Intelligence of Smart Systems: Model, Bounds, and Algorithms

Longbo Huang

***Abstract*— We present a general framework for understanding system intelligence, i.e., the level of system smartness perceived by users, and propose a novel metric for measuring the intelligence levels of dynamical human-in-the-loop systems, defined to be the maximum average reward obtained by proactively serving user demands, subject to a resource constraint. Our metric captures two important elements of smartness, i.e., being able to know what users want and pre-serve them, and achieving good resource management while doing so. We provide an explicit characterization of the system intelligence, and show that it is jointly determined by user demand volume (opportunity to impress), demand correlation (user predictability), and system resource and action costs (flexibility to pre-serve). We then propose an online learning-aided control algorithm called *learning-aided budget-limited intelligent system control* (LBISC), and show that LBISC achieves an intelligence level that is within $O(N(T)^{-\frac{1}{2}}+\epsilon)$ of the highest level, where $N(T)$ represents the number of data samples collected within a learning period T and is proportional to the user population size, while guaranteeing an $O(\max(N(T)^{-\frac{1}{2}}/\epsilon, \log(1/\epsilon)^2))$ average resource deficit. Moreover, we show that LBISC possesses an $O(\max(N(T)^{-\frac{1}{2}}/\epsilon, \log(1/\epsilon)^2)+T)$ convergence time, which is smaller compared with the $\Theta(1/\epsilon)$ time required for existing non-learning-based algorithms. Our analysis rigorously quantifies the impact of data and user population (captured by $N(T)$), learning (captured by our learning method), and control (captured by LBISC) on the achievable system intelligence, and provides novel insight and guideline into designing future smart systems.**

***Index Terms*— Network control, queueing theory, Lyapunov analysis.**

I. Introduction

DUE to rapid developments in sensing and monitoring, machine learning, and hardware manufacturing, building intelligence into systems has recently received strong attention, and clever technologies and products have been developed to enhance user experience. For instance, recommendation systems [1], smart home [2], artificial intelligence engines [3], and user behavior prediction [4]. Despite the prevailing success in practice, there has not been much theoretical understanding about system smartness. In particular, how do we measure the intelligence level of a system, how do we compare two systems and decide which one is smarter, which components of a system contribute most to the level of smartness, can the intelligence level of a system be pushed arbitrarily high.

Motivated by these fundamental questions, we propose a general framework for modeling system smartness and propose a novel metric for measuring system intelligence. Specifically, we consider a discrete time system that serves a set of applications for a user. The status of each application changes from time to time, resulting in different demand that needs to be fulfilled. At each time, the server observes the demand condition of each application over time and decides whether to *pre-serve* (serve before the user even places a request) the demand that can come in the next slot (which may not be present then), or to do nothing now and serve it then if it arrives. Depending on whether demand is served passively or proactively, the server receives different rewards representing user's satisfaction levels, or equivalently, different user perception of system smartness (a system that can serve us before being asked is often considered smarter). On the other hand, due to time-varying service conditions, the service actions incur different costs. The objective of the server is to design a control policy that achieves the *system intelligence*, defined to be the maximum achievable reward rate subject to a constraint on the average cost expenditure.

This formulation models many examples in practice. For example, newsfeed pushing and video prefetching [5], [6], instant searching [7], and branch prediction in computer architecture [8], [9]. It captures two key elements of a smart system, i.e., being able to know what users want and pre-execute actions, and performing good resource management while doing so. Note that resource management here is critical. Indeed, one can always pre-serve all possible demands to impress users at the expense of inefficient resource usage, but an intelligent system should do more than that.

Solving this problem is non-trivial. First of all, rewards generated by actions depend on their execution timing, i.e., before or after requests. Thus, this problem is different from typical network control problems, where outcomes of traffic serving actions are independent of timing. Second, application demands are often correlated over time. Hence, algorithms must be able to handle this issue, and to efficiently explore such correlations. Third, statistical information of the system dynamics are often unknown. Hence, control algorithms must be able to quickly learn and utilize the information, and be robust against errors introduced in learning.

Manuscript received May 21, 2016; revised January 31, 2017 and May 29, 2017; accepted June 3, 2017; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Paschos. Date of publication July 18, 2017; date of current version October 13, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61672316 and Grant 61303195, in part by the Tsinghua Initiative Research Grant, in part by the Microsoft Research Asia Collaborative Research Award, and in part by the China Youth 1000-Talent Grant.

The author is with IIIS, Tsinghua University, Beijing 100084, China (e-mail: longbohuang@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TNET.2017.2723300

There have been previous works studying optimal stochastic system control with resource management. Reference [10] designs algorithms for minimizing energy consumption of a stochastic network. Reference [11] studies the tradeoff between energy and robustness for downlink systems. References [12] and [13] develop algorithms for achieving the optimal utility-delay tradeoff in multihop networks. Reference [14] studies the problem of scheduling delay-constrained flows over wireless systems. However, all these works focus only on causal systems, i.e., service begins only after demand enters the system. Recent works [15]–[19] consider queueing system control with future traffic demand information. Works [20] and [21] also consider similar problems where systems can proactively serve user demand. They obtain interesting results that characterize cost reduction under proactive service, and the impact of number of users and prediction in terms of proactive service window size. However, system utilities in the aforementioned works are measured by average metrics, e.g., throughput or outage probabilities, and actions taken at different times for serving traffic are considered equivalent. Moreover, they do not investigate the impact of predictability and benefits of learning. Works [22] and [23] also consider learning and systems with Markov dynamics separately. Yet, the joint setting is different and not considered. Therefore, investigating the current problem requires non-trivial extensions of their results.

We tackle the problem by first establishing an explicit characterization of the maximum achievable intelligence level, which provides a fundamental limit of system intelligence and reveals that it is jointly determined by user demand volume (opportunity to impress), demand correlation (user predictability), and system resource and control costs (flexibility to pre-serve). Then, by carefully defining effective rewards and costs that represent action outcomes in consecutive slots, we propose an ideal control algorithm that assumes perfect system statistics, called *budget-limited intelligent system control* (`BISC`).

We further develop Learning-aided `BISC` (`LBISC`), by incorporating a maximum-likelihood-estimator (`MLE`) for estimating statistics, and a dual learning component (`DL`) [22] for learning an empirical Lagrange multiplier that can be integrated into `BISC`, to facilitate algorithm convergence and reduce resource deficit. We show that `LBISC` achieves a system intelligence that can be pushed arbitrarily close to the highest value while ensuring a deterministic budget deficit bound. Furthermore, we investigate the *user-population effect* in system intelligence, and rigorously quantify the degree to which the user population size can impact algorithm performance, i.e., algorithm convergence speed can be boosted by a factor that is proportional to the *square-root* of the user population. The analysis of `LBISC` quantifies how system intelligence depends on system resource, action costs, data sample size, and control algorithm. To the best of our knowledge, we are the first to propose a rigorous metric for quantifying system intelligence and jointly analyze the effects of different factors.

The contributions of this paper are summarized as follows.

- We propose a mathematical model for investigating intelligence in smart systems. Our model captures important components including observation (data), learning and prediction (model training), and algorithm (control).
- We propose a novel metric for measuring system intelligence, and explicitly characterize the optimal intelligence level. The characterization shows that intelligence is jointly determined by system resource and action costs (flexibility to pre-serve), steady-state user demand (opportunity to impress), and demand correlation (predictability, captured by demand *transition rates*).
- We propose an online learning-aided algorithm, called *learning-aided budget-limited intelligent system control* (`LBISC`). `LBISC` consists of three components, (i) a maximum-likelihood-estimator (`MLE`) for learning system statistics, (ii) a dual-learning component (`DL`) for learning a control-critical empirical Lagrange multiplier, and (iii) an online queue-based controller based on carefully-defined effective action rewards and costs.
- We show that `LBISC` achieves an intelligence level that is within $O(N(T)^{-1/2}+\epsilon)$ of the maximum, where T is the algorithm learning time, $N(T)$ is the number of data samples collected in learning and is proportional to user population of the system, and $\epsilon > 0$ is an tunable parameter, while guaranteeing an average resource deficit of $O(\max(N(T)^{-\frac{1}{2}}/\epsilon, \log(1/\epsilon)^2))$.
- We prove that `LBISC` achieves a convergence time of $O(T + \max(N(T)^{-1/2}/\epsilon, \log(1/\epsilon)^2))$, which can be much smaller than the $\Theta(1/\epsilon)$ time for its non-learning counterpart. The performance of `LBISC` shows that a company with more users has significant advantage over those with fewer, in that its algorithm convergence can be boosted by a factor that is proportional to the *square-root* of the user population.

The rest of the paper is organized as follows. In Section II, we provide a few examples of smart system. We then present our general model and problem formulation in Section III, and characterize the optimal system intelligence in Section IV. Our algorithms are presented in Section V. Analysis is given in Section VI. Simulation results are presented in Section VII, followed by the conclusion in Section VIII.

II. Examples

In this section, we provide a few examples that will serve both as explanations and motivation for our general model.

Instant Searching [7], [24]: Imagine you are searching on a search engine. When you start typing, the search engine tries to guess whether you will type in a certain keyword (or a set of related search queries) and pre-computes search results that it believes are relevant (predict and pre-service). If the server predicts correctly, results can be displayed immediately after typing is done, and search latency will be significantly reduced, resulting in a great user experience (high reward). If the prediction is inaccurate, the search engine can still process the query after getting the keyword, with the user being less impressed by the performance (low reward) and resources being wasted computing wrong results (cost).

Video Streaming [5]: When a user is watching videos on Youtube or a smart mobile device, the server can predict whether the user wants a particular video clip, and pre-load the

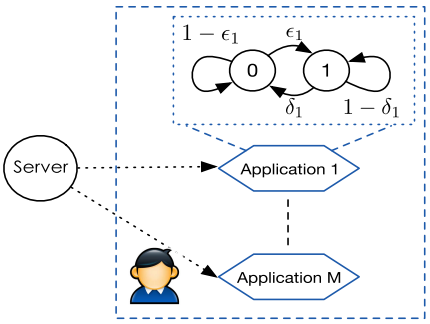

Fig. 1. A multi-application system that serves a set of applications of a customer.

video to the user device (predict and pre-service). This way, if the prediction is correct, the user's experience will be greatly improved and he enjoys a large satisfaction (high reward). If the prediction is incorrect, the bandwidth and energy spent in pre-loading are wasted (cost), but the server can still stream the content video to on the fly, potentially with a degraded quality-of-service (low reward).

Smart Home [25]: Consider a smart home environment where a thermostat manages room temperatures in the house. Depending on its prediction about the behavior of hosts, the thermostat can pre-heat/pre-cool some of the rooms (predict and pre-service). If the host enters a room where temperature is already adjusted, he receives a high satisfaction (high reward). If the prediction is incorrect, the room temperature can still be adjusted, but may affect user experience (low reward). Moreover, the energy spent is wasted (cost).

In the above examples, the smart level of a system perceived by users is closely related to whether his demand is served proactively, and whether such predictive service is carried out without too much unnecessary resource expenditure. These factors will be made precise in our general given in the next section.

III. System Model

We consider a system where a single server is serving a customer with M applications (Fig. 1). Here each application can represent, e.g., a smartphone application, watching a particular video clip, or a certain computing task the customer executes regularly. We assume that the system operates in slotted time, i.e., $t \in \{0, 1, \ldots\}$.

A. The Demand Model

We use $\boldsymbol{A}(t) = (A_m(t), m = 1, \ldots, M)$ to denote the demand state of the customer at time t. We assume that $A_m(t) \in \{0, 1\}$, where $A_m(t) = 1$ means there is a unit demand from application m at time t and $A_m(t) = 0$ otherwise. For instance, if application m represents a video clip watching task, $A_m(t)$ can denote whether the users wants to watch the video clip in the current slot. If so, the server needs to stream the video clip to the customer's device.

We assume that for each application m, $A_m(t)$ evolves according to an independent two-state Markov chain depicted in Fig. 1, where the transition probabilities ϵ_m and δ_m are as shown in the figure. This ON/OFF model captures the fact that human user actions are often correlated and predictable. It has also been commonly adopted for modeling network traffic states, e.g., [26], [27]. Our choice of the two-state Markov chain model is to both capture the correlated and predictable features of human-behavior and facility analysis. It is possible to adopt a more general multi-state Markov chain for each application.[1] We assume that ϵ_m and δ_m are unknown to the server, but the actual states can be observed every time slot. This assumption is due to the fact that the states essentially denote whether or not the user requests a particular service from the server. Thus, by observing the user's response one can see the states.

