Non-Stationary Gradient Descent for Optimal Auto-Scaling in Serverless Platforms

Jonatha Anselmi, Bruno Gaujal, Louis-Sébastien Rebuffi

***Abstract*—To efficiently manage serverless computing platforms, a key aspect is the auto-scaling of services, i.e., the set of computational resources allocated to a service adapts over time as a function of the traffic demand. The objective is to find a compromise between user-perceived performance and energy consumption. In this paper, we consider the *scale-per-request* auto-scaling pattern and investigate how many function instances (or servers) should be spawned each time an *unfortunate* job arrives, i.e., a job that finds all servers busy upon its arrival. We address this problem by following a stochastic optimization approach: we develop a stochastic gradient descent scheme of the Kiefer–Wolfowitz type that applies *over a single run of the state evolution*. At each iteration, the proposed scheme computes an estimate of the number of servers to spawn each time an unfortunate job arrives to minimize some cost function. Under natural assumptions, we show that the sequence of estimates produced by our scheme is asymptotically optimal almost surely. In addition, we prove that its convergence rate is $O(n^{-2/3})$ where n is the number of iterations.**

From a mathematical point of view, the stochastic optimization framework induced by auto-scaling exhibits non-standard aspects that we approach from a general point of view. We consider the setting where a controller can only get samples of the *transient* – rather than stationary – behavior of the underlying stochastic system. To handle this difficulty, we develop arguments that exploit properties of the mixing time of the underlying Markov chain. By means of numerical simulations, we validate the proposed approach and quantify its gain with respect to common existing scale-up rules.

***Index Terms*—Auto-scaling, serverless computing, parallel queueing system, stochastic optimization, Kiefer–Wolfowitz.**

I. Introduction

A. Auto-scaling in Serverless Computing

Auto-scaling mechanisms are considered to be essential components of serverless computing systems as they efficiently support cloud providers in handling the largest possible user base on their physical platforms. These mechanisms are designed to automatically adjust the current service capacity in response to the current load while ensuring that service level agreement (SLA) contracts are respected. In this paper, we tweak a popular auto-scaling paradigm that in the cloud computing literature is known as "per-request" [22], [5], [4] or "reactive" [14] auto-scaling. According to this paradigm, an incoming request (or *job*) is processed by an active idle function instance (or *server*) if there is any available, otherwise, the platform spawns a new server that will serve the job immediately after a *coldstart* latency. By design, the activation of a new server is only triggered at the arrival time of an *unfortunate* job, i.e., a job that finds no active idle servers upon its arrival and thus must wait. In practice, this is the de-facto auto-scaling pattern and is currently employed by serverless computing platforms such as AWS Lambda, Google Cloud Functions, Azure Functions, IBM Cloud Functions and Apache OpenWhisk.

J. Anselmi, B. Gaujal and L.-S. Rebuffi are with Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, 38000 Grenoble, France. Email: first-name.lastname@inria.fr

B. Addressed Problem

The current implementations of the auto-scaling paradigm described above operate under the assumption that *exactly one* server is spawned (or initialized) when an unfortunate job arrives [22]. The objective of this work is *to investigate whether or not it would be convenient to activate more than one server*, say $1+\theta$, instead of just one. It is worth noting that activating $\theta > 0$ extra servers results in increased energy consumption compared to the case where $\theta = 0$. On the other hand, this brings a performance benefit *proactively*, as future arrivals have a higher chance of finding active idle servers, thus avoiding the coldstart latency cost. This is particularly crucial for serverless or edge computing applications, where response time is critical to ensure optimal performance [14]. It is commonly recognized that even a minor rise of even a few milliseconds in latency can have a drastic effect on real-time applications such as e-commerce sales. Therefore, this paper aims to explore whether the activation of surplus servers at scale-up times ultimately leads to a more favourable balance between energy consumption and user-perceived performance.

