Online Learning of Weakly Coupled MDP Policies for Load Balancing and Auto Scaling

Eshwar S.R., Lucas Lopes, Alexandre Reiffers-Masson, Daniel Sadoc Menasche, Gugan Thoppe

Online Learning of Weakly Coupled MDP Policies for Load Balancing and Auto Scaling

S.R. Eshwar*, Lucas Lopes Felipe§, Alexandre Reiffers-Masson‡, Daniel Sadoc Menasché§, Gugan Thoppe*
*Department of Computer Science and Automation, Indian Institute of Science, Bangalore
‡Department of Computer Science, IMT Atlantique Bretagne-Pays de la Loire, Campus de Brest, 29238 Brest CEDEX 03
§Institute of Computing, Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil

***Abstract*—Load balancing and auto scaling are at the core of scalable, contemporary systems, addressing dynamic resource allocation and service rate adjustments in response to workload changes. This paper introduces a novel model and algorithms for tuning load balancers coupled with auto scalers, considering bursty traffic arriving at finite queues. We begin by presenting the problem as a weakly coupled Markov Decision Processes (MDP), solvable via a linear program (LP). However, as the number of control variables of such LP grows combinatorially, we introduce a more tractable relaxed LP formulation, and extend it to tackle the problem of online parameter learning and policy optimization using a two-timescale algorithm based on the LP Lagrangian. Our numerical experiments shed insight into properties of the optimal policy. In particular, we identify a phase transition in the probability of job acceptance as a function of the job dropping costs. The experiments also indicate the efficacy of the proposed online learning method, that learns parameters together with the optimal policy, in converging to the optimal solution of the relaxed LP. In summary, the contributions of this work encompass an analytical model and its LP-based solution approach, together with an online learning algorithm, offering insights into the effective management of distributed systems.**

***Index Terms*—Resource Allocation, Queuing Systems, Linear Programming, Markov Decision Process, Online Learning.**

I. Introduction

The management of distributed systems has grown increasingly complex with the widespread adoption of cloud computing and microservice architectures. These architectures rely on two fundamental pillars: load balancing, which dynamically distributes traffic among servers, and auto scaling, which adjusts service rates in response to workload changes. Load balancers and auto scalers are pervasive in numerous modern systems, ranging from routers to databases and web systems [1]–[3].

Despite the extensive literature on load balancing and auto scaling methods, most studies tend to treat these elements separately [4]–[7] or focus on formal results for asymptotic regimes [8]. However, as systems evolve over time due to workload changes, such as in e-commerce platforms experiencing surges in user traffic during peak seasons or promotional events, the need for rapid resource provisioning and efficient utilization to prevent service disruptions becomes paramount. This paper addresses the concurrent optimization of load balancing and auto scaling in such scenarios.

ISBN 978-3-903176-63-8 ©2024 IFIP

In this paper, we introduce a novel model and algorithm addressing the general problem of load balancing and auto scaling in a system of finite parallel queues with unknown parameters and bursty traffic. We present the problem as a finite horizon weakly coupled Markov Decision Process (MDP), where decisions on load balancing and auto scaling are taken at each arrival of a bulk of jobs. These decisions involve determining which queues will receive new jobs and at what rate each queue will serve them. The weakly coupled MDP solution is achieved through a linear program (LP) (see Section III-B). However, it faces challenges due to the combinatorial growth of optimization variables. This motivates a relaxation of the problem by replacing instantaneous constraints by constraints on expectations. As a result, a simpler LP formulation is obtained (see Section III-C).

Solving the proposed LP requires knowledge of system dynamics, which may not always be available upfront. To address this, we propose a sample-based online learning algorithm to find the optimal policy for load balancing and auto scaling. This approach involves using a two-timescale algorithm to solve the LP by considering its Lagrangian (see Section III-D). Numerical experiments (see Section IV) across various scenarios demonstrate the convergence of the proposed method to the optimal LP solution, indicating its effectiveness in addressing the challenges of load balancing and auto scaling in dynamic systems with bursty traffic.

Pior art. Given the widespread adoption of load balancing and auto scaling, a few frameworks considered a stochastic model for their joint analysis (see [9]–[12]). Our work also considers load balancing and auto scaling decisions. However, by tackling the problem as a weakly coupled MDP, we can embrace general dispatching and auto scaling rules, beyond the ones considered in the previous works. In this work, we leverage [13], [14] showing its applicability in optimizing resource allocation and service rates in distributed systems. In addition, we extend those previous works by proposing a method to learn LP-based policies, accounting for the case when parameters are unknown.

Contributions. In summary, our main contributions are:

Analytical model: We formulate the load balancing and service rate control problem as a Weakly Coupled MDP (WC-MDP). The objective is to minimize costs arising from delays (proportional to queue lengths), energy consumption (proportional to service rate), and job dropouts due to full queues.

These factors are integrated into an LP along with workload and resource constraints (Sections II, III-A, and III-B). Note that, by treating the joint load balancing and auto scaling problem as a weakly coupled MDP, we can adopt more general dispatching and auto scaling rules beyond traditional strategies.

Online learning algorithm: We utilize recent advancements in LP-based policies to address a relaxed version of the above mentioned LP problem (Section III-C). Recognizing that system parameters may not be initially observable, we devise an online learning algorithm to approximate the LP-based policy. Specifically, we leverage the Lagrangian of the LP in this process. Our algorithm employs a two-timescale stochastic approximation, where Lagrange multipliers are computed based on the current estimate of control variables, and then control variables are recalculated given the Lagrange multipliers (Section III-D). Notably, this approach enables decision-making at each iteration of the algorithm, aligning with an online operational paradigm.