B. The Service and Cost Model

In every time slot t, the system serves application m's demand as follows. If $A_m(t)$ has not yet been served in slot $t-1$, it will be served in the current slot. Otherwise the current demand is considered completed. Then, in addition to serving the current demand, the server can also try to *pre-serve* the demand in time $t+1$. We denote $\mu_{mc}(t) \in \{0, 1\}$ the action taken to serve the current demand and $\mu_{mp}(t) \in \{0, 1\}$ the action taken to serve the demand in time $t+1$.[2] We have:

$$\mu_{mc}(t) = \max[A_m(t) - \mu_{mp}(t-1), 0]. \tag{1}$$

That is, demand will be fulfilled in the same time slot. Since $\mu_{mc}(t)$ is completely determined by $\mu_{mp}(t-1)$ and $A_m(t)$, we define $\boldsymbol{\mu}(t) \triangleq (\mu_{mp}(t), \forall\, m)$ for notation simplicity and view $\boldsymbol{\mu}(t)$ as the only control action made at time t.

We assume that each service to application m, either proactive or passive, consumes certain resources, e.g., due to energy expenditure or bandwidth consumption. To capture the fact that the condition under which actions are taken may be time-varying, we denote $S_m(t)$ the resource state for application m at time t, which affects how much resource is needed for service, e.g., channel condition of a wireless link, or cost spent for getting a particular video clip from an external server. We denote $\boldsymbol{S}(t) = (S_1(t), \ldots, S_M(t))$ the overall system resource state, and assume that $\boldsymbol{S}(t) \in \mathcal{S} = \{s_1, \ldots, s_K\}$ with $\pi_k = \Pr\{\boldsymbol{S}(t) = s_k\}$ and is i.i.d. every slot (also independent of $\boldsymbol{A}(t)$). Here we assume that the server can observe the instantaneous state $\boldsymbol{S}(t)$ and the $\{\pi_k\}$ values are known. This assumption is made to allow us to focus on the user demand aspect. It is also not restrictive, as $\boldsymbol{S}(t)$ is a non-human parameter and can often be learned from observations when serving different applications, whereas $\boldsymbol{A}(t)$ is more personalized and targeted learning is needed. Our method also applies to the case when $\{\pi_k, k = 1, \ldots, K\}$ are unknown.

[1] Markov models are widely used for modeling user behavior for video streaming [28], smart home applications [29], [30], and internet traffic. Our results can likely be extended to model systems where user behavior exhibits periodic patterns. In these scenarios, Markov models can be built for user behavior in different time periods and learned with data collected in those periods.

[2] Our results can be extended to having $\mu_{mp}(t) \in [0, 1]$, in which case partial pre-service is allowed. They can also be further extended to include pre-service over multiple slots using a similar frame-based design approach, e.g., [31].

To model the fact that a given resource state s_k typically constrains the set of feasible actions, we denote $\mathcal{U}_{\boldsymbol{S}(t)}$ the set of feasible actions under $\boldsymbol{S}(t)$. Examples of $\mathcal{U}_{\boldsymbol{S}(t)}$ include $\mathcal{U}_{s_k} = \{0,1\}^M$ for the unconstrained case, or $\mathcal{U}_{s_k} = \{\boldsymbol{\mu} \in \{0,1\}^M : \sum_m \mu_{mp} \leq N_k\}$ when we are allowed to pre-serve only N_k applications. We assume that $\mathcal{U}_{s_k}$ is compact and if $\boldsymbol{\mu} \in \mathcal{U}_{s_k}$, then a vector obtained by setting any entry in $\boldsymbol{\mu}$ to zero remains in $\mathcal{U}_{s_k}$.[3]

Under the resource state, the instantaneous cost incurred to the server is given by $C(t) = \sum_m C_m(t)$, where

$$C_m(t) \triangleq C_m(\mu_{mc}(t), \boldsymbol{S}(t)) + C_m(\mu_{mp}(t), \boldsymbol{S}(t)). \quad (2)$$

With (2), we assume that the cost in each slot is linear in $\boldsymbol{\mu}(t)$, a model that fits situations where costs for serving applications are additive, e.g., amount of bandwidth required for streaming videos. We assume that $C_m(\cdot, s_k) = 0$ is continuous, $C_m(0, s_k) = 0$ and $C_m(1, s_k) \leq C_{\max}$ for some $C_{\max} < \infty$ for all k and m. We define

$$C_{\text{av}} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{C(\tau)\} \quad (3)$$

as the average cost spent serving the demand. For notation simplicity, we also denote $\overline{C}_m \triangleq \sum_k \pi_k C_m(1, s_k)$ as the expected cost for serving one unit demand.

C. The Reward Model

In each time slot, serving each application demand generates a reward to the server. We use $r_m(t)$ to denote the reward collected in time t from application m, which takes the following form:

$$r_m(t) = \begin{cases} 0 & A_m(t) = 0 \\ r_{mc} & A_m(t) = 1 \ \& \ \mu_{mp}(t-1) = 0 \\ r_{mp} & A_m(t) = 1 \ \& \ \mu_{mp}(t-1) = 1 \end{cases} \quad (4)$$

By varying the values of r_{mp} and r_{mc}, we can model different sensitivity levels of the user to pre-service. We assume that $r_{mp} \geq r_{mc}$ are both known to the server.[4] This is natural for capturing the fact that a user typically gets more satisfaction if his demand is pre-served. We denote $r_d \triangleq \max_m (r_{mp} - r_{mc})$ the maximum reward difference between pre-service and passive service.

To evaluate the performance of a control policy, we define the following average reward rate, i.e.,

$$r_{\text{av}} = \liminf_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_m \mathbb{E}\{r_m(\tau)\}. \quad (5)$$

r_{av} is a natural index of system smartness. A higher value of r_{av} implies that the server can better predict what the user needs and pre-serves him. As a result, the user experience better service and perceives a smarter system. In the special case when $r_{mp} = 1$ and $r_{mc} = 0$, the average reward equals the rate of correct prediction.

D. System Objective

In every time slot, the server accumulates observations about applications, and tries to learn user preferences and to choose proper actions. We define Γ the set of feasible control algorithms, i.e., algorithms that only choose feasible control actions $\boldsymbol{\mu}(t) \in \mathcal{U}_{\boldsymbol{S}(t)}$ in every time slot, possibly with help from external information sources regarding application demand statistics. We also introduce a rate of cost expenditure constraint $\rho \in (0, \rho_{\max}]$, where $\rho_{\max} \triangleq \sum_k \pi_k \sum_m \sum C_m(1, s_k)$ is the maximum budget needed to achieve the highest level of intelligence, which corresponds to the case of always pre-serving user demand. Despite a poor resource utilization, all demands will be pre-served and $I(\rho_{\max}) = \sum_m \frac{\epsilon_m r_{mp}}{\epsilon_m + \delta_m}$ where $\frac{\epsilon_m}{\epsilon_m + \delta_m}$ is the steady-state distribution of having $A_m(t) = 1$. Then, for each policy $\Pi \in \Gamma$, we denote $I(\Pi, \rho) = r_{\text{av}}(\Pi)$ and $C_{\text{av}}(\Pi)$ the resulting algorithm intelligence and average cost rate, respectively.

The objective of the system is to achieve the *system intelligence* $I(\rho)$, defined to be the maximum reward rate r_{av} achievable over all feasible policies, subject to the rate of cost expenditure being no more than ρ, i.e.,

$$I(\rho) \triangleq \max: \ I(\Pi, \rho) \quad (6)$$

$$\text{s.t. } C_{\text{av}}(\Pi) \leq \rho$$

$$\Pi \in \Gamma. \quad (7)$$

E. Discussions of the Model

In our model, we have assumed that user demand must be served within the same slot it is placed. This is a suitable model for many task management systems where jobs are time-sensitive, e.g., newsfeed pushing, realtime computation, elevator scheduling, video streaming and searching. In these problems, a user's perception about system smartness is often based on whether jobs are pre-served correctly.

Our model captures key ingredients of a general smart system including observations, learning and prediction, and control. Indeed, general monitoring and sensing methods can be integrated into the observation part, various learning methods can be incorporated into our learning-prediction step, and control algorithms can be combined with or replace our control scheme presented later. On the other hand, due to the facts that system reward can only be accumulated when demand is proactively served, and that the system dynamics are governed by unknown Markov processes, existing results on average utility optimization cannot be directly applied.[5] Below, for reader convenience, we summarize the notations in the paper in Table I.

[3]The i.i.d. assumption is commonly made in the literature to facilitate presentation and analysis, e.g., [32], [33]. Allowing different states to have different action sets is to make the model more general.

[4]This can be done by monitoring user feedbacks, e.g., display a short message and ask the user to provide instantaneous feedback. In the case when they are not known a-priori, they can be learned via a similar procedure as in the LBISC algorithm presented later.

[5]In some cases, users may also try to behave in a way that the Markov chain transitions are chosen in an adversarial manner, e.g., to protect privacy. This is a very interesting aspect not captured in the current model, and will be a subject of our future work.

TABLE I
TABLE OF NOTATIONS

Notation	Meaning
$A_m(t)$	Demand state of user m
$\mu_{mc}(t)$	Serve the demand at t
$\mu_{mc}(t)$	Serve the demand at $t+1$
$S_m(t)$	Resource state of application m
$C_m(t)$	Cost for serving application m at t
$\overline{C}_m$	Expected cost for serving one unit demand
$r_m(t)$	Reward from application m
ρ	Resource budget
$a_m^{(i_h)}$	Transition rate into $A_m(t)=1$ from i_h
$\tilde{r}_m^{(i)}(t)$ and $\tilde{r}_m^{(i)}(t)$	Effective reward and effective cost
$d(t)$	Deficit queue
$N(t)$	Number of useful samples
$\hat{\epsilon}_m(T)$ and $\hat{\delta}_m(T)$	Estimated transition rates
$g_{\boldsymbol{\pi}}(\boldsymbol{\gamma})$ and $\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma)$	Dual function and approximation with $\hat{\boldsymbol{\epsilon}}$, $\hat{\boldsymbol{\delta}}$, and $\hat{\boldsymbol{\pi}}$

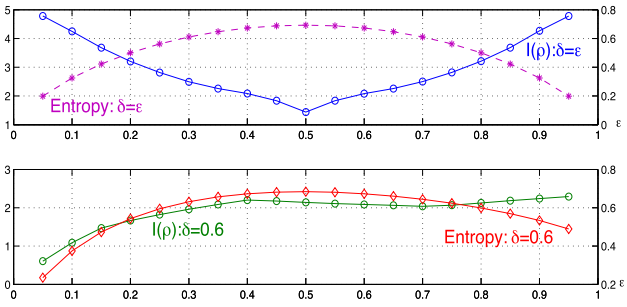

Fig. 2. System intelligence ($M=1$): The left y-axis is for $I(\rho)$ and the right y-axis is for the entropy rate. The x-axis shows the value of ϵ. We see that $I(\rho)$ is consistent with our finding that one can achieve higher intelligence levels for more predictable systems.

IV. CHARACTERIZING INTELLIGENCE

In this section, we first obtain a characterization of $I(\rho)$. This result provides interesting insight into system intelligence, and provides a useful criteria for evaluating smartness of control algorithms.

To this end, denote $z(t) = (\boldsymbol{A}(t), \boldsymbol{S}(t))$ and $\mathcal{Z} = \{z_1, \ldots, z_H\}$ the state space of $z(t)$, and denote π_h the steady-state distribution of z_h. Furthermore, define for notation simplicity the transition probability $a_m^{(i_h)}$ as:

$$a_m^{(i_h)} = \begin{cases} 1-\delta_m & \text{when } i_h = A_m^{(h)} = 1 \\ \epsilon_m & \text{when } i_h = A_m^{(h)} = 0 \end{cases} \tag{8}$$

That is, $a_m^{(i)}$ denotes the probability of having $A_m(t+1)=1$ in the next time slot given the current state being i. Then, our theorem is as follows.