C. Stochastic Optimization Framework

Several analytical performance models have been developed in the literature to evaluate the delay performance and power consumption induced by auto-scaling algorithms; see, e.g., [22], [17], [16]. The starting point of our work is the Markov model proposed in [22], which captures the unique details of several existing serverless computing platforms and is also tailored to the auto-scaling algorithm implemented in Amazon's AWS Lambda. We extend this model to the case where the platform spawns $\theta + 1$ servers at the moment of a job arrival if the job finds no active idle server – the model in [22] is recovered when $\theta = 0$.

To find the θ that minimizes a cost function that takes into account the blocking probability, i.e., the probability that a random job incurs a coldstart latency, and the energy consumption, we follow a stochastic optimization approach looking for an online learning algorithm. The main motivation for this approach is that some quantities such as the job arrival

rate are time-varying and unknown in advance. However, they are observable and can therefore be learned. Moreover, the specific structure of the cost function induced by the considered auto-scaling setting brings the additional difficulty that the *gradient* of the cost function is unknown as well, although again observable. This issue prevents us from relying on the class of Stochastic Gradient Descent (SGD) iterative schemes, which are common in stochastic optimization, and leads us to consider iterative schemes of the Kiefer-Wolfowitz type [9]; see below for further details.

To optimize over θ, an alternative approach would consist in modeling the problem via a Markov decision process (MDP) [24] and then learning the optimal policy via (variations of) algorithms such as the celebrated UCRL2 [6]. In the literature, MDP formulations for problems similar to ours have been developed recently in [11], [29]; see also the references therein. In contrast to our approach, these works assume full knowledge of the model parameters and no learning is considered. We do not follow the MDP reinforcement learning approach for two reasons. In general, the optimal policy is highly dependent on the system state and therefore not versatile. In our case, however, the optimal policy reduces to a one-dimensional parameter θ, indicating how many servers to activate upon job arrivals. Also, the size of the state space of the MDP is prohibitively large in our case, and this would make any model-based reinforcement learning algorithm impractical. We show in Section II, that the size of the state space is of the order of N^3 where N denotes the nominal number of servers that can be up and running, which for several applications is in the order of hundreds or thousands.

We also notice that the application of reinforcement learning for autoscaling of serverless applications is currently a bit underexplored [8], [2]. A Q-learning approach is considered in [1] and a comprehensive numerical evaluation of existing deep learning algorithms has been recently conducted in [2]. The downside of these works is that no theoretical properties about convergence and optimality are proven.

Finally, to minimize over θ, a further approach would consist of i) solving the global balance equations of the underlying Markov chain to get the stationary measure induced by a given θ and then ii) computing the optimal θ by binary search or relying on standard algorithms for deterministic optimization. This naive approach is not interesting either because it requires the knowledge of all the parameters that define the underlying Markov chain. As discussed above, we do not assume this knowledge.

D. Novelty of our Approach: Non-Stationary Samples

When trying to apply existing stochastic optimization techniques to the specific case of auto-scaling, the following technical difficulty appears. To fully grasp the root of the problem, let us formally introduce the general stochastic optimization framework under investigation. The objective is to minimize some real function $f(\theta) := \mathbb{E}[F(\theta, X)]$ over $\theta \in \mathbb{R}^p$, where X is some random variable over $\mathcal{X}$. The distribution of X and the mapping $F : \mathbb{R}^p \times \mathcal{X} \mapsto \mathbb{R}$ are unknown. However, it is allowed to get samples $X_1, X_2, \ldots$ and, thus, $F(\theta_1; X_1), F(\theta_2; X_2), \ldots$ for different values of θ_n. Here, $F(\theta_n, X_n)$ represents the random cost observed at time step n under the set of parameters θ. To find an optimal θ, one can only rely on such information. Under certain technical conditions, the Robbins–Monro and Kiefer-Wolfowitz algorithms are the classical iterative schemes that make the sequence θ_n converge to a minimum of f [9], [26]; see also [31]. Unfortunately, this type of approach can not be employed within our setting because our problem does not grant access to samples of X. In our case, X has the stationary distribution of a continuous-time Markov chain that models the dynamics of auto-scaling, and what we can only observe are (non-stationary) samples from the *transient* behavior of such chain; in practice, this corresponds to collecting observations from the up-and-running real system. This non-stationarity is the main technical difficulty that singles out our work from existing approaches; for further details, see Section II-C and Remark 1. To deal with this difficulty, we modify the standard Kiefer-Wolfowitz algorithm by introducing a new parameter that controls how long the system is observed for a given θ in order to obtain a non-stationary sample that is sufficiently close to the corresponding stationary distribution of the Markov chain parameterized by that given θ. While we can prove that the modified algorithm converges a.s. to the optimal value of θ (Theorem 1) with a state-of-the-art convergence rate in $O(n^{2/3})$ (Theorem 2), there is still a price to pay for non-stationarity:

- *Additional assumptions:* Assumption 2, which requires the underlying Markov chain to mix *uniformly*, is critical to our proof technique. It provides a means to control the accuracy of non-stationary samples and is, to some extent, necessary, as we demonstrate that without it, the desired convergence properties fail to hold. This is supported by numerical evidence.
- *Technical difficulty:* The proof for the convergence rate requires a truncation/extension of the control policy to ensure smoothness of the stationary policy with respect to θ.
- *Increased convergence time:* The convergence rate involves a term depending on the mixing time, more precisely $\log(1/\rho)$ where ρ is the uniform mixing rate.

The closest reference to our work is the classical work [23], which presents a scheme of the Kiefer-Wolfowitz type as in our setting. Under technical conditions, that scheme converges almost surely to a minimum of the cost function. However, no convergence rate is proven for that scheme. Another reference that is close to ours is [10]. Here, the authors consider a general iteration scheme for solving a stochastic optimization problem as in our setting, modulo some minor technical assumptions. The main differences with respect to our work are that their iterative scheme is of the Robbins-Monroe type and that their main result (Theorem 1) does not specify the convergence speed of the scheme towards the minimum of the cost function. More precisely, they show that it has a polynomial structure, but the exponent depends on a parameter, α, related to the Lipschitz constant of the average cost, which may be difficult to get depending on the application considered.

We also mention a number of related works that have recently

appeared in the literature to address settings similar to ours [28], [15], [20], [19], [25], i.e., where samples of X are not available. These propose SGD methods where approximate samples of X are taken on the trajectory of a Markov chain, as we do within our approach. In particular, the algorithm proposed in [28] has nice convergence properties even in the case of non-convex cost functions and non-reversible Markov chains, which is the setting considered in this paper. The crucial difference between all these works and ours is that they all assume the knowledge of the gradient of the cost function f. In our case, this information is not available as it depends on the unknown transition rate matrix of the underlying Markov chain.

E. Summary of our Contribution

First, we model the job and server dynamics induced by our auto-scaling mechanism in terms of a Markov chain that generalizes the one recently proposed in [22]. This is parameterized by θ, which defines a set of auto-scaling policies and may be interpreted as a *reserve* of extra servers that are ready to be used – the model and the algorithm discussed in [22] are recovered if $\theta = 0$. The parameter θ is under the control of the system manager and, following the stochastic optimization approach discussed above, the aim is to design an iterative scheme capable of making θ to converge to θ^*, i.e., the reserve size that minimizes some cost function. lthough our optimization framework is inspired by auto-scaling, we look at the problem from a larger perspective than auto-scaling and propose a general iterative scheme, see Algorithm 2, which is an adaptation of the celebrated Kiefer-Wolfowitz scheme. Under natural conditions, in Theorem 1, we show that the sequence of scalars generated by the proposed algorithm converges almost surely to a minimum of the cost function of interest, and in Theorem 2, we show that the convergence rate is $O(n^{-2/3})$, where n denotes the number of iterations on θ. Finally, we apply the proposed algorithm to the special case of auto-scaling. By means of numerical simulations, we validate the proposed approach and quantify the cost function relative gains with respect to the common scale-up rule where $\theta = 0$. Within a realistic parametrization of our problem, we show that $\theta^* \approx 6$ and that the proposed algorithm indeed generates a sequence $(\theta_n)_n$ that converges to such a value, yielding relative gains of around 5-8%.