Numerical experiments: We illustrate properties of optimal policies, such as a phase transition in job acceptance probability, indicating a shift from predominant rejection to full acceptance as rejection costs increase (Section IV). We also discuss convergence properties of the proposed algorithms.

Outline. The remainder of this paper is organized as follows. In the upcoming section, we present background and related work. Then, Section II introduces the proposed model, Section III contains our problem formulation and solution, Section IV reports our results and Section V concludes.

II. Model

Next, we detail the considered queuing system and its corresponding dynamics. We also introduce basic notation.

The **system** is composed of N queues designated for the processing of jobs, each equipped with a finite buffer of capacity K. The **controller** has two controls for each of the N queues: (1) *load balancing* involves deciding whether to send a job to the tail of the queue; (2) *service rate management* determines the service rate of the queue, which in turn impacts the likelihood of processing a job in a given time slot, i.e., the probability of a service completion at a given slot is a function of whether the high or low service rates were chosen.

The **job arrival process** consists of arrivals of batches of αN jobs, where $0 < \alpha < 1$. The arrival time of the i-th batch of new jobs is denoted as $T_i \in \mathbb{N}_+$. The inter-arrival time, $T_i - T_{i-1}$, follows a geometric distribution with parameter p, given by $\mathbb{P}(T_i - T_{i-1} = \tau) = (1-p)^{\tau-1}p,\ \forall \tau \geq\ 1$.

The **dynamics of each queue** is characterized at embedded points, corresponding to batch arrivals. Let $S_n(T_i)$ denote the number of jobs in the n-th queue at the beginning of slot T_i. We denote by $A_n(T_i)$ an indicator variable, equal to 1 if a job is admitted to the n-th queue at T_i and 0 otherwise. Let $D_n(T_i)$ be the number of jobs processed between T_i and T_{i+1}.

Arrivals at time slot T_i precede departures spanning from T_i to T_{i+1}. Following the transition at time T_i, if $A_n(T_i) = 1$, an extra job is admitted to the n-th queue, and can be processed between T_i and T_{i+1} with the other jobs already in the queue. Therefore,

$$S_n(T_{i+1}) = S_n(T_i) + A_n(T_i) - D_n(T_i). \tag{1}$$

Then, $D_n(T_i) \leq S_n(T_i) + A_n(T_i)$ and $S_n(T_i) = K \Rightarrow A_n(T_i) = 0$. The number of rejected jobs across all queues is given by

$$R(T_i) = \alpha N - \sum_{n=1}^{N} A_n(T_i). \tag{2}$$

We denote by $B_n(T_i)$ an indicator variable representing the service rate of the n-th queue, with the values 0 and 1 indicating that the server operates at a low rate and at a high rate, respectively. It is assumed that the service rate remains constant between two arrivals. We let $q(b)$ denote the probability of a service completion, at a given slot, when the controller sets the service rate indicator to b. The service completion probabilities corresponding to the low and high service rates are given by $q(0) = \underline{b}$ and $q(1) = \overline{b}$.

The **transition probabilities** between states of a queue are defined by P. To derive those probabilities, we let s be the current state, a be the current job allocation action, and s' be the next state.

Next, we consider the interval comprised of τ time slots, wherein the system switches from state s to state s'. We account for two possible scenarios, varying on how jobs are serviced across those slots:

Concurrent job service per slot, per queue (CJS): Each pending job at each queue can be served concurrently with other jobs. In this case, more than one job can be served per queue per time slot. This corresponds to a setup wherein a cluster of servers, or a single server with multiple cores, is available for each queue.

Single job service per slot, per queue (SJS): Only one job can be processed per queue at a given time slot. In particular, at each time slot, we assume that **head-of-line (HOL)** jobs across queues are candidates to be served. This corresponds to a setup wherein a single core is available to serve each queue per time slot.

Let $\mathbb{P}(\mathcal{B}_{m,p} = \ell)$ denote the probability mass function of the binomial distribution, indicating the probability of ℓ successes in m independent Bernoulli trials. Then, $\mathbb{P}(\mathcal{B}_{m,p} = \ell) = \binom{m}{\ell} p^{\ell}(1-p)^{m-\ell}$ where $\binom{m}{\ell} = m!/((m-\ell)!\ell!)$ for $0 \leq \ell \leq m$, and 0 otherwise. We use the above notation to describe the distribution of the number of jobs processed in a given time period. The transition probabilities for CJS and SJS are presented in the following lemma:

Lemma 1. *Let $s, s' \in \{0, 1, ..., K\}, a \in \{0, 1\}$ and $b \in \{0, 1\}$. Then, the transition probabilities $\mathbb{P}(S_n(T_{i+1}) = s' \mid S_n(T_i) = s, A_n(T_i) = a, B_n(T_i) = b)$, denoted by $P_{s,s',a,b}$ are given by:*

$$P_{s,s',a,b} = \sum_{\tau=1}^{\infty} (1-p)^{\tau-1} p P_{s,s',a,b,\tau} \tag{3}$$

where

$$P_{s,s',a,b,\tau} := \mathbb{P}(S_n(T_{i+1}) = s' \mid S_n(T_i) = s, A_n(T_i) = a, B_n(T_i) = b, T_{i+1} - T_i = \tau). \tag{4}$$