Theorem 1: $I(\rho)$ is equal to the optimal value of the following optimization problem[6]*:*

$$\max: \sum_h \pi_h \sum_{j=1}^{3} \theta_j^{(h)} \sum_m a_m^{(i_h)} [\mu_{mpj}^{(h)} r_{mp} + (1-\mu_{mpj}^{(h)}) r_{mc}] \tag{9}$$

$$\begin{aligned} s.t. \ & \sum_h \pi_h \sum_{j=1}^{3} \theta_j^{(h)} \sum_m [C_m(\mu_{mpj}^{(h)}, s(z_h)) \\ & \qquad + (1-\mu_{mpj}^{(h)}) a_m^{(i_h)} \overline{C}_m] \le \rho \\ & \theta_j^{(h)} \ge 0, \quad \sum_j \theta_j^{(h)} = 1, \ \forall z_h, j \\ & \boldsymbol{\mu}_j^{(h)} \in \mathcal{U}_{z_h}, \quad \forall z_h, j. \end{aligned} \tag{10}$$

Here $\theta_j^{(h)}$ represents the probability of adopting the pre-service vector $\boldsymbol{\mu}_j^{(h)}$ under state z_h, and $s(z_h)$ is the channel state under z_h.

Proof: See Appendix A. □

Theorem 1 states that no matter what learning and prediction methods are adopted and however the system is controlled, the intelligence level perceived by users will not exceed the value in (9). This is a powerful result and provides a *fundamental limit* about the system intelligence. Theorem 1 also reveals some interesting facts and rigorously justify various common beliefs about system intelligence. (i) When user demands are more predictable (captured by transition rates ϵ_m and δ_m, represented by $a_m^{(i_h)}$ in (9)), the system can achieve a higher intelligence level. (ii) A system with more resources (larger ρ) or better cost management (smaller C_m functions) can likely achieve a higher level of perceived smartness. (iii) When there is more demand from users (captured by distribution π_h), there are more opportunities for the system to impress the user, and to increase the perceived smartness level. The inclusion of transition rates in the theorem shows that our problem can be very different from existing network optimization problems, e.g., [34], [35], where typically steady-state distributions matter most.

As a concrete example, Fig. 2 shows the $I(\rho)$ values for a single-application system with $r_p = 10$, $r_c = 1$ and $\rho = 0.8$. The channel has two states $S(t)=1$ and $S(t)=2$ with equal probabilities, and the cost is $S(t)$. We examine two cases, (i) $\epsilon = \delta$ and (ii) $\delta = 0.6$. We see that in the symmetric case, where the steady-state distribution is always $(0.5, 0.5)$, $I(\rho)$ is inverse-proportional to the entropy rate of the demand Markov chain, which is consistent with our finding that a higher intelligence level is achievable for more predictable systems (lower entropy). For the $\delta = 0.6$ case, $I(\rho)$ first increases, then decreases, and then increases again. The reason is as follows. At the beginning, as ϵ increases, demand increases. Then, when $\epsilon \in [0.4, 0.7]$, $I(\rho)$ is reduced by either the increasing randomness (less predictable) or budget constraint, as demand increases, which reduces the pre-service attempts that can potentially be risky. As a result, the intelligence level also decreases. After that, predictability increases and $I(\rho)$ increases again. This shows that $I(\rho)$ is jointly determined by the steady-state distribution and the transition rates.

Note that solving problem (9) is non-trivial due to the need of system statistics and the potentially complicated structure of $\mathcal{U}_{z_h}$. Thus, in the next section, we propose a learning-based algorithm for solving the optimal control problem. For our algorithm design and analysis, we define the following modified dual function of (9), where $V \ge 1$ is a control

[6]Here the index $j = 1, 2, 3$ is a result of the Caratheodory's theorem (see Appendix A).

parameter introduced for later use[7]:

$$g_{\boldsymbol{\pi}}(\gamma) \triangleq \sum_h \pi_h \sup_{\boldsymbol{\mu}_p^{(h)}} \sum_m \Big\{ V a_m^{(i_h)} [\mu_{mp}^{(h)} r_{mp} + (1-\mu_{mp}^{(h)}) r_{mc}] - \gamma [C_m(\mu_{mp}^{(h)}, z_h) + (1-\mu_{mp}^{(h)}) a_m^{(i_h)} \overline{C}_m - \rho] \Big\}. \tag{11}$$

Here we use the subscript $\boldsymbol{\pi}$ to denote that the dual function is defined with distribution $\boldsymbol{\pi}$. We also use γ^* to denote a minimizer of the dual function. It is shown in [23] that:

$$g_{\boldsymbol{\pi}}(\gamma^*) \geq V \times I(\rho). \tag{12}$$

We also define the dual function for each state z_h as follows:

$$g_h(\gamma) \triangleq \sup_{\boldsymbol{\mu}_p^{(h)}} \sum_m \Big\{ V a_m^{(i_h)} [\mu_{mp}^{(h)} r_{mp} + (1-\mu_{mp}^{(h)}) r_{mc}] - \gamma [C_m(\mu_{mp}^{(h)}, z_h) + (1-\mu_{mp}^{(h)}) a_m^{(i_h)} \overline{C}_m - \rho] \Big\}. \tag{13}$$

It can be seen that $g_{\boldsymbol{\pi}}(\gamma) = \sum_h \pi_h g_h(\gamma)$.

V. Algorithm Design

In this section, we present an online learning-aided control algorithm for achieving maximum system intelligence. To facilitate understanding, we first present an ideal algorithm that assumes full information of ϵ_m, δ_m, and π_h. It serves as a building block for our actual algorithm.

A. An Ideal Algorithm

To do so, we first define the *effective* reward and cost for each application m as functions of $\boldsymbol{A}(t)$ and $\boldsymbol{\mu}(t)$. Specifically, if $A_m(t) = i$, we have: :

$$\tilde{r}_m^{(i)}(\mu_{mp}(t)) = \begin{cases} a_m^{(i)} r_{mp} & \text{if } \mu_{mp}(t) = 1 \\ a_m^{(i)} r_{mc} & \text{if } \mu_{mp}(t) = 0 \end{cases} \tag{14}$$

Here $a_m^{(i)}$ is defined in (8) as the probability of having state $A_m(t+1) = 1$ conditioning on the current state i. To understand the definition, we see that when $A_m(t) = i$, by taking $\mu_{mp}(t) = 0$, the server does not pre-serve the potential future demand at $A_m(t+1)$. Hence, if there is demand in slot $t+1$ (happens with probability $a_m^{(i)}$), it will be served by $\mu_{mc}(t+1)$, resulting in a reward of r_{mc}. On the other hand, if $\mu_{mp}(t) = 1$, with probability $a_m^{(i)}$, the future demand will be pre-served and a reward r_{mp} can be collected. It is important to note that the effective reward is defined to be the reward collected in slot $t+1$ as a result of actions at time t. We denote $\tilde{r}(\boldsymbol{\mu}(t)) \triangleq \sum_m \tilde{r}_m^{(A_m(t))}(\mu_{mp}(t))$.

Similarly, we define the *effective* cost as a function of $\boldsymbol{\mu}(t)$ for $A_m(t) = i$:

$$\tilde{C}_m^{(i)}(\mu_{mp}(t)) = \begin{cases} C_m(1, \boldsymbol{S}(t)) & \text{if } \mu_{mp}(t) = 1 \\ a_m^{(i)} \sum_k \pi_k C_m(1, s_k) & \text{if } \mu_{mp}(t) = 0 \end{cases} \tag{15}$$

[7]Although (11) does not include $\theta_j^{(h)}$, it can be shown to be equivalent. Moreover, (11) is sufficient for our algorithm design and analysis.

Note that $\tilde{C}_m(\mu_{mp}(t))$ is the expected cost spent in slots t and $t+1$. As in the effective reward case, we denote $\tilde{C}(t) \triangleq \sum_m \tilde{C}_m^{(A_m(t))}(\mu_{mp}(t))$.

With the above definitions, we introduce a *deficit* queue $d(t)$ that evolves as follows:

$$d(t+1) = \max[d(t) + \tilde{C}(t) - \rho, 0], \tag{16}$$

with $d(0) = 0$. Note that this deficit is not the actual deficit, instead, it is defined with the expected cost incurred in slots where no pre-service takes place. Nonetheless, it can be shown that the stability of $d(t)$ implies stability of the actual deficit. Next, we define a Lyapunov function $L(t) \triangleq \frac{1}{2} d^2(t)$ and define a single-slot sample-path drift $\Delta(t) \triangleq L(t+1) - L(t)$. By squaring both sides of (16), using $(\max[x,0])^2 \leq x^2$ for all $x \in \mathbb{R}$, and $\tilde{C}(\boldsymbol{\mu}(t)) \leq M C_{\max}$, we obtain the following inequality:

$$\Delta(t) \leq B - d(t)[\rho - \tilde{C}(t)]. \tag{17}$$

Here $B \triangleq \rho_{\max}^2 + M^2 C_{\max}^2$. Adding to both sides the term $V \sum_m \tilde{r}_m(\mu_{mp}(t))$, where $V \geq 1$ is a control parameter, we obtain:

$$\Delta(t) - V\tilde{r}(\boldsymbol{\mu}(t)) \leq B - \Big(V\tilde{r}(\boldsymbol{\mu}(t)) + d(t)[\rho - \tilde{C}(\boldsymbol{\mu}(t))] \Big). \tag{18}$$

Having established (18), we construct the following ideal algorithm by choosing pre-service actions to minimize the right-hand-side of the drift.

*Budget-Limited Intelligent System Control (*BISC*):* At every time t, observe $\boldsymbol{A}(t)$, $\boldsymbol{S}(t)$ and $d(t)$. Do:

- For each m, define the cost-differential as follows:

$$D_m(t) \triangleq C_m(1, \boldsymbol{S}(t)) - a_m^{(i)} \overline{C}_m. \tag{19}$$

  Here $i = A_m(t)$. Then, solve the following problem to find the optimal pre-service action $\boldsymbol{\mu}(t)$:

$$\begin{aligned} \max: \ & \sum_m \mu_{mp}(t)[V a_m^{(i)}(r_{mp} - r_{mc}) - d(t) D_m(t)] \\ \text{s.t. } & \boldsymbol{\mu}(t) \in \mathcal{U}_{\boldsymbol{S}(t)}. \end{aligned} \tag{20}$$

- Update $d(t)$ according to (16). $\Diamond$

A few remarks are in place. (i) The value $a_m^{(i)}(r_{mp} - r_{mc})$ can be viewed as the expected reward loss if we choose $\mu_{mp}(t) = 0$ and the value $D_m(t)$ is the expected cost saving for doing so. The parameter V and $d(t)$ provide proper weights to the terms for striking a balance between them. If the cost saving does not overweight the reward loss, it is more desirable to pre-serve the demand in the current slot. (ii) For applications where $a_m^{(i)}(r_{mp} - r_{mc})$ is smaller, it is less desirable to pre-serve the demand, as the user perception of system intelligence may not be heavily affected. (iii) In the special case when $\rho \geq \rho_{\max}$, we see that $d(t)$ will always stay near zero, resulting in $\mu_{mp}(t) = 1$ most of the time. (iv) BISC is easy to implement. Since each $\mu_{mp}(t)$ is either 0 or 1, problem (20) is indeed finding the maximum-weighted vector from $\mathcal{U}_{\boldsymbol{S}(t)}$. In the case when $\mathcal{U}_{\boldsymbol{S}(t)}$ only limits the number of non-zero entries, we can sort the applications according to

the value $Va_m^{(i)}(r_{mp}-r_{mc})-d(t)D_m(t)$ and choose the top ones.

For readers who are familiar with the drift-plus-penalty literature, e.g., [36], you see that BISC is indeed a drift-plus-penalty algorithm. However, due to the Markov dynamics of the system and that transition rates are unknown, a learning component must be incorporated into algorithm design properly. This requires a different analysis from existing results.

B. Learning-Aided Algorithm With User Population Effect

In this section, we present an algorithm that learns δ_m and ϵ_m online and performs optimal control simultaneously. We also explicitly investigate how the system user population size affects algorithm performance.

To rigorously quantify the user effect, we first introduce the following *user-population effect* function $N(T)$.

Definition 1: A system is said to have a user population effect $N(T)$ if within T slots, (i) it collects a sequence of demand samples $\{\boldsymbol{A}(0),\ldots,\boldsymbol{A}(N(T)-1)\}$ generated by the Markov process $\boldsymbol{A}(t)$, and (ii) $N(T)\geq T$ for all T. $\Diamond$

$N(t)$ captures the number of useful user demand samples a system can collect in t time slots, and is an indicator of how user population contributes to learning user preferences. For instance, if there is only one user, $N(T)=T$. On the other hand, if a system has many users, it can often collect samples from similar users (often determined via machine learning techniques, e.g., clustering) to study a target user's preferences. One typical example for $N(T)$ can be:

$$N(T)=f(\#\text{ of user})\cdot T, \tag{21}$$

where $f(\#\text{ of user})$ computes the number of similar users that generate useful samples.