F. Organization

This paper is organized as follows. Section II introduces a Markov model for the considered auto-scaling system and formalizes the stochastic optimization problem of interest. Section III defines our non-stationary Kiefer–Wolfowitz algorithm (Algorithm 2) and presents our main results (Theorems 1 and 2). We stress that this algorithm is general and that it can be applied outside the auto-scaling framework introduced in Section II. Finally, Section IV is dedicated to the application of Algorithm 2 in the context of auto-scaling. Here, we validate its behavior and evaluate its performance numerically.

II. Framework

A. System Description and Auto-scaling Algorithm

We consider an architecture composed of N parallel servers; in serverless computing, servers are also called function instances. These represent the nominal service capacity, i.e., the upper limit on the number of servers that one user of the platform can have up and running at the same time. To ensure service availability for other users, existing serverless platforms require the specification of such limit [22].

Each server can be in one of the following three macro states: *warm* if turned on, *cold* if turned off, and *initializing* if making the transition from cold to warm. An initializing server cannot process jobs yet as it performs basic startup operations such as connecting to database, loading libraries, etc. We also say that a server is *idle-on* if it is warm but not processing any job, and *busy* if it is warm and processing some job. For our purposes, it is convenient to split the set of initializing servers into two groups, say $init_0$ and $init_1$. Init-1 servers are initializing servers that are already bound to a job, i.e., the job that triggered their activation. Upon finishing their initialization phase, they process their associated job immediately. Init-0 servers are initializing servers that are not necessarily bound to any job. Upon finishing their initialization phase, they become either idle-on or busy depending on whether a job is blocked in the queue. Warm servers can make the transition to cold only if they are idle-on. Figure 1 summarizes the possible transitions among the server states.

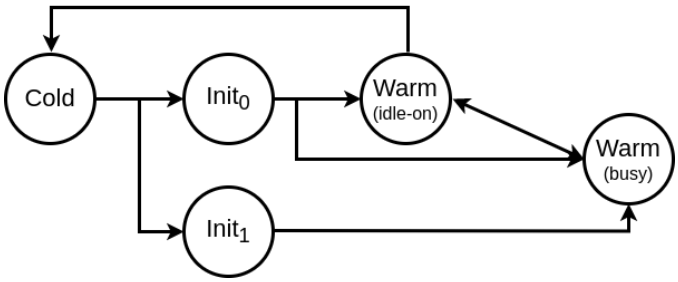

Figure 1. State transitions for each server.

Jobs join the system from an exogenous source to receive service. Upon each job arrival:

- if an idle-on server exists, then the job is sent and processed by an idle-on server selected uniformly at random;
- if all servers are either busy or init_1, all resources are saturated and the job is rejected;
- otherwise, a certain number of cold servers is selected uniformly at random to become initializing, and in the meanwhile the job waits in a central queue for the activation of its init_1 server. that is again selected upon arrival of the job itself.

In the central queue, jobs wait following the first-come first-served scheduling discipline, and each job leaves the system upon completion at its designated server. We notice that scale-up decisions are taken at job arrival times and by a central monitor, which at any point in time has full knowledge of the server states. In the literature, this auto-scaling mechanism

is called "scale-per-request" [22], [5]. On the other hand, the decision of making a server cold is taken by the server itself after an *expiration time* if during such time the server received no job. This scale-down rule is well consolidated in practice [32], [30], [16] and it will be assumed in the following.