Under CJS,

$$P_{s,s',a,b,\tau} = \mathbb{P}\left(\mathcal{B}_{s+a,1-(1-q(b))^\tau} = s + a - s'\right). \tag{5}$$

Under SJS,

$$P_{s,s',a,b,\tau} = \begin{cases} \mathbb{P}\left(\mathcal{B}_{\tau,q(b)} \geq s + a\right), & \text{if } s' = 0 \\ \mathbb{P}\left(\mathcal{B}_{\tau,q(b)} = s + a - s'\right), & \text{if } s' > 0. \end{cases} \tag{6}$$

Proof. Let s, a, b denote the state of the queue, job allocation action and service rate action at time T_i. Let s' denote the state of the queue at time T_{i+1}. We denote $\mathbb{P}(S_n(T_{i+1}) = s' \mid S_n(T_i) = s, A_n(T_i) = a, B_n(T_i) = b)$ with some abuse of notation as $P(s'|s, a, b)$. Let $\Delta T_i = T_{i+1} - T_i$. Then,

$$\mathbb{P}(s' \mid s, a, b) = \sum_{\tau=1}^{\infty} \mathbb{P}(s', \Delta T_i = \tau \mid s, a, b) = \sum_{\tau=1}^{\infty} \mathbb{P}(\Delta T_i = \tau) P_{s,s',a,b,\tau}$$

where the conditional probability $P_{s,s',a,b,\tau}$ is defined in (4).

SJS. We now derive $P_{s,s',a,b,\tau}$ for SJS. Across τ time slots, we aim at characterizing the occurrence of $s + a - s'$ successes, if $s' > 0$, and at least $s + a$ successes, if $s' = 0$, with the success probability at each time slot being given by $q(b)$. Indeed, given an interarrival time between jobs of τ, the number of jobs processed in this time period, at the considered queue, is $s + a - s'$, where $s + a - s' \leq \tau$. The service rate b entails a probability of successful job completion per slot given by $q(b)$. Hence, if $s' > 0$, the probability of processing $s + a - s'$ jobs during τ time slots is given by the binomial distribution $\mathbb{P}\left(\mathcal{B}_{\tau,q(b)} = s + a - s'\right)$. Note that this probability is strictly positive if $0 \leq s + a - s' \leq \tau$, and equals 0 otherwise. The case $s' = 0$ is similar, noting that even if more than $s + a$ potential successes occur, only $s + a$ jobs can be served.

CJS. We now derive the $P_{s,s',a,b,\tau}$ for CJS. Recall that under CJS multiple jobs can be processed per queue at a given time slot. With $s + a - s'$ jobs served during an interval of τ time slots, a binomial distribution with $s + a - s'$ successes among $s + a$ trials is used to characterize the probability of transitioning from s to s', where each potential service is completed with probability $q(B_n(t))$ per slot. Indeed, at each time slot, each of the pending jobs is processed with probability $q(b)$. Therefore, the probability that a job is served during a period of τ time slots is $1 - (1 - q(b))^\tau$. The probability that $s+a-s'$ jobs are served is $\mathbb{P}\left(\mathcal{B}_{s+a,1-(1-q(b))^\tau} = s + a - s'\right)$. Note that this probability is strictly positive if $s' \leq s + a$, and equals 0 otherwise. □

In Lemma 1 we are considering the system dynamics from the perspective of one queue. In particular, the dimensions of the transition matrix P are given by the maximum values of (s, s', a, b), which are $(K + 1, K + 1, 2, 2)$. The combinatorial growth in the state space cardinality occurs when examining a system comprised of multiple queues, as detailed next.

III. Problem formulation and solution

In this section, we begin by introducing our cost functions and constraints (Section III-A) and then formulate our problem as a weakly coupled MDP (Section III-B). The solution of a weakly coupled MDP is known to be an NP-hard problem when constraints are set on sample paths [14]. Therefore, we relax the problem to consider constraints on expectations, amenable to be solved with a manageable LP (Section III-C). The latter, in turn, is asymptotically optimal, in the sense that as the number of queues grows to infinity its solution converges to that of the original problem [13]. We proceed to expand the LP problem into its Lagrangian form and introduce a two-timescale stochastic approximation algorithm to tackle the problem in cases where transition probabilities are unknown, and the optimal policy must be learned in an online fashion (Section III-D).

A. Optimization problem

We introduce the cost functions and constraints to formulate the optimization problem.

The **cost functions** comprise two components:

1) The **storage and processing costs**, comprising the *storage cost* $C_s(S_n(t))$ which, due to Little law, is proportional to delays, and the *processing cost* $C_p(B_n(t))$, which is proportional energy consumption. Both costs are assumed to be convex and increasing;
2) The **job rejection cost**, given by $\gamma R(t)$ (see (2)), where γ is a positive factor capturing the weight of job rejections when compared against other costs. Note that as αN is a constant, minimizing job rejection costs $\gamma R(t)$ is equivalent to minimizing $-\gamma \sum_k A_n(t)$.

The **constraints** also comprise two components:

1) For the **storage cost**, the allocation of jobs to queues must not exceed the batch size of αN jobs, where $\alpha \in (0, 1)$;
2) For the **energy cost**, the total number of queues operating at a high processing rate $\overline{b}$ must not exceed βN, where $\beta \in (0, 1)$.

Collectively, these elements define the optimization problem, as detailed in the subsequent subsection.

