($e_w = 0$) is the perfect case and PLC ($e_w = 0.04$) contains prediction error. Both versions converge much faster compared to BP.

PLC ($e_w = 0.04$) drops packets 3 times (zero for the first half), validating the results in Lemma 7.1 and Theorem 7.3. Moreover, after the distribution change, PLC ($e_w = 0.04$) quickly adapts to the new equilibrium, despite having imperfect prediction. The fast convergence result also validates our theorem about short term utility performance under PLC. Indeed, if we look at slots during time $200 - 500$, and slots between $2500 - 3500$, we see that when BP is learning the target backlog, PLC already operates near the optimal mode. This shows the benefits of prediction and learning in stochastic network control.

9 CONCLUSION

We investigate the problem of stochastic network optimization in the presence of imperfect state prediction and non-stationarity. Based on a novel distribution-accuracy curve prediction model, we develop the *predictive learning-aided control* (PLC) algorithm. PLC is an online algorithm that requires *zero* a-prior system statistical information, and contains three main functionalities, sequential distribution estimation and change detection, dual learning, and online queue-based control. We show that PLC simultaneously achieves good long-term performance, short-term queue size reduction, accurate change detection, and fast algorithm convergence. Our results demonstrate that state prediction (even imperfect) can help improve performance and quantify the benefits of prediction in four important metrics, i.e., utility (efficiency), delay (quality-of-service), detection (robustness), and convergence (adaptability). They provide new insight for joint prediction, learning and optimization in stochastic networks.

10 ACKNOWLEDGEMENT

The work of Longbo Huang was supported in part by the National Natural Science Foundation of China Grants 61672316, 61303195, the Tsinghua Initiative Research Grant, the China youth 1000-talent grant, and the Microsoft Research Asia collaborative research grant. The work of Minghua Chen was supported in part by National Basic Research Program of China (Project No. 2013CB336700) and the University Grants Committee of the Hong Kong Special Administrative Region, China (Area of Excellence Grant Project No. AoE/E-02/08 and General Research Fund No. 2150871).

REFERENCES

[1] L. Georgiadis, M. J. Neely, and L. Tassiulas. *Resource Allocation and Cross-Layer Control in Wireless Networks.* Foundations and Trends in Networking Vol. 1, no. 1, pp. 1-144, 2006.
[2] Y. Chon, E. Talipov, H. Shin, and H. Cha. Mobility prediction-based smartphone energy optimization for everyday location monitoring. *ACM Sensys*, 2011.
[3] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica. Effective straggler mitigation: Attack of the clones. *ACM NSDI*, 2014.
[4] X. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, . Jin, J. Rexford, and R. K. Sinha. Can accurate predictions improve video streaming in cellular networks? *ACM HotMobile*, 2015.
[5] TechCrunch. Amazon patents "anticipatory" shipping - to start sending stuff before you've bought it. *http://techcrunch.com/2014/01/18/amazon-pre-ships/*, Jan 2014.
[6] Adweek. Facebook begins prefetching to improve mobile site speed. *http://www.adweek.com/socialtimes/prefetching/644281*, Aug 2016.
[7] M. Gatzianas, L. Georgiadis, and L. Tassiulas. Control of wireless networks with rechargeable batteries. *IEEE Trans. on Wireless Communications, Vol. 9, No. 2*, Feb. 2010.
[8] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *IEEE/ACM Trans. Netw.*, 15(6):1333–1344, 2007.
[9] B. Li and R. Srikant. Queue-proportional rate allocation with per-link information in multihop networks. *Proceedings of ACM Sigmetrics*, 2015.
[10] B. Ji and Y. Sang. Throughput characterization of node-based scheduling in multihop wireless networks: A novel application of the gallai-edmonds structure theorem. *Proceedings of ACM MobiHoc*, 2016.
[11] H. Zhao, C. H. Xia, Z. Liu, and D. Towsley. A unified modeling framework for distributed resource allocation of general fork and join processing networks. *Proc. of ACM Sigmetrics*, 2010.
[12] L. Jiang and J. Walrand. Stable and utility-maximizing scheduling for stochastic processing networks. *Allerton Conference on Communication, Control, and Computing*, 2009.
[13] R. Urgaonkar and M. J. Neely. Opportunistic scheduling with reliability guarantees in cognitive radio networks. *IEEE Transactions on Mobile Computing*, 8(6):766–777, June 2009.
[14] H. Su and A. El Gamal. Modeling and analysis of the role of fast-response energy storage in the smart grid. *Proc. of Allerton*, 2011.
[15] M. J. Neely R. Urgaonkar, B. Urgaonkar and A. Sivasubramaniam. Optimal power cost management using stored energy in data centers. *Proceedings of ACM Sigmetrics*, June 2011.
[16] L. Huang, X. Liu, and X. Hao. The power of online learning in stochastic network optimization. *Proceedings of ACM Sigmetrics*, 2014.
[17] L. Huang. Receding learning-aided control in stochastic networks. *IFIP Performance*, Oct 2015.
[18] J. Tadrous, A. Eryilmaz, and H. El Gamal. Proactive resource allocation: harnessing the diversity and multicast gains. *IEEE Tansactions on Information Theory*, 2013.
[19] J. Spencer, M. Sudan, and K Xu. Queueing with future information. *ArXiv Technical Report arxiv:1211.0618*, 2012.
[20] S. Zhang, L. Huang, M. Chen, and X. Liu. Proactive serving reduces user delay exponentially. *Proceedings of ACM Sigmetrics (Poster Paper)*, 2014.
[21] K. Xu. Necessity of future information in admission control. *Operations Research*, 2015.
[22] L. Huang, S. Zhang, M. Chen, and X. Liu. When Backpressure meets Predictive Scheduling. *Proceedings of ACM MobiHoc*, 2014.
[23] L. Muppirisetty, J. Tadrous, A. Eryilmaz, and H. Wymeersch. On proactive caching with demand and channel uncertainties. *Proceedings of Allerton Conference*, 2015.
[24] S. Zhao, X. Lin, and M. Chen. Peak-minimizing online ev charging: Price-ofuncertainty and algorithm robustification. *Proceedings of IEEE INFOCOM*, 2015.