Since the scale-down rule is fixed, we focus on the design of a scale-up rule. As commented in the Introduction, current scale-up rules initialize at most one server for each job arrival. The novelty of our approach consists in providing more flexibility in this respect, and this flexibility is captured by controlling the number of init_0 servers. More precisely, Algorithm 1 defines our auto-scaling algorithm.

Algorithm 1: The proposed auto-scaling algorithm.

Input: $\theta \in \mathbb{R}_+$, "system state"
Output: Number of servers to initialize:
#init_0_servers:=0 and #init_1_servers:=0

```
for each job arrival do
    #init_0_servers:=0
    #init_1_servers:=0
    if #idle-on_servers=0 and #cold_servers>0 then
        #init_1_servers:=1
        #init_0_servers:=π_θ("system_state")
end
```

Let us elaborate a bit more on Algorithm 1. First, servers are only initialized if there is no idle-on server. In this case, the number of init_1 servers to activate is always set to one, because by definition this init_1 server will be the server that will process the incoming job. Then, the number of init_0 servers to activate is given by the "black box" Π_θ. The form of this function is not important for now and will be given in Section IV-D. What should be retained is that it depends on the system state and on the parameter $\theta \in \mathbb{R}_+$, i.e., the design parameter that will be the subject of stochastic optimization. We anticipate that the idea behind our scale-up rule π_θ will be to choose #init_0_servers to ensure that an average reserve of θ servers is ready for use at least in the short future. Within this interpretation, if $\theta = 0$ then no init_0 servers exist and Algorithm 1 boils down to the auto-scaling algorithm investigated in [22].

B. Markov Model

We introduce a continuous-time Markov chain that models the dynamics induced by the system described above.

We assume that jobs join the system following a Poisson process with rate λ. Service, initialization and expiration times are exponentially distributed random variables with rate μ, β, γ, respectively. The four sequences of job inter-arrival, service, initialization and expiration times are i.i.d. and independent of each other.

Let $\mathcal{X} := \{x = (x_1, x_2, x_3, x_4) \in \mathbb{N}^4 : \sum_{i=1}^3 x_i \leq N, x_4 \leq x_3\}$. A Markovian representation of the system dynamics is captured by the state variable $x \in \mathcal{X}$ where x_1, x_2 and x_3 represent the number of idle-on, busy and initializing (both init_0 and init_1) servers, respectively, and x_4 represents the number of init_1 servers only. Note that the number of init_1 servers can be equivalently interpreted as the number of *blocked* jobs, i.e., the jobs that are waiting in the central queue. Thus, the number of cold servers in state x is $N - \sum_{i=1}^3 x_i$.

Let e_i be the size-4 unit vector in direction i.

The Markov chain $(X_t)_t$ that describes the dynamics of the proposed auto-scaling algorithm is defined by the transition matrix

$$Q_{x,x'} = \begin{cases} \lambda \mathbb{I}_{\{x_1>0\}} & \text{if } x' = x - e_1 + e_2 \\ \lambda \mathbb{I}_{\{x_1=0\}} & \text{if } x' = x + \pi_\theta(x) e_3 + e_4 \\ \mu x_2 & \text{if } x' = x + (e_1 - e_2)\mathbb{I}_{\{x_4=0\}} \\ & \qquad - e_4 \, \mathbb{I}_{\{x_4>0\}} \\ \gamma x_1 & \text{if } x' = x - e_1 \\ \beta x_3 \mathbb{I}_{\{x_4>0\}} & \text{if } x' = x + e_2 - e_3 - e_4 \\ \beta x_3 \mathbb{I}_{\{x_4=0\}} & \text{if } x' = x + e_1 - e_3 \end{cases} \tag{1}$$

for all states $x, x' \in \mathcal{X}$, with $x' \neq x$, where $\pi_\theta(x)$ represents the scale-up policy, i.e., the number of servers to initialize upon job arrival and when the system is in state x with $x_1 = 0$ and $N - \sum_{i=1}^3 x_i > 0$. Again, the precise form of $\pi_\theta(x)$ will be given in Section IV-D.