B. Problem formulation

Before we state the optimization problem, we introduce some key notations. Let $Y^{(N)}_{s,a,b}(t)$ be the fraction of queues in state s and subject to actions a, b at time t. We consider a finite time horizon T, so that $0 \leq t \leq T - 1$. Let Y be a vector of random variables, $Y^{(N)} = (Y^{(N)}(1), \ldots, Y^{(N)}(t), \ldots, Y^{(N)}(T))$, where each $Y^{(N)}(t)$ is a random vector comprised of elements $Y^{(N)}_{s,a,b}(t)$.

Let $M^{(N)}(t) = (M^{(N)}_s(t))_{s \in \mathcal{S}}$ be a vector whose s-th entry corresponds to the fraction of queues containing s jobs at time

t. Note that $M_s^{(N)}(t) = \sum_{a,b} Y_{s,a,b}^{(N)}(t)$. We denote by m_s^0 the initial system state, $m_s^0 = M_s^{(N)}(0)$. It corresponds to the fraction of queues in state s at time $t = 0$ for all $s \in \mathcal{S}$. Unless otherwise noted, we assume that queues begin in an empty state, i.e., $m_0^0 = 1$.

1) Optimization problem: Let Π be a decision rule (or policy) that depends on both time and the state of the system. Then, $\Pi_t : m \mapsto y$ and $\Pi_t(M^{(N)}(t)) \mapsto Y^{(N)}(t)$. The variables $Y^{(N)}(t)$ are then set as $Y^{(N)}(t) = \Pi_t(M^{(N)}(t))$. The optimization problem can now be formulated as follows:

PROBLEM WITH INEQUALITY CONSTRAINTS ON SAMPLE PATHS:

$$\min_{\pi} \quad \mathbb{E} \sum_{t=0}^{T-1} \left(\sum_{s,a,b} (C_s(s)+C_p(b)) Y_{s,a,b}^{(N)}(t) + \gamma \sum_{s,b} Y_{s,0,b}^{(N)}(t) \right)$$

s.t. Queues evolve according to a Markov process,

$$\mathbb{P}(\boldsymbol{s}(t+1)|\boldsymbol{s}(t), \boldsymbol{a}(t), \boldsymbol{b}(t)) = \prod_{n=1}^{N} P_{s_n(t), s_n(t+1), a_n(t), b_n(t)}, \tag{7a}$$

$$\sum_{s,b} Y_{s,1,b}^{(N)}(t) \leq \alpha, \quad \sum_{s,a} Y_{s,a,1}^{(N)}(t) \leq \beta, \tag{7b}$$

$$\sum_{a,b} Y_{s,a,b}^{(N)}(0) = m_s^0, \quad Y_{K,1,b}^{(N)} = 0, \tag{7c}$$

$$Y_{s,a,b}^{(N)}(t) \geq 0, \forall t \in \{0, \ldots, T-1\}, \forall s \in \{0, \ldots, K\}, \quad \forall a \in \{0,1\}, \forall b \in \{0,1\} \tag{7d}$$

The objective function captures costs introduced in the previous section. State dynamics in (7a) follow the transitions derived in Lemma 1. Inequality constraints (7b) capture constraints on the workload and energy. Boundary conditions, i.e., initial conditions and droppings due to full buffers, are captured by (7c), and non-negativity constraints by (7d). It is possible to write the above problem as an LP. However, the number of optimization variables grow combinatorially with buffer size and number of queues.

2) Complexity: It is possible to write the above problem as a very large LP. Let set $\mathcal{Y}(m)$ contain all vectors $\hat{y}$ that satisfy (7b), and such that the fraction of queues in state s is given by m_s, with $m = (m_0, m_1, \ldots, m_K)$.

The variables in our LP are probabilities $p_{t,\hat{y}}$ for all admissible $\hat{y}$, $\hat{y} \in \mathcal{Y}(m)$, where $p_{t,\hat{y}} = \mathbb{P}(Y(t) = \hat{y})$. We also let $q_{t,m} = \mathbb{P}(M(t) = m)$, and $P(m|\hat{y}) = \mathbb{P}(M(t+1) = m|Y(t) = \hat{y})$. Then, equation (7a) is captured in the LP through the following constraint $q_{t+1,m} = \sum_{\hat{y}} p_{t,\hat{y}} P(m|\hat{y})$. The Markovian evolution is given by $P(m|\hat{y})$, which denotes the probability of the system transitioning from state-action pair $\hat{y}$ to state m. Inequalities (7b) are captured in the LP by $q_{t,m} = \sum_{\hat{y} \in \mathcal{Y}(m)} p_{t,\hat{y}}$, indicating that $\hat{y}$ must be feasible for $p_{t,\hat{y}}$ to contribute to $q_{t,m}$, and relating the probabilities of state-action vectors $\hat{y} \in \mathcal{Y}(m)$ to the probability of state vector m.

The issue with the above formulation arises as the number of possible values for $\hat{y}$ quickly becomes very large. In fact, the primary challenge in solving the optimization problem above lies in its state space cardinality. Note that the random vector $Y(t)$ can be instantiated in up to $\binom{4(K+1)-1+N}{N}$ possible ways, each corresponding to a decision variable $p_{t,\hat{y}}$ to be determined. The binomial term corresponds to the number of ways of dividing the N queues into $4(K+1)$ buckets, each bucket corresponding to one of the possible instantiations of (s, a, b). Despite the large number of variables $p_{t,\hat{y}}$ to be determined, the variables are all continuous, in the range between 0 and 1. Hence, the problem is in the realm of LPs.