Now we present our algorithm. We begin with the first component, which is a maximum likelihood estimator (MLE) [37] for estimating user demand statistics.[8]

Maximum Likelihood Estimator (MLE(T)): Fix a learning time T and obtain $\{\boldsymbol{A}(0),\ldots,\boldsymbol{A}(N(T)-1)\}$ in $[0,T-1]$. Output:

$$\hat{\epsilon}_m(T)=\frac{\sum_{t=0}^{N(T)-1}1_{\{A_m(t)=0,A_m(t+1)=1\}}}{\sum_{t=0}^{T-1}1_{\{A_m(t)=0\}}} \tag{22}$$

$$\hat{\delta}_m(T)=\frac{\sum_{t=0}^{N(T)-1}1_{\{A_m(t)=1,A_m(t+1)=0\}}}{\sum_{t=0}^{T-1}1_{\{A_m(t)=1\}}}. \tag{23}$$

That is, use empirical frequencies to estimate the transition probabilities. $\Diamond$

Note that after estimating $\hat{\boldsymbol{\epsilon}}$ and $\hat{\boldsymbol{\delta}}$, we also obtain an estimation of $\hat{\boldsymbol{\pi}}$. We now have the second component, which is a *dual learning* module [22] that learns an empirical Lagrange multiplier based on $\hat{\boldsymbol{\epsilon}}$ and $\hat{\boldsymbol{\delta}}$, and $\hat{\boldsymbol{\pi}}$.

Dual Learning (DL($\hat{\boldsymbol{\epsilon}}$, $\hat{\boldsymbol{\delta}}$, $\hat{\boldsymbol{\pi}}$)): Construct $\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma)$ with $\hat{\boldsymbol{\epsilon}}$, $\hat{\boldsymbol{\delta}}$, and $\hat{\boldsymbol{\pi}}$ according to (11). Solve the following problem and output the optimal solution γ_T^*.

$$\min:\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma),\quad \text{s.t. } \gamma\geq 0.\ \Diamond \tag{24}$$

[8]We adopt MLE to demonstrate how learning can be rigorously and efficiently combined with control algorithms to achieve good performance. Alternative estimators that possess similar features as MLE can also be used.

Here $\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma)$ is the dual function with true statistics being replaced by $\hat{\boldsymbol{\epsilon}}$, $\hat{\boldsymbol{\delta}}$, and $\hat{\boldsymbol{\pi}}$. Since the dual problem is always concave in the dual variable, (24) can always be solved efficiently be standard optimization algorithms, e.g., dual subgradient. However, similar to the classic NUM problems, whether or not it can be solved in a distributed manner depends heavily on properties of the underlying physical system. Specifically, for systems where action sets are separable, e.g., rate allocation in wired networks, distributed algorithms can be designed to solve the dual problem, whereas if the action sets are not-separable, e.g., power allocation with interference, NUM problems do not admit exact distributed algorithms, and neither does our problem.

With MLE(T) and DL($\hat{\boldsymbol{\epsilon}}$, $\hat{\boldsymbol{\delta}}$, $\hat{\boldsymbol{\pi}}$), we have our learning-aided BISC algorithm.[9]

Learning-Aided BISC (LBISC(T, θ)): Fix a learning time T and perform the following[10]:

- (Estimation) For $t=0,\ldots,T-1$, choose any $\boldsymbol{\mu}_p(t)\in\mathcal{U}_{S(t)}$. At time T, perform MLE(T) to obtain $\hat{\boldsymbol{\epsilon}}$ and $\hat{\boldsymbol{\delta}}$, and $\hat{\boldsymbol{\pi}}$.
- (Learning) At time T, apply DL($\hat{\boldsymbol{\epsilon}}$, $\hat{\boldsymbol{\delta}}$, $\hat{\boldsymbol{\pi}}$) and compute γ_T^*. If $\gamma_T^*=\infty$, set $\gamma_T^*=V\log(V)$. Reset $d(T)=0$.
- (Control) For $t\geq T$, run BISC with $\hat{\boldsymbol{\pi}}$, $\hat{\boldsymbol{\epsilon}}$ and $\hat{\boldsymbol{\delta}}$, and with effective queue size $\tilde{d}(t)=d(t)+(\gamma_T^*-\theta)^+$. $\Diamond$

Here θ (to be specified) is a tuning parameter introduced to compensate for the error in γ_T^* (with respect to γ^*).[11] It is interesting to note that LBISC includes three important functions in control, namely, estimation (data), learning (training) and control (algorithm execution). This structure highlights three major sources that contribute to making a system non-intelligent: lack of data samples, incorrect training and parameter tuning, and inefficient control algorithms. An intelligent system requires all three to provide good user experience and to be considered smart (Thus, if a search engine does not provide good performance for you at the beginning, it may not be because its algorithm is bad).

Our approach can be viewed as trying to combine the offline optimization approach that makes decisions purely based on system statistics, and the queue-based control approach that control the system based on Lyapunov optimization. Doing so not only enables rigorous analysis of the resulting algorithm, but also provides tighter budget management. These advantages are hard to obtain by either approach alone.

VI. Performance Analysis

In this section, we analyze the performance of LBISC. We focus on three important performance metrics, i.e., achieved system intelligence, budget guarantee, and algorithm convergence time. The optimality and convergence analysis is challenging. In particular, the accuracy of the

[9]The methodology can be applied to the case when r_{mp} and r_{mc} are also unknown.

[10]The main reason to adopt a finite T is for tractability. In actual implementation, one can continuously refine the estimates for $\hat{\boldsymbol{\epsilon}}$, $\hat{\boldsymbol{\delta}}$, and $\hat{\boldsymbol{\pi}}$.

[11]Due to estimation error, it is possible that $\gamma_T^*>\gamma^*$. Thus, if θ is not introduced, it can happen that $\gamma_T^*+d(t)>\gamma^*$ for all time (since $d(t)\geq 0$). If this happens, the system will always make conservative actions, resulting in a poor system performance.

MLE estimator affects the quality of dual-learning, which in turn affects algorithm convergence and performance. Thus, the analysis must simultaneously take into account all three components.

Throughout our analysis, we make the following assumptions, in which we use $\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma)$ to denote the dual function in (11) with different $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}$ and $\boldsymbol{\pi}$ values.

Assumption 1: There exists a constant $\nu = \Theta(1) > 0$ such that for any valid state distribution $\boldsymbol{\pi}' = (\pi'_1, \ldots, \pi'_H)$ with $\|\boldsymbol{\pi}' - \boldsymbol{\pi}\| \leq \nu$, there exist a set of actions $\{\boldsymbol{\mu}_j^{(h)}\}_{j=1,2,3}$ with $\boldsymbol{\mu}_j^{(h)} \in \mathcal{U}_{z_h}$ and some variables $\{\theta_j^{(h)} \geq 0\}_{j=1,2,3}$ with $\sum_j \theta_j^{(h)} = 1$ for all z_h (possibly depending on $\boldsymbol{\pi}'$), such that:

$$\sum_h \pi'_h \sum_{j=1}^{3} \theta_j^{(h)} \sum_m [C_m(\mu_{mpj}^{(h)}, z_h) + (1 - \mu_{mpj}^{(h)}) a_m^{(i_h)} \overline{C}_m] \leq \rho_0,$$

where $\rho_0 \triangleq \rho - \eta > 0$ with $\eta = \Theta(1) > 0$ is independent of $\boldsymbol{\pi}'$. Moreover, for all transition probabilities $\boldsymbol{\epsilon}'$ and $\boldsymbol{\delta}'$ with $\|\boldsymbol{\epsilon}' - \boldsymbol{\epsilon}\| \leq \nu$ and $\|\boldsymbol{\delta}' - \boldsymbol{\delta}\| \leq \nu$, ρ satisfies:

$$\rho \geq \sum_m \max[\epsilon'_m \overline{C}_m, (1 - \delta'_m)\overline{C}_m]. \ \diamondsuit \tag{25}$$

Assumption 2: There exists a constant $\nu = \Theta(1) > 0$ such that, for any valid state distribution $\boldsymbol{\pi}' = (\pi'_1, \ldots, \pi'_H)$ with $\|\boldsymbol{\pi}' - \boldsymbol{\pi}\| \leq \nu$, and transition probabilities $\boldsymbol{\epsilon}'$ and $\boldsymbol{\delta}'$ with $\|\boldsymbol{\epsilon}' - \boldsymbol{\epsilon}\| \leq \nu$ and $\|\boldsymbol{\delta}' - \boldsymbol{\delta}\| \leq \nu$, $\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma)$ has a unique optimal solution $\gamma^* > \mathbf{0}$ in $\mathbb{R}$. $\diamondsuit$

These two assumptions are standard in the network optimization literature, e.g., [32], [33]. They are necessary conditions to guarantee the budget constraint and are often assumed with $\nu = 0$. In our case, having $\nu > 0$ means that systems that are alike have similar properties. (25) is also not restrictive. In fact, $\sum_m [\pi_{m0} \epsilon'_m \overline{C}_m + \pi_{m1}(1 - \delta'_m)\overline{C}_m]$ (π_{mi} is the steady-state probability of being in state i for m) is the overall cost without any pre-service. Hence, (25) is close to being a necessary condition for feasibility.

We now have the third assumption, which is related to the structure of the problem. To state it, we have the following system structural property introduced in [13].

Definition 2: A system is polyhedral with parameter $\beta > 0$ under distribution $\boldsymbol{\pi}$ if the dual function $g_{\boldsymbol{\pi}}(\gamma)$ satisfies:

$$g_{\boldsymbol{\pi}}(\gamma) \geq g_{\boldsymbol{\pi}}(\gamma^*) + \beta \|\gamma^* - \gamma\|. \ \diamondsuit \tag{26}$$

Assumption 3: There exists a constant $\nu = \Theta(1) > 0$ such that, for any valid state distribution $\boldsymbol{\pi}' = (\pi'_1, \ldots, \pi'_H)$ with $\|\boldsymbol{\pi}' - \boldsymbol{\pi}\| \leq \nu$, and transition probabilities $\boldsymbol{\epsilon}'$ and $\boldsymbol{\delta}'$ with $\|\boldsymbol{\epsilon}' - \boldsymbol{\epsilon}\| \leq \nu$ and $\|\boldsymbol{\delta}' - \boldsymbol{\delta}\| \leq \nu$, $\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma)$ is polyhedral with the same β. $\diamondsuit$

The polyhedral property often holds for practical systems, especially when control action sets are finite (see [13] for more discussions).

A. System Intelligence and Budget

We first present the performance of LBISC in system intelligence and budget guarantee. The following theorem summarizes the results.

Theorem 2: Suppose the system is polyhedral with $\beta = \Theta(1) > 0$. By choosing $\theta = \max(\frac{V \log(V)^2}{\sqrt{N(T)}}, \log(V)^2)$ and a sufficiently large V, with probability at least $1 - 2Me^{-\log(V)^2/4}$, LBISC achieves:

- *Budget:*

$$\tilde{d}(t) \leq d_{\max} \triangleq Vr_d/\hat{D}_{\min} + MC_{\max}, \quad \forall t \tag{27}$$

$$\overline{d} = O(\max(\frac{V \log(V)^2}{\sqrt{N(T)}}, \log(V)^2)). \tag{28}$$

  Here $\overline{d} = \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{d(t)\}$ and $\hat{D}_{\min} = \Theta(1)$ (defined in (36)). (27) implies $C_{av}(\Pi) \leq \rho$.

- *System intelligence:*

$$I(\texttt{LBISC}, \rho) \geq I(\rho) - \frac{B_1 + 1}{V} - \max_{z_h} \frac{e_{\max}(\overline{T}_h^2 - \overline{T}_h)}{2V\overline{T}_h}. \tag{29}$$

  Here $B_1 \triangleq B + 2M(Vr_d + d_{\max}C_{\max}) \log(V)/\sqrt{N(T)}$, $e_{\max} \triangleq (MC_{\max} + \rho)^2$, and $\overline{T}_h$ and $\overline{T}_h^2$ are the first and second moments of return times of state z_h. $\diamondsuit$

Proof: See Appendix B. □

Theorem 2 shows that LBISC achieves an $[O(N(T)^{-\frac{1}{2}} + \epsilon), O(\max(N(T)^{-\frac{1}{2}} \log(1/\epsilon)^2/\epsilon, \log(1/\epsilon)^2)]$ intelligence-budget tradeoff (taking $\epsilon = 1/V$), and the system intelligence level can be pushed arbitrarily close to $I(\rho)$ under LBISC. Thus, by varying the value of V, one can tradeoff the intelligence level loss and budget deficit as needed. Although Theorem 2 appears similar to previous results with learning, e.g., [22], its analysis is different due to (i) the Markov nature of the demand state $\boldsymbol{A}(t)$, and (ii) the learning error in both transition rates in (20) and the distribution. Finally, note that since our Markov model includes the i.i.d. setting as a special case, the optimality of the $[O(\epsilon), O(\log(1/\epsilon))]$ tradeoff derived in [12] also holds as a lower bound for our setting. This further confirms that LBISC achieves a good performance, especially when $N(T)$ is large.