C. Stochastic Optimization Problem

Our objective is to find θ that minimizes a trade-off between user-perceived performance and energy consumption, and as discussed in the Introduction, we follow a stochastic optimization approach. To investigate this trade-off, we first define the instant random cost:

$$C(x) := \sum_{i=1}^4 w_i x_i + \mathbb{I}_{\{x_2+x_4=N\}} w_{\text{rej}}, \tag{2}$$

where w_i, for $i = 1, \ldots, 4$, and w_{rej} are positive weights; here, for instance, w_{rej} is the weight for rejecting a job because $\mathbb{I}_{\{x_2+x_4=N\}}$ represents the event that the system contains N jobs or, equivalently, that all the N servers are used. This cost function is known to the optimizer. However, we are interested in finding θ that minimizes the average long-run cost in a setting where *the transition rate matrix Q in* (1) *is unknown but the states of the underlying Markov chain, $(X_t)_t$, can be observed* by the optimizer. More precisely, we want to develop an iterative scheme able to find $\theta \in \mathbb{R}_+$ that minimizes

$$c(\theta) := \mathbb{E}[C(X_\infty^\theta))] \tag{3}$$

where X_∞^θ is a random variable having the invariant distribution of $(X_t)_t$, which exists because it is ergodic and depends on θ because of (1). In addition, we require that the iterative scheme that we look for should be potentially implemented in a real system. To achieve this goal, the optimizer can only use samples from a *single run* of the Markov chain and adjust the value of θ on the fly.

Remark 1. *The structures of* (2) *and the cost function described in Section I-D are different in the sense that* (2) *does not depend on θ* directly. *It only depends on θ via X_∞^θ. This poses the technical difficulty that $\frac{\mathrm{d}}{\mathrm{d}\theta} C$ is not accessible and rules out the application of a large class of stochastic gradient descent algorithms based on stochastic approximations with Markovian noise (see for example [3], [10]).*

Remark 2. *If one could sample directly from X_∞^θ, then the optimization problem under investigation could be solved via the celebrated Kiefer–Wolfowitz algorithm which, under certain conditions, can provide a sequence of θ's converging to a minimum of $c(\theta)$. However, as discussed in the introduction, we have only access to (transient) samples of $(X_t)_t$ rather than X_∞^θ and this makes our problem more difficult.*

III. A Non-stationary Kiefer–Wolfowitz Algorithm

In this section, we propose a general stochastic descent algorithm that solves a general optimization problem. The relationship with auto-scaling as formulated previously is quite intuitive but it will be formally explained in Section IV. Here, we consider a quite general parametric finite Markovian system, with parameter denoted $\theta \in \mathbb{R}$. The finite state space is denoted by $\mathcal{X}$ and the transition matrix under θ is P_θ. The aim is to design an online learning algorithm that computes the optimal value θ^*, which minimizes an expected cost under the *stationary* regime of the Markov chain.

In the following, we denote by f the function to minimize, defined as

$$f : \mathbb{R} \to \mathbb{R}$$
$$\theta \mapsto \mathbb{E}\left[F(\theta, X_\infty^\theta)\right]$$

where $F(\theta, x)$ is the cost under state x and parameter θ. Here, X_∞^θ is a random state whose distribution is m_θ, the stationary distribution of the Markov chain with parameter θ.

The rest of this section is organized as follows. Section III-A introduces our online algorithm; Section III-B shows that under some regularity assumptions on the cost, our algorithm converges to the optimal parameter θ^* almost surely (Theorem 1). In addition, a state-of-the-art convergence rate in $O(n^{-2/3})$ is proved in Theorem 2.