As the constraints (7b) couple the states of all queues, they preclude the possibility of dividing the problem into N problems of linear size each, rendering its solution unwieldy. In what follows, we resort to a relaxation of the constraints, which allows us to formulate the relaxed problem as a manageable LP.

C. Relaxed problem becomes a manageable linear program

Next, we consider a relaxed version of the problem introduced in the previous section. As discussed above, the key difficulty in solving the posed problem arises from constraints (7b) which couple all the queues together. To overcome the challenge, we relax the constraints. The common approach is to relax the constraints by posing a problem where they only need to be met in expectation. Let

$$y_{s,a,b}(t) = \mathbb{E}[Y_{s,a,b}^{(N)}(t)]. \tag{8}$$

Let $\pi_H(t)$ and $\pi_A(t)$ be the probability of using the high service rate and the probability of accepting a job, respectively,

$$\pi_H(t) = \sum_{s=0}^{K} \sum_{a=0}^{1} y_{s,a,1}(t), \quad \pi_A(t) = \sum_{s=0}^{K-1} \sum_{b=0}^{1} y_{s,1,b}(t). \tag{9}$$

The relaxed problem to derive the optimal policy is an LP:

PROBLEM WITH INEQUALITY CONSTRAINTS ON EXPECTATIONS:

$$\min_{y} \quad \sum_{t=0}^{T-1} \left(\sum_{s,a,b} (C_s(s) + C_p(b)) y_{s,a,b}(t) + \gamma(1 - \pi_A(t)) \right)$$

$$\text{s.t.} \quad \sum_{a,b} y_{s,a,b}(t+1) = \sum_{s',a,b} y_{s',a,b}(t) P_{s',s,a,b}, \tag{10a}$$

$$\pi_A(t) \leq \alpha, \quad \pi_H(t) \leq \beta, \tag{10b}$$

$$\sum_{a,b} y_{s,a,b}(0) = m_s^0, y_{K,1,b}(t) = 0, y_{s,a,b}(t) \geq 0, \tag{10c}$$

$$\forall t \in \{0, \ldots, T-1\}, \forall s \in \{0, \ldots, K\}, \quad \forall a \in \{0,1\}, \forall b \in \{0,1\}.$$

The solution to the relaxed LP converges asymptotically to the solution of the original LP problem as N grows to infinity [13]. However, applying the solution of the relaxed LP directly to the original system may not be feasible under the current system configuration. To address this limitation, in [13] the authors suggest to solve the LP taking the current

system configuration as the initial state, and the remaining time of interest as the horizon T [13, Algorithm 1]. An alternative approach is proposed in [15]. In essence, those works provide ways to map a solution to the LP with constraints on expectations (10b) to a solution to the stricter LP with constraints on samples paths (7b). Given that such mappings are known to exist, in the remainder of this paper we focus on the problem with constraints on expectations, noting that constraints on expectations are of interest by itself [16], and leaving a detailed analysis of the mapping between constraints on expectations and sample paths as subject for future work.

Handling the above LP becomes challenging when transition probabilities $P^{a,b}_{s',s}$ are unknown. One of our contributions lies in solving the relaxed LP in the presence of unknown transition probabilities. To this aim, we derive the Lagrangian of the relaxed LP and employ a two-timescale stochastic approximation scheme to tackle the LP in an online fashion. Our approach is further elaborated in the following section, with numerical experimental results presented in Section IV.

D. A two-timescale stochastic approximation algorithm

Next, we derive the Lagrangian for the relaxed LP formulation. Subsequently, we leverage the Lagrangian to propose a two-timescale stochastic approximation algorithm to solve the problem introduced in the previous section. We denote the Lagrangian of the LP in (10) as $L(y, \lambda, \mu)$. Here, λ and μ are vectors of Lagrange multipliers corresponding to the inequality and equality constraints in (10), respectively.

$$L(y,\lambda,\mu;P,m^0) = \sum_{t=0}^{T-1} \mathcal{C}(y,t) + \mathcal{D}_1(y,\mu,t;P,m^0) + \mathcal{D}_2(y,\lambda,t), \quad (11)$$

$\mathcal{C}(y,t) = \sum_{s,a,b}(C_s(s) + C_q(b))y_{s,a,b}(t) + \gamma \sum_{s,b} y_{s,0,b}(t)$.

The Lagrangian terms corresponding to equality and inequality constraints are denoted by $\mathcal{D}_1(y,\mu,t;P,m^0)$ and $\mathcal{D}_2(y,\lambda,t)$, respectively.

$$\mathcal{D}_1(y,\mu,t;P,m^0) = \sum_s \mu_1^s \left(\sum_{a,b} y_{s,a,b}(0) - m_s^0 \right) + \\ + \sum_b \mu_2^b(t) y_{K,1,b}(t) + \\ + \sum_s \mu_3^s(t) \left(\sum_{a,b} y_{s,a,b}(t+1) - \sum_{s',a,b} y_{s',a,b}(t) P^{a,b}_{s',s} \right),$$

$$\mathcal{D}_2(y,\lambda,t) = \lambda_1(t) \left(\sum_{s,b} y_{s,1,b}(t) - \alpha \right) + \\ + \lambda_2(t) \left(\sum_{s,a} y_{s,a,1}(t) - \beta \right) - \sum_{s,a,b} \lambda_3^{s,a,b}(t) y_{s,a,b}(t).$$