[25] N. Chen, A. Agarwal, A. Wierman, S. Barman, and L. L. H. Andrew. Online convex optimization using predictions. *Proceedings of ACM Sigmetrics*, 2015.

[26] N. Chen, J. Comden, Z. Liu, A. Gandhi, and A. Wierman. Using predictions in online optimization: Looking forward with an eye on the past. *Proceedings of ACM Sigmetrics*, 2016.

[27] M. Hajiesmaili, C. Chau, M. Chen, and L. Huang. Online microgrid energy generation scheduling revisited: The benefits of randomization and interval prediction. *Proceedings of ACM e-Energy*, 2016.

[28] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew. Online algorithms for geographical load balancing. *IEEE IGCC*, 2012.

[29] J. Tadrous, A. Eryilmaz, and H. El Gamal. Pricing for demand shaping and proactive download in smart data networks. *The 2nd IEEE International Workshop on Smart Data Pricing (SDP), INFOCOM*, 2013.

[30] M. Qu, H. Zhu, J. Liu, G. Liu, and H. Xiong. A cost-effective recommender system for taxi drivers. *ACM KDD*, 2014.

[31] L. Huang. The value-of-information in matching with queues. *IEEE/ACM Trans. on Netwroking*, to appear.

[32] O. Simeone C. Tapparello and M. Rossi. Dynamic compression-transmission for energy-harvesting multihop networks with correlated sources. *IEEE/ACM Trans. on Networking*, 2014.

[33] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. J. Neely. Data centers power reduction: A two time scale approach for delay tolerant workloads. *IEEE Transactions on Parallel and Distributed Systems (TPDS), vol. 25, no. 1, pp. 200-211*, Jan 2014.

[34] W. Wang, K. Zhu, Lei Ying, J. Tan, and L. Zhang. Map task scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality. *IEEE/ACM Transactions on Networking*, to appear.

[35] L. Gan, A. Wierman, U. Topcu, N. Chen, and S. Low. Real-time deferrable load control: Handling the uncertainties of renewable generation. *ACM e-Energy*, 2013.