A. Description of the Algorithm

Our non-stationary Kiefer–Wolfowitz (KW) algorithm is based on the following idea. After each policy update, from episode $n-1$ to n, the classical KW algorithm needs to sample two independent states from the stationary measures of the Markov chain with parameters $\theta_n + \delta_n$ and $\theta_n - \delta_n$, where δ_n is the stepsize that is needed to approximate the gradient of the cost function at θ_n. However, we do not have access to these stationary measures. Instead, we simulate one Markov chain over τ_n timesteps twice to reach two states close to stationarity, where τ_n is related to the mixing time of the Markov chain and scales in $\log n$.

We start from the initial state x_{start} and initial policy θ_0. Denote by T_n the number of timesteps at the end of episode $n-1$. For each episode n, we also choose x_{start} to be the initial state of the trajectories we will simulate.

From T_n to $T_n + \tau_n - 1$, we first simulate the Markov chain with initial state x_{start} and parameter $\theta_n + \delta_n$, and observe the states $X_{T_n, T_n+i}^{\theta_n+\delta_n}$ for $i = 1, \dots, \tau_n$, and the random cost $F(\theta_n+\delta_n, X_{T_n, T_n+\tau_n}^{\theta_n+\delta_n})$. We reiterate this process K times, and do similar simulations for the Markov chain with parameter $\theta_n - \delta_n$. We then evaluate the average cost of the Markov chain with parameters $\theta_n + \delta_n$ and $\theta_n - \delta_n$ with the samples $X_{T_n+i\tau_n, T_n+(i+1)\tau_n}^{\theta_n+\delta_n}$ and $X_{T_n+(K+i)\tau_n, T_n+(K+i+1)\tau_n}^{\theta_n-\delta_n}$ for $i = 0, \dots, K-1$, from which we may approximate the derivative of the cost function at θ_n, and eventually compute the following update of the parameter θ:

$$\theta_{n+1} = \theta_n - \gamma_n \frac{\hat{f}_n(\theta_n + \delta_n) - \hat{f}_n(\theta_n - \delta_n)}{2\delta_n}, \tag{4}$$

where γ_n is the stepsize of the parameter update, and

$$\hat{f}_n(\theta_n + \delta_n) = \frac{1}{K} \sum_{i=0}^{K-1} F\left(\theta_n + \delta_n, X_{T_n+i\tau_n, T_n+(i+1)\tau_n}^{\theta_n+\delta_n}\right), \tag{5a}$$

$$\hat{f}_n(\theta_n - \delta_n) = \frac{1}{K} \sum_{i=0}^{K-1} F\left(\theta_n - \delta_n, X_{T_n+(K+i)\tau_n, T_n+(K+i+1)\tau_n}^{\theta_n-\delta_n}\right). \tag{5b}$$

This process is formalized in Algorithm 2.

Algorithm 2: Non-stationary Gradient Descent Algorithm.

Input: γ_n step-size sequence, δ_n discretization step, initial parameter θ_0 and state x_{start}, T the total simulation time and a parameter τ.

1 Set $n = 0$ the algorithm time-step and $t = 0$ the current simulation time-step, $x = x_{\text{start}}$ the initial state.
2 **while** $t \leq T$ **do**
3 $\tau_n = \tau \log(n+1)$
4 **for** *the simulation number* $i = 0, 1, \dots, K-1$ **do**
5 Simulate the Markov chain starting at x_{start} with parameter $\theta_n + \delta_n$ over τ_n steps, by choosing at each step of the simulation the action $\hat{\theta}$ randomly between $\lfloor \theta_n + \delta_n \rfloor$ and $\lfloor \theta_n + \delta_n \rfloor + 1$.
6 Repeat this process overall K times, and observe the empirical average reward and the average visit count for each state.
7 Repeat this process for the parameter $\theta_n - \delta_n$.
8 **end**
9 Compute the average over the K simulations to obtain $\hat{f}_n(\theta_n + \delta_n)$ and $\hat{f}_n(\theta_n - \delta_n)$.
10 Update the empirical stationary measure: the number of visits of a given state under any parameter, divided by the total number of visits.
11 Compute the parameter update (4): $\theta_{n+1} = \theta_n - \gamma_n \frac{\hat{f}(\theta_n+\delta_n) - \hat{f}(\theta_n-\delta_n)}{2\delta_n}$.
12 $t := t + 2K\tau_n$ and $n := n + 1$
13 **end**

In the loop in Line 4, we simulate the trajectory induced by $\theta_n + \delta_n$ before the one induced by $\theta_n - \delta_n$. Because of the memoryless property, this does not affect our results.