Since the Lagrangian exhibits linearity with respect to y, the derivative of the Lagrangian with respect to y is constant. Therefore, we proceed by extending the previous formulation and introducing a regularization term to the problem. The regularized Lagrangian is expressed as follows:

$$\tilde{L}(y,\lambda,\mu;P,m^0) = L(y,\lambda,\mu;P,m^0) + \Gamma||y||_2^2. \quad (12)$$

Then, we minimize the Lagrangian with respect to y and maximize with respect to the Lagrange multipliers to find a saddle point which corresponds to an optimal policy:

PROBLEM WITH CONSTRAINTS AS PENALTIES:

$$\max_{\lambda \geq 0, \mu} \min_y \tilde{L}(y,\lambda,\mu;P,m^0). \quad (13)$$

If all parameters are known, the above problem can be solved using a Primal-Dual Gradient Descent-Ascent (GDA) method [17]–[19]. However, in this work we assume that the transition probabilities P is unknown. To this aim, we perform stochastic gradient updates. While computing the Lagrangian, at each step, instead of using the actual P matrix, we use an estimate of P. We denote by $\hat{P}^{(k)}_{s,s',a,b}$ the estimate of $P_{s,s',a,b}$ (see (3)) at iteration k, e.g., derived from a simulator. The estimate $\hat{P}^{(k)}_{s,s',a,b}$ is obtained by repeatedly performing actions a, b in state s. Let I be the number of collected observations. $\hat{P}^{(k)}_{s,s',a,b}$ corresponds to the fraction of those observations that lead to state s'. Our final stochastic gradient iterates are:

$$y^{(k+1)} = y^{(k)} - \eta_1 \nabla_y \tilde{L}(y^{(k)}, \lambda^{(k)}, \mu^{(k)}; \hat{P}^{(k)}, m^0) \quad (14)$$

$$\lambda^{(k+1)} = \left[\lambda^{(k)} + \eta_2 \nabla_\lambda \tilde{L}(y^{(k)}, \lambda^{(k)}, \mu^{(k)}; \hat{P}^{(k)}, m^0) \right]_+ \quad (15)$$

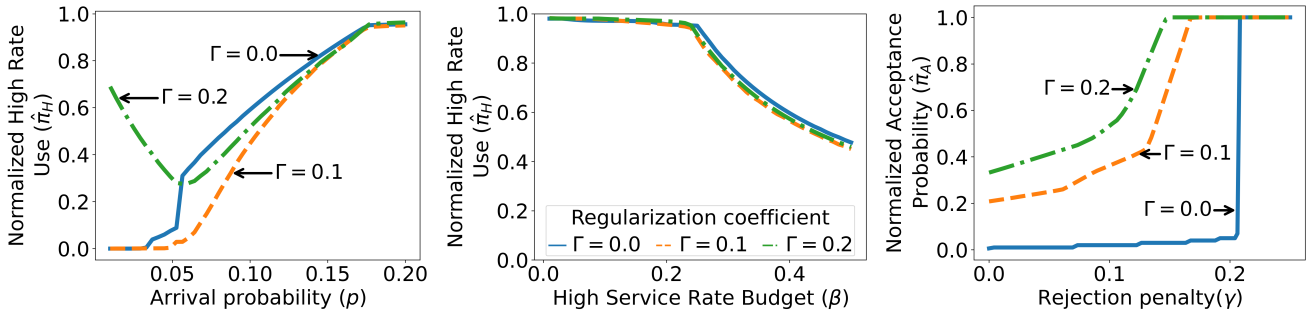
$$\mu^{(k+1)} = \mu^{(k)} + \eta_2 \nabla_\mu \tilde{L}(y^{(k)}, \lambda^{(k)}, \mu^{(k)}; \hat{P}^{(k)}, m^0) \quad (16)$$

The above description to solve the stochastic problem corresponds to a stochastic variant of GDA, referred to in the literature as Stochastic Gradient Descent-Ascent (SGDA). We let y vary at a fast time scale, and λ, μ vary at a slow scale. We denote by η_1 and η_2 the learning rates of the fast and slow time scales, respectively. Note that, $[x]_+ = \max(0, x)$, enforcing that $\lambda \geq 0$. Recall that the Lagrange multipliers λ and μ correspond to inequality and equality constraints, respectively. Therefore, after convergence of the above iterative procedure, the fixed point satisfies the KKT conditions, including complementary slackness, as $\lambda \geq 0$, and corresponds to an optimal solution to the LP problem.

IV. EXPERIMENTS

In this section we investigate properties of load balancing and auto scaling strategies, by analyzing the impact of different parameters on the optimal policy (Section IV-A). We also numerically investigate convergence properties of the considered algorithms (Section IV-B).

For our numerical evaluation, the storage cost is given by $C_s(s) = s/K$ and the processing costs are given by $C_p(b) = 2(1 + q(b))$. In particular, it follows from Little's law that the linear storage cost is proportional to the delay.



(a) (b) (c)

Fig. 1: Analysis of the impact of (a) arrival probability p, (b) high service rate budget β, and (c) job dropping cost γ on the solution of LP-based policy (10). The figure also shows the impact of the regularization coefficient, Γ, on the optimal policy.

A. How does the optimal policy behave?

Next, we report properties of the optimal policy, indicating how different parameters impact actions. To this goal, we consider the policy obtained through the solution of (10), i.e., the PROBLEM WITH INEQUALITY CONSTRAINTS ON EXPECTATIONS, leveraging CVXPY for that matter.