[36] L. Ying, S. Shakkottai, and A. Reddy. On combining shortest-path and back-pressure routing over multihop wireless networks. *Proceedings of IEEE INFOCOM*, April 2009.

[37] L. Bui, R. Srikant, and A. Stolyar. Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing. *Proceedings of IEEE INFOCOM Mini-Conference*, April 2009.

[38] L. Huang and M. J. Neely. Delay reduction via Lagrange multipliers in stochastic network optimization. *IEEE Trans. on Automatic Control*, 56(4):842–857, April 2011.

[39] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Boston: Athena Scientific, 2003.

[40] L. Huang and M. J. Neely. Max-weight achieves the exact $[O(1/V), O(V)]$ utility-delay tradeoff under Markov dynamics. *arXiv:1008.0200v1*, 2010.

[41] L. Huang and M. J. Neely. The optimality of two prices: Maximizing revenue in a stochastic network. *IEEE/ACM Transactions on Networking*, 18(2):406–419, April 2010.

[42] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics, 6, 4-22.*, 1985.

[43] L. Huang, M. Chen, and Y. Liu. Learning-aided stochastic network optimization with imperfect state prediction. *arXiv report arXiv:1705.05058*, 2017.

[44] M. J. Neely. Energy-aware wireless scheduling with near optimal backlog and convergence time tradeoffs. *Proceedings of IEEE INFOCOM*, 2016.

[45] J. Liu. Achieving low-delay and fast-convergence in stochastic network optimization: A nesterovian approach. *Proceedings of ACM Sigmetrics*, 2016.

[46] Albert Bifet and Ricard Gavald. Learning from time-changing data with adaptive windowing. *SIAM International Conference on Data Mining*, 2007.

[47] F. Chung and L. Lu. Concentration inequalities and martingale inequalities - a survey. *Internet Math.*, 3 (2006-2007), 79–127.

[48] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58 (301): 13-30*, 1963.

APPENDIX A - PROOF OF LEMMA 7.1

(Proof of Lemma 7.1) We prove the performance of $\texttt{ADE}(T_l, d, \epsilon)$ with an argument inspired by [46]. We will make use of the following concentration result.

THEOREM 10.1. *[47] Let $X_1, ..., X_n$ be independent random variables with $Pr\{X_i = 1\} = p_i$, and $Pr\{X_i = 0\} = 1 - p_i$. Consider $X = \sum_{i=1}^n X_i$ with expectation $\mathbb{E}\{X\} = \sum_{i=1}^n p_i$. Then, we have:*

$$Pr\{X \leq \mathbb{E}\{X\} - m\} \leq e^{\frac{-m^2}{2\mathbb{E}\{X\}}}, \quad (23)$$

$$Pr\{X \geq \mathbb{E}\{X\} + m\} \leq e^{\frac{-m^2}{2(\mathbb{E}\{X\}+m/3)}}. \quad \diamond \quad (24)$$

PROOF. (Lemma 7.1) (**Part (a)**) In this case, it suffices to check condition (i) in ADE. Define

$$\tilde{\pi}_i^d(t) \triangleq \frac{1}{d}\Big(\sum_{\tau=(t+w-d)_+}^{t-1} 1_{[S(\tau)=s_i]} + \sum_{\tau\in\mathcal{W}_w(t)} \pi_i(\tau)\Big),$$

i.e., $\tilde{\pi}_i^d(t)$ is defined with the true distributions in $\mathcal{W}_d(t)$. Denote $\epsilon_1 = (w+1)e_w/d$, we see then $\|\tilde{\boldsymbol{\pi}}^d(t) - \hat{\boldsymbol{\pi}}^d(t)\| \leq \epsilon_1$. Thus, for any $\epsilon > 0$, we have:

$$\begin{aligned} & \Pr\{\|\hat{\boldsymbol{\pi}}^d(t) - \hat{\boldsymbol{\pi}}^m(t)\|_{tv} \leq \epsilon\} \\ \leq\ & \Pr\{\|\tilde{\boldsymbol{\pi}}^d(t) - \hat{\boldsymbol{\pi}}^m(t)\|_{tv} \leq \epsilon + \epsilon_1\} \\ \leq\ & \Pr\{|\tilde{\pi}_i^d(t) - \hat{\pi}_i^m(t)| \leq \epsilon + \epsilon_1\}. \end{aligned} \quad (25)$$