B. Convergence Time

We now look at the convergence speed of LBISC, which measures how fast the algorithm learns the desired system operating point. This is an important metric for dynamical systems, as a faster convergence implies both a faster matching between demand and resource allocation, and a better algorithm robustness against the changing environment. Indeed, how to improve algorithm convergence time has recently received an increasing attention in the literature, e.g., [38]–[40].

To present our result, we adopt the following definition of convergence time from [22] to our setting.

Definition 3: Let $\zeta > 0$ be a given constant. The ζ-convergence time of a control algorithm, denoted by T_ζ, is the time it takes for $\tilde{d}(t)$ to get to within ζ distance of γ^*, i.e., $T_\zeta \triangleq \inf\{t : |\tilde{d}(t) - \gamma^*| \leq \zeta\}$. $\diamondsuit$

The intuition behind Definition 3 is as follows. Since the LBISC algorithm is a queue-based algorithm, the algorithm will starting making optimal choice of actions once $\tilde{d}(t)$ gets

close to γ^*. Hence, T_ζ naturally captures the time it takes for LBISC to converge. For comparison, we also analyze the convergence time of BISC. The slower convergence speed of BISC is due to the fact that it does not utilize system information to perform learning-aided control. It is important to note that BISC assumes full statistical information beforehand, whereas LBISC learns everything online.

Theorem 3: Suppose the conditions in Theorem 2 hold. Under LBISC*, with probability at least* $1-2Me^{-\log(V)^2/4}$,

$$\mathbb{E}\{T_{D_1}^{\texttt{LBISC}}\} = O(\max(\frac{V\log(V)^2}{\sqrt{N(T)}}, \log(V)^2)) + T, \quad (30)$$

$$\mathbb{E}\{T_{D_1}^{\texttt{BISC}}\} = O(V - \max(\frac{V\log(V)^2}{\sqrt{N(T)}}, \log(V)^2)), \quad (31)$$

$$\mathbb{E}\{T_{D_2}^{\texttt{BISC}}\} = \Theta(V). \quad (32)$$

Here $D_1 = O(V\log(V)/\sqrt{N(T)} + D)$ *with* $D = \Theta(1)$ *and* $D_2 = \Theta(1)$.

Proof: See Appendix C. □

Here the reason why D_1 may be larger than D_2 is due to the fact that LBISC uses inaccurate estimates of $a_m^{(i)}$ for making decisions. In the case when $N(T) = T$, we can recover the $O(V^{2/3})$ convergence time results in [22] by choosing $T = V^{2/3}$, whereas BISC requires a time that is $O(V)$. Theorem 3 also shows that it is possible to achieve faster convergence if a system has a larger population of users from which it can collect useful samples for learning the target user quickly. It also explicitly quantifies the speedup factor to be proportional to the *square-root* of the user population size, i.e., when $N(T) = N * T$ where N is the number of users, the speedup factor is $\sqrt{N(T)}/\sqrt{T} = \sqrt{N}$.

This result reveals the interesting fact that a big company with many users naturally has advantage over companies with smaller user populations, since they can collect more useful data and adapt to a "smart" state faster. Also note that although we consider a stationary system in this paper, the results here can serve as an building block for the heterogeneous case where dynamics are non-stationary. Indeed, using a similar design approach as in [41], we can likely extend our results to the case where the underlying distribution changes over time.

VII. Simulation

We now present simulation results for BISC and LBISC. We simulate a three-application system ($M = 3$) with the following setting. $(r_{1p}, r_{2p}, r_{3p}) = (3, 5, 8)$ and $r_{mc} = 1$ for all m. Then, we use $\boldsymbol{\epsilon} = (0.6, 0.5, 0.3)$ and $\boldsymbol{\delta} = (0.2, 0.6, 0.5)$. The channel state space is $\mathcal{S} = \{1, 2\}$ for all m, with $\Pr\{S_1(t) = 1\} = 0.5$, $\Pr\{S_2(t) = 1\} = 0.3$, and $\Pr\{S_3(t) = 1\} = 0.3$. The service cost is given by $C_m(1, \boldsymbol{S}(t)) = S_m(t)$. We simulate the system for $T_{sim} = 10^5$ slots, with $V = \{5, 10, 20, 50, 100\}$. For LBISC, we simulate a user population effect function as in (21), i.e., $N(T) = f(\#\text{ of user}) \cdot T$ and choose $f(\#\text{ of user}) = 2, 5, 8$. We also fix the value $\rho = 3.5$ and choose the learning time $T = V^{2/3}$.

We first present Fig. 3 that shows $I(\rho)$ as a function of ρ. For comparison, we include a second setting, where we change $(r_{1p}, r_{2p}, r_{3p}) = (4, 5, 3)$, $\boldsymbol{\epsilon} = (0.8, 0.4, 0.3)$, $\boldsymbol{\delta} = (0.2, 0.9, 0.5)$, and $\Pr\{S_2(t) = 1\} = 0.8$. It can be seen that $I(\rho)$ first increases as ρ increases. Eventually ρ becomes more than needed after all the possible predictability has been exploited. Then, $I(\rho)$ becomes flat. This *diminishing return* property is consistent with our understanding obtained in Section IV.

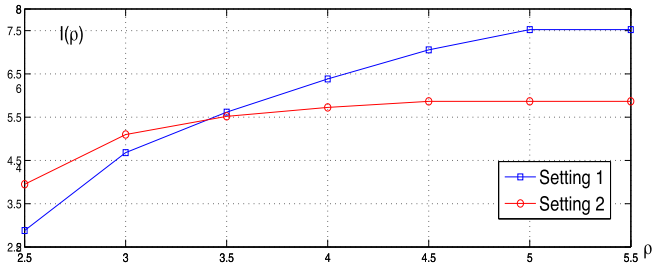

Fig. 3. $I(\rho)$ versus ρ: in the two settings tested, $I(\rho)$ are both concave increasing in ρ.

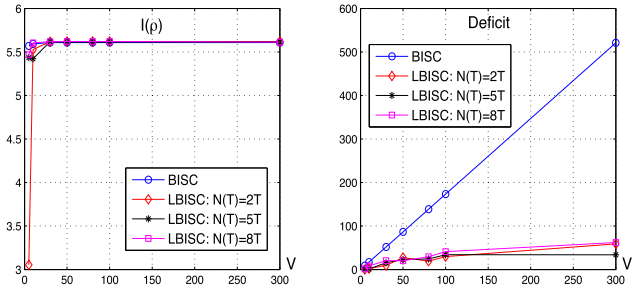

Fig. 4. Intelligence and deficit performance of BISC and LBISC with different user-population effect.

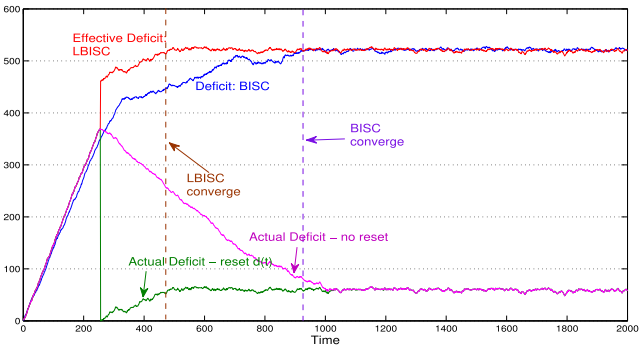

Fig. 5. Convergence of BISC and LBISC with $N(T) = 8T$ for $V = 300$.

We then look at algorithm performance. From Fig. 4 we see that both BISC and LBISC are able to achieve high intelligence levels. Moreover, LBISC does much better in controlling the deficit (2×-4× saving compared to BISC). Note that we are plotting the real deficit incurred instead of the expected value. Also, the actual deficit measures the steady-state deficit under LBISC. To demonstrate this, we include in Fig. 5 the deficit incurred if $d(t)$ is not reset to zero. We see that it eventually converges to the actual deficit value. From the plot, we also see that the reset step is very useful for accelerating algorithm convergence. We remark here that BISC assumes full knowledge beforehand, while LBISC learns them online.

Finally, we look at algorithm convergence. Fig. 5 compares BISC and LBISC with $N(t) = 8T$ and $V = 300$. We see that LBISC converges at around 460 slots, whereas BISC converges at around 920 slots, resulting in a $2\times$ improvement. Moreover, the actual deficit level under LBISC is much smaller compared to that under BISC (80 versus 510, a $6\times$ improvement). From this result, we see that it is important to efficiently utilize data collected over time, and dual learning provides one way to boost algorithm convergence. In both Fig. 4 and Fig. 5, the learning period does not heavily affect performance because its length is short compared to the simulation time.

VIII. Conclusion

In this paper, we present a general framework for defining and understanding system intelligence, and propose a novel metric for measuring the smartness of dynamical systems, defined to be the maximum average reward rate obtained by proactively serving user demand subject to a resource constraint. We show that the highest system intelligence level is jointly determined by system resource, action costs, user demand volume, and correlation among demands. We then develop a learning-aided algorithm called Learning-aided Budget-limited Intelligent System Control (LBISC), which efficiently utilizes data samples of system dynamics and achieves a near-optimal intelligence, and guarantees a deterministic deficit bound. Moreover, LBISC converges much faster compared to its non-learning based counterpart.

Appendix A
Proof of Theorem 1

We prove Theorem 1 here using an argument similar to that in [10].

Proof (Theorem 1): Consider any control scheme Π and fix a time T. Define the joint state $z(t) = (\boldsymbol{A}(t), \boldsymbol{S}(t))$ and denote its state space as $\mathcal{Z} = \{z_1, \ldots, z_H\}$. Then, consider a state z_h and let $\mathcal{T}_h(T)$ be the set of slots with $z(t) = z_h$ for $t = 1, \ldots, T$. Denote $\{\boldsymbol{\mu}_p(0), \ldots, \boldsymbol{\mu}_p(T)\}$ the pre-service decisions made by Π. Denote the following joint reward-cost pair:

$$\begin{aligned}&(Reward^{(h)}(T), Cost^{(h)}(T))\\ &\triangleq \frac{1}{T}\sum_{\tau=0}^{T-1}\sum_{m}\mathbb{E}\big\{I_{[A_m(\tau+1)=1]}[\mu_{mp}(\tau)r_{mp}\\ &\qquad +(1-\mu_{mp}(\tau))r_{mc}];\\ &\qquad C_m(\mu_{mp}(\tau), s(z_h))\\ &\qquad +(1-\mu_{mp}(\tau))I_{[A_m(\tau+1)=1]}\overline{C}_m \mid z(\tau)=z_h\big\}.\end{aligned}$$

Here we use the notation $\mathbb{E}\{X; Y \mid A\}$ to denote $(\mathbb{E}\{X \mid A\}, \mathbb{E}\{Y \mid A\})$ to save space. Notice that this is a mapping from z_h to a subset in $\mathbb{R}^2$ and that both the reward and cost are continuous. Also note that $\mathbb{E}\{I_{[A_m(\tau+1)=1]}\} = a_m^{(i_h)}$, where $i_h = A_m^{(h)}$ is defined in (14) to denote the probability of having $A_m(\tau+1) = 1$ given $A_m(\tau) = A_m^{(h)}$. Using the independence of $\boldsymbol{A}(t)$ and $\boldsymbol{S}(t)$, and Caratheodory's theorem [42], it follows that there exists three vectors $\boldsymbol{\mu}_j^{(h)}(T) \in \mathcal{U}_{z_h}, j = 1, 2, 3$, with appropriate weights $\theta_j^{(h)}(T) \geq 0, j = 1, 2, 3$, and $\sum_i \theta_j^{(h)}(T) = 1$, so that:

$$\begin{aligned}&(Reward^{(h)}(T), Cost^{(h)}(T))\\ &\triangleq \sum_{j=1}^{3}\theta_j^{(h)}(T)\sum_m \big(a_m^{(i_h)}[\mu_{mpj}^{(h)}(T)r_{mp} + (1-\mu_{mpj}^{(h)}(T))r_{mc}];\\ &\qquad C_m(\mu_{mpj}^{(h)}(T), s(z_h)) + (1-\mu_{mpj}^{(h)}(T))a_m^{(i_h)}\overline{C}_m\big). \qquad (33)\end{aligned}$$