1) What are the impacts of different parameters on the optimal policy?: We consider the impact of p, β and γ on the optimal policy. To this aim, recall that $\pi_A(t)$ and $\pi_H(t)$ refer, respectively, to the probability of accepting a job and the probability of choosing the high service rate at time t. We derive from (9) the normalized quantities $\hat{\pi}_A$ and $\hat{\pi}_H$, $\hat{\pi}_A = \frac{1}{\alpha T}\sum_{t=0}^{T-1}\pi_A(t)$, and $\hat{\pi}_H = \frac{1}{\beta T}\sum_{t=0}^{T-1}\pi_H(t)$. Recall also that $\pi_A(t) \leq \alpha$ and $\pi_H(t) \leq \beta$. Therefore, $0 \leq \hat{\pi}_A \leq 1$ and $0 \leq \hat{\pi}_H \leq 1$, with $\hat{\pi}_A = 1$ and $\hat{\pi}_H(t) = 1$ if the corresponding inequality constraints are active.

We let $\alpha = 0.5$, $q(0) = 0.05$, $q(1) = 0.1$, $K = 10$ and $T = 100$, under the CJS scenario. In our reference setup, we also let $p = 0.14$, $\beta = 0.3$ and $\gamma = 10$. The latter three parameters are varied according to the experimental goals. Those values are selected to simplify presentation, allowing us to illustrate insights by varying a parameter, while maintaining all others fixed, as indicated in the sequel. The discussion that follows considers $\Gamma = 0$ (solid lines in Figure 1). The impact of Γ is pointed in the end of this section, motivated by SGDA.

Arrival rate up, high rate use rises. Figure 1(a) shows that as job arrival probability (p) increases, overall system utilization and the likelihood of using the high service rate increase. This implies that with more jobs, there's a greater chance of assigning a high service rate to minimize dropping probability and delay costs. Once p hits a threshold (e.g., $p \approx 0.2$ in this case), the high service rate constraint becomes active, preventing further increase of $\hat{\pi}_H$ (recall that $\hat{\pi}_H$ is normalized by the high service rate budget β).

Budget up, high rate use steady. Figure 1(b) illustrates the impact of the high service rate budget (β) on its allocated fraction ($\hat{\pi}_H$). When the constraint is stringent ($\beta \leq 0.28$), nearly the entire budget is utilized ($\hat{\pi}_H \approx 1$). Conversely, with a relaxed constraint, i.e., as β increases, the allocated fraction $\hat{\pi}_H$ decreases, and the high-rate use $\beta\hat{\pi}_H$ remains roughly steady. This suggests that increasing the budget for high service rate doesn't necessarily lead to increased resource consumption due to associated energy costs affecting the objective function. The motivation behind limiting high service rate is driven by budget constraints (β) and the need to manage energy expenditure reflected in the objective function's cost term ($C_p(b)$). Increasing β beyond a certain threshold leads to a decrease in normalized high rate use $\hat{\pi}_H$ due to $C_p(b)$.

Costlier drops, fewer rejections. As the cost associated with dropping jobs increases, the normalized probability of job acceptance shows a phase transition (see Figure 1(c)). Up to $\gamma = 0.2$, the normalized acceptance probability remains roughly 0, meaning that almost all jobs are rejected. Indeed, if the rejection cost is low, it is beneficial to keep the system almost always empty, to avoid incurring energy and storage costs. However, as γ surpasses the threshold 0.2, there is a phase transition, and the normalized acceptance probability remains stable at 1 for $\gamma > 0.2$. For large enough dropping costs, the acceptance budget should be fully utilized.

Regularization coefficient. Up until now, we considered $\Gamma = 0$. In the next section, we consider $\Gamma > 0$, as required by GDA and SGDA. Figure 1 shows how, as Γ increases, the behavior of the optimal policy diverges from that obtained with $\Gamma = 0$. For instance, we see a smooth transition of $\hat{\pi}_A$ as a function of γ, for $\Gamma \in \{0.1, 0.2\}$. This illustrates that one must carefully set Γ while relying on SGDA, noting that a detailed sensitivity analysis of Γ is subject for future work.

B. How do the proposed algorithms behave?

Next, we consider convergence properties. Our reference setup consists of $p = 1/2$, $\alpha = 1/2$, $\beta = 1/2$, $\gamma = 100$, ($\underline{b} = 0.4, \bar{b} = 0.8$) and $T = 10$. We let $\Gamma = 0.5$. K is varied between 1, 2, 4 and 8. Under SGDA, we collect a mini-batch of $I = 10$ observations before each gradient update, with stepsizes $\eta_1 = 0.1$ and $\eta_2 = 0.01$.

Recall that we leverage the gradient descent ascent algorithm (GDA) and its stochastic version (SGDA) to find solutions to the load balancing and auto scaling problem, by finding the saddle point associated with its Lagrangian (12). While GDA uses full information, SGDA learns the transition probabilities and the optimal policy concurrently. In the rest

of this section, we compare GDA, SGDA, given by (14)-(16) and the LP solution obtained with CVXPY, given by (10).

Firstly, our analysis focuses on the convergence of GDA towards the optimal solution of the LP problem in (10). The convergence reveals that the Frobenius norm of the difference between the GDA output and the optimal solution diminishes, approaching zero after 20,000 iterations, for buffer sizes (K) ranging from 1 to 8. It's noted that prior to stabilization, the solution discrepancy exhibits oscillatory behavior. This is typical of GDA, wherein disparities in learning rates used for minimization and maximization cause some updates to be too aggressive compared to others, leading to overshooting and thus inducing oscillatory behavior towards convergence [17].