Choose $\epsilon = \frac{1}{2}\max|\pi_{1i} - \pi_{2i}| - 2\epsilon_1 > 0$ and let $\epsilon_0 = \epsilon + \epsilon_1$. Fix $\alpha \in (0,1)$ and consider $i \in \arg\max_i |\pi_{1i} - \pi_{2i}|$. We have:

$$\begin{aligned} & \Pr\{|\tilde{\pi}_i^d(t) - \hat{\pi}_i^m(t)| \leq \epsilon_0\} \\ & \leq \Pr\{\{|\tilde{\pi}_i^d(t) - \pi_{1i}| \geq \alpha\epsilon_0\} \\ & \qquad \cup\{|\hat{\pi}_i^m(t) - \pi_{2i}| \geq (1-\alpha)\epsilon_0\}\} \\ & \leq \Pr\{|\tilde{\pi}_i^d(t) - \pi_{1i}| \geq \alpha\epsilon_0\} \\ & \qquad + \Pr\{|\hat{\pi}_i^m(t) - \pi_{2i}| \geq (1-\alpha)\epsilon_0\}. \end{aligned} \quad (26)$$

Here the first inequality follows because if we have both $\{|\tilde{\pi}_i^d(t) - \pi_{1i}| < \alpha\epsilon_0\}$ and $\{|\hat{\pi}_i^m(t) - \pi_{2i}| < (1-\alpha)\epsilon_0$, and $|\tilde{\pi}_i^d(t) - \hat{\pi}_i^m(t)| \leq \epsilon_0$, we must have:

$$\begin{aligned} |\pi_{1i} - \pi_{2i}| &\leq |\tilde{\pi}_i^d(t) - \pi_{1i}| + |\hat{\pi}_i^m(t) - \pi_{2i}| + |\tilde{\pi}_i^d(t) - \hat{\pi}_i^m(t)| \\ &= 2\epsilon_0 < |\pi_{1i} - \pi_{2i}|, \end{aligned}$$

which contradicts the fact that i achieves $\max_i |\pi_{1i} - \pi_{2i}|$. Using (26) and Hoeffding inequality [48], we first have:

$$\Pr\{|\hat{\pi}_i^m(t) - \pi_{2i}| \geq (1-\alpha)\epsilon_0\} \leq 2\exp(-2((1-\alpha)\epsilon_0)^2 W_m(t)). \quad (27)$$

For the first term in (26), we have:

$$\begin{aligned} & \Pr\{|\tilde{\pi}_i^d(t) - \pi_{1i}| \geq \alpha\epsilon_0\} \\ & \qquad \leq 2\exp(-2(\alpha\epsilon_0)^2(W_d(t) - w - 1)). \end{aligned} \quad (28)$$

Equating the above two probabilities and setting the sum equal to δ, we have $\alpha = \frac{\sqrt{W_m(t)/(W_d(t)-w-1)}}{1+\sqrt{W_m(t)/(W_d(t)-w-1)}}$, and

$$\epsilon_0 = \sqrt{\ln\frac{4}{\delta}} \cdot \frac{1 + \sqrt{(W_d(t) - w - 1)/W_m(t)}}{\sqrt{2(W_d(t) - w - 1)}}. \quad (29)$$

In order to detect the different distributions, we can choose $\epsilon_d < \epsilon_0$, which on the other hand requires that:

$$\begin{aligned} \epsilon_d &\overset{(*)}{\leq} \sqrt{\ln\frac{4}{\delta}} \cdot \sqrt{\frac{1}{2(d-w-1)}} < \epsilon_0 \\ &\Rightarrow\ d > \ln\frac{4}{\delta} \cdot \frac{1}{2\epsilon_d^2} + w + 1. \end{aligned} \quad (30)$$

Here (*) follows because $W_d(t) = d \leq W_m(t)$. This shows that whenever $W_d(t) = d \leq W_m(t)$ and the windows are loaded with non-coherent samples, error will be detected with probability $1 - \delta$.

(**Part (b)**) Omitted due to space limitation. Please see [43] for details. □