Now consider averaging the above over all z_h states. We get:

$$\begin{aligned}&(Reward_{av}(T), Cost_{av}(T))\\ &\quad\triangleq \sum_h \pi_h \sum_{j=1}^{3}\theta_j^{(h)}(T)\sum_m \big(a_m^{(i_h)}[\mu_{mpj}^{(h)}(T)r_{mp}\\ &\qquad +(1-\mu_{mpj}^{(h)}(T))r_{mc}];\\ &\qquad C_m(\mu_{mpj}^{(h)}(T), s(z_h)) + (1-\mu_{mpj}^{(h)}(T))a_m^{(i_h)}\overline{C}_m\big).\end{aligned}$$

Using a similar argument as in the proof of [10, Th. 1], one can show that there exist limit points $\theta_j^{(h)} \geq 0$ and $\boldsymbol{\mu}_j^{(h)}$ as $T \to \infty$, so that the reward-cost tuple can be expressed as:

$$\begin{aligned}&(Reward_{av}, Cost_{av})\\ &= \sum_h \pi_h \sum_{j=1}^{3}\theta_j^{(h)}\sum_m \big(a_m^{(i_h)}[\mu_{mpj}^{(h)}r_{mp} + (1-\mu_{mpj}^{(h)})r_{mc}];\\ &\qquad C_m(\mu_{mpj}^{(h)}, s(z_h)) + (1-\mu_{mpj}^{(h)})a_m^{(i_h)}\overline{C}_m\big).\end{aligned}$$

This shows that for an arbitrarily control algorithm Π, its average reward and cost can be expressed as those in problem (9). Hence, its budget limited average reward cannot exceed Φ, which is the optimal value of (9). This shows that $\Phi \geq I(\rho)$.

The other direction $I(\rho) \geq \Phi$ will be shown in the analysis of the LBISC algorithm, where we show that LBISC achieves an intelligence level arbitrarily close to Φ. □

Appendix B
Proof of Theorem 2

We prove Theorem 2 here with the following two lemmas, whose proofs are given in Appendix D. The first lemma bounds the error in estimating ϵ_m and δ_m.

Lemma 1: MLE(T) *ensures that:*

$$\begin{aligned}&Pr\big\{\max_m |\hat{\epsilon}_m - \epsilon_m, \hat{\delta}_m - \delta_m| \geq \frac{\log(V)}{\sqrt{N(T)}}\big\}\\ &\qquad\qquad \leq 2Me^{-\log(V)^2/4}. \qquad (34)\end{aligned}$$

The second lemma bounds the error in estimating γ^*. In the lemma, we define an intermediate dual function $\hat{g}_{\boldsymbol{\pi}}(\gamma)$ defined as:

$$\begin{aligned}\hat{g}_{\boldsymbol{\pi}}(\gamma) \triangleq \sum_h \pi_h \sum_m \sup_{\boldsymbol{\mu}_p^{(h)}} \Big\{&V\hat{a}_m^{(i_h)}[\mu_{mp}^{(h)}r_{mp} + (1-\mu_{mp}^{(h)})r_{mc}]\\ &-\gamma[C_m(\mu_{mp}^{(h)}, z_h) + (1-\mu_{mp}^{(h)})\hat{a}_m^{(i_h)}\overline{C}_m - \rho]\Big\}. \qquad (35)\end{aligned}$$

That is, $\hat{g}_{\boldsymbol{\pi}}(\gamma)$ is the dual function defined with the true distribution π_h and the estimated $\hat{a}_m^{(i_h)}$. We use $\hat{\gamma}^*$ to denote the optimal solution of $\hat{g}_{\boldsymbol{\pi}}(\gamma)$.

Lemma 2: With probability at least $1-2Me^{-\log(V)^2/4}$, DL *outputs a* γ_T^* *that satisfies* $|\gamma_T^*-\gamma^*|\leq d_\gamma$ *where* $d_\gamma \triangleq \frac{cV\log(V)}{\sqrt{N(T)}}$ *and* $c=\Theta(1)>0$. *Moreover,* $|\gamma^*-\hat{\gamma}^*|\leq d_\gamma$.

We now prove Theorem 2. *Proof:* (Theorem 2) We first prove the budget bound, followed by the intelligence performance.

(**Budget**) We want to show that under LBISC, if γ_T^* is estimated accurate enough (happens with high probability), $\tilde{d}(t)$ is deterministically bounded throughout. This will imply that the budget constraint is met. Note that under LBISC, $D_m(t)$ in (19) is defined with $\hat{a}_m^{(i)}$. Thus, we denote it as $\hat{D}_m(t)$.

To see this, first note from Lemma 1 that with probability at least $1-2Me^{-\log(V)^2/4}$, $\max_m|\hat{\epsilon}_m-\epsilon_m,\hat{\delta}_m-\delta_m|\leq\frac{\log(V)}{\sqrt{N(T)}}$. Hence, $\|\boldsymbol{\epsilon}-\hat{\epsilon}\|\leq\nu$ and $\|\boldsymbol{\delta}-\hat{\delta}\|\leq\nu$ when V is large. Thus, (25) in Assumption 1 holds. Denote

$$\hat{D}_{\min} \triangleq \min_{m,k,i}\{C_m(1,s_k)-\hat{a}_m^{(i)}\overline{C}_m : C_m(1,s_k)-\hat{a}_m^{(i)}\overline{C}_m>0\}, \tag{36}$$

and look at the algorithm steps (19) and (20). We claim that whenever $\tilde{d}(t)\geq Vr_d/\hat{D}_{\min}$, it stops increasing. To see this, let us fix an m and consider two cases.

(i) Suppose $\hat{D}_m(t)>0$ in (19) (defined with $\hat{a}_m^{(i)}$), then we must have $\mu_{mp}(t)=0$, as $V\hat{a}_m^{(i)}(r_{mp}-r_{mc})-d(t)\hat{D}_m(t)\leq 0$ in (20) by definition of $\hat{D}_{\min}$.

(ii) Suppose instead $\hat{D}_m(t)\leq 0$. Although in this case we set $\mu_{mp}(t)=1$, it also implies that the state s_k is such that $C_m(1,s_k)\leq\hat{a}_m^{(i)}\overline{C}_m$ (from the definition in (19)).

Combining the two cases, we see that whenever $\tilde{d}(t)\geq Vr_d/\hat{D}_{\min}$, $\tilde{C}(t)\leq\sum_m\hat{a}_m^{(i)}\overline{C}_m$, which implies that $\tilde{C}(t)\leq\rho$ by (25) in Assumption 1. Therefore, we conclude that:

$$\tilde{d}(t)\leq d_{\max}\triangleq Vr_d/\hat{D}_{\min}+MC_{\max}. \tag{37}$$

This completes the proof of (27). The proof for (28) is given in the intelligence part at (43).

(**Intelligence**) First, we add to both sides of (18) the *drift-augmentation* term $\Delta_a(t)\triangleq-(\gamma_T^*-\theta)^+[\rho-\tilde{C}(\boldsymbol{\mu}(t))]$ and obtain:

$$\Delta(t)+\Delta_a(t)-V\tilde{r}(\boldsymbol{\mu}(t)) \leq B-\left(V\tilde{r}(\boldsymbol{\mu}(t))+\tilde{d}(t)[\rho-\tilde{C}(\boldsymbol{\mu}(t))]\right). \tag{38}$$

From Lemma 2, we know that the event $\{|\gamma_T^*-\gamma^*|\leq d_\gamma\}$ takes place with probability at least $1-2Me^{-\log(V)^2/4}$. Hence, from now on, we carry out our argument conditioning on this event.

Note that in every time, we use the estimated $\hat{\boldsymbol{\epsilon}}$ and $\hat{\boldsymbol{\delta}}$ for decision making, i.e., we maximize:

$$\sum_m\mu_{mp}(t)[V[a_m^{(i)}+e(a_m^{(i)})](r_{mp}-r_{mc}) -d(t)(C_m(1,\boldsymbol{S}(t))-[a_m^{(i)}+e(a_m^{(i)})]\overline{C}_m)],$$

where $e(a_m^{(i)})=\hat{a}_m^{(i)}-a_m^{(i)}$. Let $\boldsymbol{\mu}_p^L(t)$ be the action chosen by LBISC, and let $\boldsymbol{\mu}_p^*(t)$ be the actions chosen by BISC, i.e. with the exact $a_m^{(i)}$ values. We have

$$\sum_m\mu_{mp}^L(t)V[\hat{a}_m^{(i)}(r_{mp}-r_{mc})-d(t)\hat{D}_m(t)] \geq\sum_m\mu_{mp}^*(t)V[\hat{a}_m^{(i)}(r_{mp}-r_{mc})-d(t)\hat{D}_m(t)].$$

With the definition of $\hat{D}_m(t)$ and that $\max_m|\hat{\epsilon}_m-\epsilon_m,\hat{\delta}_m-\delta_m|\leq\frac{\log(V)}{\sqrt{N(T)}}$ in Lemma 1, this implies that:

$$\sum_m\mu_{mp}^L(t)V[a_m^{(i)}(r_{mp}-r_{mc})-d(t)D_m(t)] \geq\sum_m\mu_{mp}^*(t)V[a_m^{(i)}(r_{mp}-r_{mc})-d(t)D_m(t)]-E_{tot},$$

where $E_{tot}\triangleq 2M(Vr_d+d_{\max}C_{\max})\frac{\log(V)}{\sqrt{N(T)}}$. This means that the actions chosen by LBISC approximately maximize (20). Therefore, by comparing the term in $V\tilde{r}(\boldsymbol{\mu}(t))+\tilde{d}(t)[\rho-\tilde{C}(\boldsymbol{\mu}(t))]$ in (38) and the definition of $g_h(\gamma)$ in (13), we have:

$$\Delta(t)+\Delta_a(t)-V\tilde{r}(\boldsymbol{\mu}(t))\leq B_1-g_{z(t)}(\tilde{d}(t)).$$

Here $B_1\triangleq B+E_{tot}$ and $B\triangleq\rho_{\max}^2+M^2C_{\max}^2$.

Now assume without loss of generality that $z(0)=z$ and let t_n be the n-th return time for $z(t)$ to visit z ($z(t)$ is a finite-state irreducible and aperiodic Markov chain. Hence, the return time is well-defined). We get:

$$\begin{aligned}
&\sum_{t=t_n}^{t_{n+1}-1}[\Delta(t)+\Delta_a(t)-V\tilde{r}(\boldsymbol{\mu}(t))]\\
&\leq B_1(t_{n+1}-t_n)-\sum_{t=t_n}^{t_{n+1}-1}g_{z(t)}(\tilde{d}(t))\\
&\overset{(*)}{\leq} B_1(t_{n+1}-t_n)-\sum_{t=t_n}^{t_{n+1}-1}g_{z(t)}(\tilde{d}(t_n))\\
&\quad+\sum_{t=t_n}^{t_{n+1}-1}(t-t_n)e_{\max}\\
&\leq B_1(t_{n+1}-t_n)-\sum_{t=t_n}^{t_{n+1}-1}g_{z(t)}(\tilde{d}(t_n))+\frac{T_n^2-T_n}{2}e_{\max}.
\end{aligned}$$

Here $e_{\max}=(MC_{\max}+\rho)^2$ and $T_n\triangleq t_{n+1}-t_n$ denotes the length of the n-th return period, and (*) follows since $|g_{z(t)}(\tilde{d}(t_n))-g_{z(t)}(\tilde{d}(t))|\leq|\tilde{d}(t_n)-\tilde{d}(t)|(MC_{\max}+\rho)$, and $|\tilde{d}(t_n)-\tilde{d}(t)|\leq(t-t_n)(MC_{\max}+\rho)$.