Both CJS and SJS show similar trends, with SJS oscillations reaching slightly larger values. This occurs because SJS costs can vary across a broader range of values than CJS, as SJS relies on a single service per slot whereas CJS can concurrently serve multiple jobs at a given slot. Additionally, an increase in buffer size (K) decreases GDA's convergence speed, due to the larger cardinality of the set of control variables.

Secondly, we contrast the convergence behaviors of GDA and SGDA. In particular, we focus on the Frobenius norm of the difference between the solutions found by the two algorithms at each simulation epoch. Drawing from the theory of stochastic approximation (see Chapter 9 of [20]), SGDA's trajectories are expected to mirror that of GDA. Our results indicate a close agreement between SGDA and GDA after 35,000 iterations. A formal treatment of the convergence of SGDA, leveraging [20], is left as subject for future work.

V. Conclusion

We approached the challenge of load balancing and auto scaling across parallel queues through the lens of a weakly coupled MDP. Drawing on recent advancements in solving such systems, we proposed an LP-based policy and developed a novel online learning algorithm to refine this policy when parameters are unknown and may change dynamically. Our numerical experiments shed light on policy behaviors, including a phase transition in job acceptance probability as the dropping cost grows. The experiments also allow us to assess the efficacy of our online learning algorithm. Future work includes a real-world reality-check of the quality of the policies obtained through the proposed LP. To ensure reproducibility we make code and experiments publicly available [21].

Acknowledgements: Eshwar was supported by the Prime Minister's Research Fellowship (PMRF). G. Thoppe's research was supported in part by DST-SERB's Core Research Grant CRG/2021/008330, Walmart Center for Tech Excellence, and the Pratiksha Trust Young Investigator Award. This work was partially supported by CAPES STIC AMSUD project RAMONaaS, CNPq and FAPERJ through grants 315110/2020-1, E-26/211.144/2019 and E-26/201.376/2021.

References

[1] Shamsuddeen Rabiu, Chan Huah Yong, et al. A cloud-based container microservices: A review on load-balancing and auto-scaling issues. *International Journal of Data Science*, 3(2):80–92, 2022.

[2] Amazon. AWS Auto Scaling. https://aws.amazon.com/autoscaling/.

[3] Amazon. AWS Elastic Load Balancing (ELB). https://aws.amazon.com/elasticloadbalancing/.

[4] E. Gures, I. Shayea, Mustafa Ergen, M. Azmi, and A. El-Saleh. Machine learning-based load balancing algorithms in future heterogeneous networks: A survey. *IEEE Access*, 10:37689–37717, 2022.

[5] Divyansh Singh, Vandit Bhalla, and Neha Garg. Load balancing algorithms with the application of machine learning: A review. *MR International Journal of Engineering and Technology*, 10(1), 2023.

[6] Juliet G Muchori and Peter M Mwangi. Machine learning load balancing techniques in cloud computing: A review. 2022.

[7] Y. Garí, D. Monge, E. Pacini, C. Mateos, and C. Garino. Reinforcement learning-based application autoscaling in the cloud: A survey. *Engineering Applications of Artificial Intelligence*, 102:104288, 2021.

[8] R. Srikant and L. Ying. *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge, 2013.

[9] Jonatha Anselmi. Asynchronous load balancing and auto-scaling: Mean-field limit and optimal design. *arXiv:2204.02352*, 2022.

[10] Debankur Mukherjee, Souvik Dhara, Sem C Borst, and Johan SH van Leeuwaarden. Optimal service elasticity in large-scale distributed systems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):1–28, 2017.

[11] Debankur Mukherjee and Alexander Stolyar. Join idle queue with service elasticity: Large-scale asymptotics of a nonmonotone system. *Stochastic Systems*, 9(4):338–358, 2019.

[12] Yoann Desmouceaux, Marcel Enguehard, and Thomas H Clausen. Joint monitorless load-balancing and autoscaling for zero-wait-time in data centers. *IEEE Transactions on Network and Service Management*, 18(1):672–686, 2020.

[13] Nicolas Gast, Bruno Gaujal, and Chen Yan. The LP-update policy for weakly coupled Markov decision processes. *arXiv:2211.01961*, 2022.

[14] Nicolas Gast, Bruno Gaujal, and Chen Yan. Lp-based policies for restless bandits: necessary and sufficient conditions for (exponentially fast) asymptotic optimality. *arXiv:2106.10067*, 2021.

[15] Chen Yan and Alexandre Reiffers-Masson. Certainty equivalence control-based heuristics in multi-stage convex stochastic optimization problems. *arXiv:2308.13166*, 2023.

[16] Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

[17] Gauthier Gidel, Hugo Bérard, et al. A variational inequality perspective on generative adversarial networks. *arXiv:1802.10551*, 2020.

[18] Yi Chen, Jing Dong, and Zhaoran Wang. A primal-dual approach to constrained markov decision processes. *arXiv:2101.10895*, 2021.

[19] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[20] Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.

[21] S.R. Eshwar, L. Lopes Felipe, A. Reiffers-Masson, D.S. Menache, and G. Thoppe. Online Learning of Weakly Coupled MDP Policies for Load Balancing and Auto Scaling. *Technical report*, 2024. https://github.com/lucaslopes/lp-learner/blob/main/pdf/full-paper.pdf and https://github.com/lucaslopes/lp-learner.