Taking an expectation over T_n, and using that for any state z_h, the expected time to visit z_h during the return period to z is π_h/π_z and that $\overline{T}_z=1/\pi_z$ [43], we get:

$$\begin{aligned}
&\mathbb{E}\Big\{\sum_{t=t_n}^{t_{n+1}-1}[\Delta(t)+\Delta_a(t)-V\tilde{r}(\boldsymbol{\mu}(t))]\mid z(t_n)=z,\tilde{d}(t_n)\Big\}\\
&\leq B_1\overline{T}_z-\overline{T}_z\sum_h\pi_h g_{z(t)}(\tilde{d}(t_n))+\frac{\overline{T}_z^2-\overline{T}_z}{2}e_{\max}\\
&\leq B_1\overline{T}_z-V\overline{T}_zI(\rho)+\frac{\overline{T}_z^2-\overline{T}_z}{2}e_{\max}
\end{aligned}$$

In the last step we have used $g_{\boldsymbol{\pi}}(\gamma)\geq g_{\boldsymbol{\pi}}(\gamma^*)\geq VI(\rho)$ in (12). Therefore, taking an expectation over $\tilde{d}(t_n)$ and taking

a telescoping sum over $\{t_0, t_1, \dots t_n\}$, and using an argument almost identical as in the proof of [23, Th. 2], we obtain:

$$\liminf_{n\to\infty} \frac{1}{n}\sum_{t=0}^{n-1} \mathbb{E}\big\{\tilde{r}(t) \mid z(0)=z\big\} \geq I(\rho) - B_1/V - \frac{e_{\max}(\overline{T}_z^2 - \overline{T}_z)}{2V\overline{T}_z} + \frac{1}{V}\liminf_{n\to\infty}\frac{1}{n}\sum_{t=0}^{n-1}\mathbb{E}\big\{\Delta_a(t) \mid z(0)=z\big\}. \tag{39}$$

It remains to show that the last term is small. To do so, recall its definition $\Delta_a(t) = -(\gamma_T^* - \theta)^+[\rho - \tilde{C}(\boldsymbol{\mu}(t))]$. According to LBISC, γ_T^* is fixed at time T. Hence, we have (from now on we drop the conditioning on $z(0)=z$ for brevity):

$$\liminf_{n\to\infty} \frac{1}{n}\sum_{t=0}^{n-1}\mathbb{E}\big\{\Delta_a(t)\big\} = -(\gamma_T^*-\theta)^+ \liminf_{n\to\infty}\frac{1}{n}\sum_{t=0}^{n-1}\mathbb{E}\big\{\rho - \tilde{C}(\boldsymbol{\mu}(t))\big\}$$

We thus want to show that this term is $O(1)$. To do so, note that ρ and $\tilde{C}(\boldsymbol{\mu}(t))$ are the service and arrival rates to the *actual* queue $d(t)$. From the queueing dynamics we have:

$$\begin{aligned}\liminf_{n\to\infty} \frac{1}{n}\sum_{t=0}^{n-1}\mathbb{E}\big\{\rho - \tilde{C}(\boldsymbol{\mu}(t))\big\} &\leq \liminf_{n\to\infty}\frac{1}{n}\sum_{t=0}^{n-1}\mathbb{E}\big\{1_{d(t)<\rho}\big\}(\rho + C_{\max}) \\ &= (\rho + C_{\max})\liminf_{n\to\infty}\frac{1}{n}\sum_{t=0}^{n-1}\Pr\big\{d(t)<\rho\big\}. \end{aligned} \tag{40}$$

Hence, it remains to show that the last term, i.e., $\liminf_{n\to\infty}\frac{1}{n}\sum_{t=0}^{n-1}\Pr\big\{d(t)<\rho\big\}$ is small.

Recall that $\hat{\gamma}^*$ is the optimal solution for $\hat{g}_{\boldsymbol{\pi}}(\gamma)$ defined in (35), and that $\hat{g}_{\boldsymbol{\pi}}(\gamma)$ is polyhedral with parameter $\beta = \Theta(1)$ according to Assumption 3. Hence, [13, Th. 1] shows that there exist $\Theta(1)$ constants N_1, D and $\eta_1 > 0$, such that at every time t, if $|\tilde{d}(t) - \hat{\gamma}^*| \geq D$,

$$\mathbb{E}\big\{|\tilde{d}(t+N_1) - \hat{\gamma}^*| \mid \tilde{d}(t)\big\} \leq |\tilde{d}(t) - \hat{\gamma}^*| - \eta_1. \tag{41}$$

Using $\theta = \max(\frac{V\log(V)^2}{\sqrt{N(T)}}, \log(V)^2)$, we see that when V is large, $\theta/2 \geq d_\gamma = \frac{cV\log(V)}{\sqrt{N(T)}}$. Combining it with $\tilde{d}(t) = d(t) + \gamma_T^* - \theta$ (conditioning on $\{|\gamma_T^* - \gamma^*| \leq d_\gamma\}$, we have $(\gamma_T^* - \theta)^+ = \gamma_T^* - \theta$) and $\hat{\theta} \triangleq \hat{\gamma}^* - \gamma_T^* + \theta \in [\theta/2, \theta]$, which implies that whenever $|d(t) - \hat{\theta}| \geq D$,

$$\mathbb{E}\big\{|d(t+N_1) - \hat{\theta}| \mid \tilde{d}(t)\big\} \leq |d(t) - \hat{\theta}| - \eta_1. \tag{42}$$

Using the same proof argument in [13, Th.1], one can show that there exist $\Theta(1)$ positive constants c_1 and c_2 that:

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\Pr\big\{|d(t) - \hat{\theta}| > D + l\big\} \leq c_1 e^{-c_2 l}. \tag{43}$$

This shows that $\overline{d(t)} = \Theta(\hat{\theta}) = O(\max(\frac{V\log(V)^2}{\sqrt{N(T)}}, \log(V)^2))$.

Since $D = \Theta(1)$ and $\theta \geq \log(V)^2$, it can be seen that when V is large,

$$\liminf_{n\to\infty}\frac{1}{n}\sum_{t=0}^{n-1}\mathbb{E}\big\{\rho - \tilde{C}(\boldsymbol{\mu}(t))\big\} \leq \liminf_{n\to\infty}\frac{1}{n}\sum_{t=0}^{n-1}\Pr\big\{|d(t)-\hat{\theta}| > D + \frac{2\log(V)}{c_2}\big\}(\rho + C_{\max}) \tag{44}$$

$$= O(1/V^2). \tag{45}$$

Combining with the fact that $\gamma_T^* - \theta = \Theta(V)$, we conclude that the last term in (39) is $O(1/V)$. This completes the proof of (29). □

APPENDIX C
PROOF OF THEOREM 3

We will use of the following technical lemma from [42] for our proof.

Lemma 3 [42]: Let $\mathcal{F}_n$ be filtration, i.e., a sequence of increasing σ-algebras with $\mathcal{F}_n \subset \mathcal{F}_{n+1}$. Suppose the sequence of random variables $\{y_n\}_{n\geq 0}$ satisfy:

$$\mathbb{E}\big\{||y_{n+1} - y^*|| \mid \mathcal{F}_n\big\} \leq \mathbb{E}\big\{||y_n - y^*|| \mid \mathcal{F}_n\big\} - u_n, \tag{46}$$

where u_n takes the following values:

$$u_n = \begin{cases} u & \text{if} \quad ||y_n - y^*|| \geq D, \\ 0 & \text{else}. \end{cases} \tag{47}$$

Here $u > 0$ is a given constant. Then, by defining $N_D \triangleq \inf\{k \mid \|y_n - y^*\| \leq D\}$, we have:

$$\mathbb{E}\big\{N_D\big\} \leq ||y_0 - y^*||/u. \ \Diamond \tag{48}$$

We now present the proof.

Proof (Theorem 3): To start, note that the first T slots are spent learning $\hat{\boldsymbol{\epsilon}}$, $\hat{\boldsymbol{\delta}}$, and $\hat{\boldsymbol{\pi}}$. Lemma 2 shows that after T slots, with high probability, we have $|\gamma_T^* - \gamma^*| \leq \frac{cV\log(V)}{\sqrt{N(T)}}$ for some $c = \Theta(1)$. Using the definition of $\tilde{d}(t)$, this implies that when V is large,

$$|\tilde{d}(T) - \gamma^*| \leq \theta/2 = \max(V\log(V)^2/\sqrt{N(T)}, \log(V)^2)/2. \tag{49}$$

Using (41) in the proof of Theorem 2, and applying Lemma 3, we see that the expected time for $\tilde{d}(t)$ to get to within D of $\hat{\gamma}^*$, denoted by $\tilde{T}_D$, satisfies:

$$\mathbb{E}\big\{\tilde{T}_D\big\} \leq N_1 \max(V\log(V)^2/\sqrt{N(T)}, \log(V)^2)/2\eta_1. \tag{50}$$

Here N_1 is due to the fact that the drift takes N_1 steps in (41), and recall that $N_1 = \Theta(1)$ is a constant in (41).

Since $|\gamma^* - \hat{\gamma}^*| \leq d_\gamma = cV\log(V)/\sqrt{N(T)}$, by defining $D_1 = cV\log(V)/\sqrt{N(T)} + D$, we conclude that:

$$\mathbb{E}\big\{T_{D_1}^{\texttt{LBISC}}\big\} \leq N_1 \max(\frac{V\log(V)^2}{2\eta_1\sqrt{N(T)}}, \frac{\log(V)^2}{2\eta_1}) + T. \tag{51}$$

In the case of BISC, one can similarly show that there exists $\Theta(1)$ constants N_1, D (only related to $z(t)$ and β), and η_2, so that,

$$\mathbb{E}\big\{|d(t+N_1)-\gamma^*| \mid d(t)\big\} \le |d(t)-\gamma^*| - \eta_2. \quad (52)$$

Since $\gamma^* = \Theta(V)$ [13] and $d(0)=0$, we conclude that:

$$\mathbb{E}\big\{T_D^{\texttt{LBISC}}\big\} \le N_1 V/\eta_2. \quad (53)$$

This completes the proof of the theorem. □

Appendix D
Proof of Lemmas 1 and 2

We will make use of the following results from [44]

Theorem 4 [44]: For a finite state irreducible and aperiodic and reversible Markov chain with state space with state space $\mathcal{G}$ steady-state distribution $\boldsymbol{\pi}$ and an initial distribution $\mathbf{q}$. Let $y = \max_{s_1,s_2\in\mathcal{G}} \frac{\pi_{s_1}}{\pi_{s_2}}$ and let $N_q = \|\frac{q_s}{\pi_s}, s\in\mathcal{G}\|_2$. Moreover, let $a = 1-\lambda_2$ where $\lambda_2 < 1$ is the second largest eigenvalue of the transition matrix. Then, for any subset $A \subset \mathcal{G}$ and let t_n be the number of visits to A in n steps. We have for any b that:

$$Pr\big\{|t_n - n\pi_A| \ge b\big\} \le 2N_q e^{-b^2 a/20ny}. \quad (54)$$

Proof (Lemma 1): We first estimate the number of times state $A_m(t)=0$ is visited, denoted by N_{m0}. Using Theorem 4, with $b=\sqrt{N(T)}\log(V)$ and using the fact that both N_q, y and $a>0$ are $\Theta(1)$, we have:

$$\Pr\big\{|N_{m0} - N(T)\pi_{m0}| \ge \sqrt{N(T)}\log(V)\big\} \le c_{m1}e^{-c_{m2}\log(V)^2}. \quad (55)$$

Thus, with high probability, the number of times we visit state $A_m(t)=0$ is within $N(T)\pi_{m0} \pm \sqrt{N(T)}\log(V)$. This implies that we try to sample the transition $0\to 1$ at least $N_{m0} = N(T)\pi_{m0} - \sqrt{N(T)}\log(V)$ times with high probability, and each time we succeed with probability ϵ_m. Thus, using the concentration bound in [45] for i.i.d. Bernoulli variables, we have that the number of times transition $0\to 1$ takes place, denoted by N_{m01}, satisfies:

$$\Pr\big\{|N_{m01} - \epsilon_m N_{m0}| \ge 0.95\sqrt{N(T)}\log(V)\big\} \le 2e^{\frac{-(0.95)^2 N(T)\log(V)^2}{2(\epsilon_m N_{m0} - \sqrt{N(T)}\log(V) + 0.95\sqrt{N(T)}\log(V)/3)}} \le 2e^{-\log(V)^2/2}. \quad (56)$$

Combining (55) and (56), we conclude that:

$$\begin{aligned}\Pr\big\{|\hat{\epsilon}_m - \epsilon_m| \le \frac{\log(V)}{\sqrt{N(T)}}\big\} &\ge 1 - 2e^{-\log(V)^2/2} - c_{m1}e^{-c_{m2}\log(V)^2} \\ &\ge 1 - e^{-\log(V)^2/4}. \end{aligned} \quad (57)$$

We can now repeat the above argument for $\hat{\delta}_m$ to obtain a similar result. Then, (34) can be obtained by applying the union bound. □

Proof (Lemma 2): In LBISC, we actually solve $\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma) = \sum_h \hat{\pi}_h \hat{g}_h(\gamma)$ for computing γ_T^*. Here $\hat{g}_h(\gamma)$ is due to the fact that we use the estimated values $\hat{\delta}_m$ and $\hat{\epsilon}_m$ in the dual function.

Define $e(\pi_h) = \hat{\pi}_h - \pi_h$ and $e(a_m^{(i_h)}) = \hat{a}_m^{(i_h)} - a_m^{(i_h)}$. Then, we can rewrite $\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma)$ as:

$$\begin{aligned}&\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma) \\ &\triangleq \sum_h [\pi_h + e(\pi_h)] \sum_m \sup_{\boldsymbol{\mu}_p^{(h)}} \Big\{ V[a_m^{(i_h)} + e(a_m^{(i_h)})] \\ &\quad \times [\mu_{mp}^{(h)} r_{mp} + (1-\mu_{mp}^{(h)}) r_{mc}] \\ &\quad - \gamma[C_m(\mu_{mp}^{(h)}, z_h) + (1-\mu_{mp}^{(h)})[a_m^{(i_h)} + e(a_m^{(i_h)})]\overline{C}_m - \rho]\Big\}. \end{aligned} \quad (58)$$

Using Lemma 1, we see that when V is large, with probability $1-2Me^{-\log(V)^2/4}$, $e(a_m^{(i_h)}) \le \frac{\log(V)}{\sqrt{N(T)}}$, which also implies that the estimated $\hat{\pi}_h$ is within $O(\frac{\log(V)}{\sqrt{N(T)}})$ of π_h. In this case, Assumption 1 implies that there exist a set of actions that achieve:

$$\sum_h \hat{\pi}_h \sum_{j=1}^{3} \theta_j^{(h)} \sum_m [C_m(\mu_{mpj}^{(h)}, z_h) + (1-\mu_{mpj}^{(h)})\hat{a}_m^{(i_h)}\overline{C}_m] \le \rho_0,$$

for some $\rho_0 = \rho - \eta > 0$ for some $\eta > 0$. Using the fact that $e(a_m^{(i_h)}) \le \frac{\log(V)}{\sqrt{N(T)}}$, we see that the same set of actions ensures that:

$$\begin{aligned}\sum_h \hat{\pi}_h \sum_{j=1}^{3} \theta_j^{(h)} \sum_m [&C_m(\mu_{mpj}^{(h)}, z_h) \\ &+ (1-\mu_{mpj}^{(h)})\hat{a}_m^{(i_h)}\overline{C}_m] \le \rho - \eta/2. \end{aligned} \quad (59)$$

Using [22, Lemma 1], this implies $\gamma_T^* \le \xi \triangleq \frac{2Vr_p}{\eta}$. Therefore, we have:

$$\begin{aligned}\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma_T^*) \ge g_{\boldsymbol{\pi}}(\gamma_T^*) &- \sum_h |e(\pi_h)| M(Vr_p + \frac{2Vr_p}{\eta}C_{\max}) \\ &- \sum_m \ (V \max_m |e(a_m^{(i_h)})| r_p + \frac{2Vr_p}{\eta} \max_m |e(a_m^{(i_h)})| C_{\max}).\end{aligned}$$

Similarly, we have:

$$\begin{aligned}\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma^*) \le g_{\boldsymbol{\pi}}(\gamma^*) &+ \sum_h |e(\pi_h)| M(Vr_p + \frac{2Vr_p}{\eta}C_{\max}) \\ &+ \sum_m \ (V \max_m |e(a_m^{(i_h)})| r_p + \frac{2Vr_p}{\eta} \max_m |e(a_m^{(i_h)})| C_{\max}).\end{aligned}$$

Denoting the last two terms as e_{tot}, we see that $e_{tot} = \Theta(V)$. Using $\hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma_T^*) \le \hat{g}_{\hat{\boldsymbol{\pi}}}(\gamma^*)$, we get:

$$g_{\boldsymbol{\pi}}(\gamma_T^*) \le g_{\boldsymbol{\pi}}(\gamma^*) + 2e_{tot}. \quad (60)$$

Using the polyhedral property (26), we conclude that:

$$|\gamma_T^* - \gamma^*| \le 2e_{tot}/\beta. \quad (61)$$

Finally using the fact that $\max(|e(\pi_h)|, |e(a_m^{(i_h)}|) \le \frac{\log(V)}{\sqrt{N(T)}}$ proves the bound for $|\gamma_T^* - \gamma^*|$. The bound for $|\hat{\gamma}^* - \gamma^*|$ can be similarly proven. □

REFERENCES

[1] G. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 4, pp. 1–13, Dec. 2015.

[2] M. M. Gear. (Jan. 2016). *How to Make the Amazon Echo the Center of Your Smart Home Wired*. [Online]. Available: http://www.wired.com/2016/01/iot-cookbook-amazon-echo/

[3] N. Benaich. (Dec. 2015). *Investing in Artificial Intelligence TechCrunch*. [Online]. Available: http://techcrunch.com/2015/12/25/investing-in-artificial-intelligence/

[4] I. Weber and A. Jaimes, "Who uses Web search for what? and how?" in *Proc. (WSDM)*, 2011, pp. 21–30.

[5] V. N. Padmanabhan and J. C. Mogul, "Using predictive prefetching to improve world wide Web latency," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 3, pp. 22–36, Jul. 1996.

[6] J. Lee, H. Kim, and R. Vuduc, "When prefetching works, when it doesn't, and why," *ACM Trans. Archit. Code Optim.*, vol. 9, no. 1, Mar. 2012, Art. no. 2.

[7] M. Marrs. (Jun. 2013). *Predictive Search: Is This the Future or the End of Search? WordStream*. [Online]. Available: http://techcrunch.com/2015/12/25/investing-in-artificial-intelligence/

[8] T. Ball and J. R. Larus, "Branch prediction for free," in *Proc. Conf. Programm. Lang. Design Implement. (ACM SIGPLAN)*, vol. 28. 1993, pp. 300–313.

[9] M. U. Farooq, "Store-load-branch (SLB) predictor: A compiler assisted branch prediction for data dependent branches," in *Proc. 19th IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2013, pp. 59–70.

[10] M. J. Neely, "Energy optimal control for time-varying wireless networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 7, pp. 2915–2934, Jul. 2006.

[11] C. W. Tan, D. P. Palomar, and M. Chiang, "Energy-robustness tradeoff in cellular network power control," *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 912–925, Jun. 2009.

[12] M. J. Neely, "Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1489–1501, Aug. 2006.

[13] L. Huang and M. J. Neely, "Delay reduction via Lagrange multipliers in stochastic network optimization," *IEEE Trans. Autom. Control*, vol. 56, no. 4, pp. 842–857, Apr. 2011.

[14] I. Hou and P. R. Kumar, "Utility-optimal scheduling in time-varying wireless networks with delay constraints," in *Proc. ACM MobiHoc*, 2010, pp. 31–40.

[15] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Proactive resource allocation: Harnessing the diversity and multicast gains," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 4833–4854, Aug. 2013.

[16] J. Spencer, M. Sudan, and K. Xu. (Nov. 2012). "Queuing with future information." [Online]. Available: https://arxiv.org/abs/1211.0618

[17] S. Zhang, L. Huang, M. Chen, and X. Liu, "Proactive serving decreases user delay exponentially: The light-tailed service time case," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 708–723, Apr. 2017.

[18] K. Xu, "Necessity of future information in admission control," *Oper. Res.*, vol. 63, no. 5, pp. 1213–1226, 2015.

[19] L. Huang, S. Zhang, M. Chen, and X. Liu, "When backpressure meets predictive scheduling," in *Proc. ACM MobiHoc*, 2014, pp. 33–42.

[20] J. Tadrous and A. Eryilmaz, "On optimal proactive caching for mobile networks with demand uncertainties," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2715–2727, Oct. 2016.

[21] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Proactive data download and user demand shaping for data networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1917–1930, Dec. 2015.

[22] L. Huang, X. Liu, and X. Hao, "The power of online learning in stochastic network optimization," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 1, pp. 153–165, 2014.

[23] L. Huang and M. J. Neely. (Aug. 2010). "Max-weight achieves the exact $[O(1/V), O(V)]$ utility-delay tradeoff under Markov dynamics." https://arxiv.org/abs/1008.0200

[24] G. Venkataraman, A. Lad, V. Ha-Thuc, and D. Arya, "Instant search: A hands-on tutorial," in *Proc. ACM SigIR*, 2016, pp. 1211–1214.

[25] M. Shann and S. Seuken, "An active learning approach to home heating in the smart grid," in *Proc. IJCAI*, 2013, pp. 2892–2899.

[26] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. Netw.*, vol. 2, no. 1, pp. 1–15, Feb. 1994.

[27] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM IMC*, 2010, pp. 267–280.

[28] S. Fowler, J. Sarfraz, M. Abbas, and V. Angelakis, "Gaussian semi-Markov model based on real video multimedia traffic," in *Proc. IEEE ICC*, Jun. 2015, pp. 6971–6976.

[29] A. Paris, S. Arbaoui, N. Cislo, A. El-Amraoui, and N. Ramdani, "Using hidden semi-Markov model for learning behavior in smarthomes," in *Proc. IEEE CASE*, Aug. 2015, pp. 752–757.

[30] W. Kong, Z. Dong, and D. Hill, "A hierarchical hidden Markov model framework for home appliance modelling," *IEEE Trans. Smart Grid*, to be published, doi: 10.1109/TSG.2016.2626389.

[31] H. Yu, M. Cheung, L. Huang, and J. Huang, "Power-delay tradeoff with predictive scheduling in integrated cellular and Wi-Fi networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 4, pp. 735–742, Apr. 2016.

[32] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1333–1344, Dec. 2007.

[33] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 302–315, Apr. 2006.

[34] L. Ying, S. Shakkottai, and A. Reddy, "On combining shortest-path and back-pressure routing over multihop wireless networks," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 1–9.

[35] M. J. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 396–409, Apr. 2008.

[36] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.

[37] J. Walrand, *Probability in Electrical Engineering and Computer Science*. Seattle, WA, USA: Amazon, 2014.

[38] J. Liu, A. Eryilmaz, N. B. Shroff, and E. S. Bentley, "Heavy-ball: A new approach to tame delay and convergence in wireless network optimization," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[39] B. Li, R. Li, and A. Eryilmaz, "On the optimal convergence speed of wireless scheduling for fair resource allocation," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 631–643, Apr. 2015.

[40] M. J. Neely, "Energy-aware wireless scheduling with near optimal backlog and convergence time tradeoffs," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2223–2236, Aug. 2016.

[41] L. Huang, "Receding learning-aided control in stochastic networks," *Perform. Eval.*, vol. 91, pp. 150–169, Oct. 2015.

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

[43] D. Aldous and J. Fill. *Reversible Markov Chains Random Walks Graphs Monograph in Preparation*, accessed on Jul. 8, 2017. [Online]. Available: http://www.stat.berkeley.edu/~aldous/RWG/book.html

[44] D. Gillman, "A chernoff bound for random walks on expander graphs," in *Proc. IEEE FOCS*, Nov. 1993, pp. 680–691.

[45] F. Chung and L. Lu, "Concentration inequalities and martingale inequalities: A survey," *Internet Math.*, vol. 3, no. 1, pp. 79–127, 2006.

Longbo Huang received the Ph.D. degree in EE from the University of Southern California in 2011. He was a Post-Doctoral Researcher with the EECS Department, University of California (UC) at Berkeley, from 2011 to 2012. He is currently an Assistant Professor with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China.

He was a Visiting Scientist with the Simons Institute for the Theory of Computing, UC Berkeley, in 2016. He has been a Visiting Scholar with the LIDS Laboratory, MIT, and the EECS Department, UC Berkeley, and a Visiting Professor with The Chinese University of Hong Kong, Bell-Labs, France, and Microsoft Research Asia. His current research interests are in the areas of online learning, network optimization, online algorithm design, and sharing economy. He was selected into the China's Youth 1000-Talent Program in 2013, and received the Outstanding Teaching Award from Tsinghua University in 2014. He has served as the Lead Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Special Issue on Human-In-The-Loop Mobile Networks in 2016. He is an Associate Editor of the *ACM Transactions on Modeling and Performance Evaluation of Computing Systems* in 2017–2019.