B. Convergence Results

We will use the following assumptions.

Assumption 1 (Regularity of the cost function)**.**
(1.a) $f : \mathbb{R} \to \mathbb{R} \in \mathcal{C}^3$,

(1.b) f' is Lipschitz,

(1.c) the cost function F can be written as $F(\theta, x) = G(\theta, x) + b(\theta)$, where b is a penalty function such that $|b(\theta+\varepsilon) - b(\theta-\varepsilon)| \leq C_2(\theta-\theta^*)\varepsilon$ for $0 < \varepsilon < \varepsilon_0$, where C_2 and ε_0 are positive constant, and G is positive and bounded by $G_{\max}$ and θ^* is a minimum of f.

(1.d) There exists $L > 0$ and $r > 0$ such that $f'(\theta) > r$ for $\theta > L$, and $f'(\theta) < -r$ for $\theta < -L$.

While Assumption 1 is quite technical and may look restrictive, it is satisfied in the autoscaling case by simple inspection of the cost function. This will be shown in Section IV.

Assumption 2 (Uniform mixing time)**.** *Let P_θ be the transition matrix of a Markov chain indexed by the parameter $\theta \in \mathbb{R}$. Let x be an initial state and m_θ denote the corresponding stationary measure within parameter θ. There exists $C_1 > 0$ and $\rho < 1$ independent of θ such that*

$$\left\| P_\theta^t(x, \cdot) - m_\theta \right\|_1 \leq C_1 \rho^t$$

for any θ and t.

The previous assumption is not common in stochastic gradient descent algorithms because standard approaches assume to be able to sample from a stationary distribution. Indeed, in our framework it relates the transient regime that we can observe over time to stationary properties of the system.

Assumption 3 (Uniqueness)**.** *The function f has a unique minimum θ^*.*

Assumption 4 (Strong convexity)**.** *The function f is strongly convex: for some $\kappa > 0$, for any $\theta, \theta' \in \mathbb{R}^+$, it holds that*

$$f'(\theta)(\theta - \theta') \geq \kappa(\theta - \theta')^2. \tag{6}$$

We can now state our main results.

The next theorem states the almost sure convergence of the sequence θ_n produced by the proposed algorithm, under the following parametrization: The sequences $(\gamma_n)_n$, $(\delta_n)_n$ and $(\tau_n)_n$ are such that:

$$\lim_{n\to\infty} \delta_n = 0, \qquad \lim_{n\to\infty} \tau_n = +\infty \tag{7a}$$

$$\sum_n \gamma_n = \infty, \qquad \sum_n \gamma_n^2 \delta_n^{-2} < \infty \tag{7b}$$

and

$$(\gamma_n)_n, (\delta_n)_n \text{ and } \left(\frac{\gamma_n}{\delta_n}\right)_n \text{ are decreasing.} \tag{8}$$

Theorem 1. *Let $(\theta_n)_n$ be the sequence of random variables generated by Algorithm 2 with parametrization* (7)-(8)*. Under Assumptions 1, 2, and 3, we have*

$$\theta_n \overset{a.s.}{\to} \theta^*. \tag{9}$$

The next theorem provides a result on the convergence rate to the minimizer of f.

Theorem 2. *Let Assumptions 1, 2, 3 and 4 hold. Under well-chosen parameters $\delta_n = n^{-2/3}$, $\gamma_n = n^{-1}$ and $T_n = \alpha \frac{\log n}{\log 1/\rho}$ with $\alpha > 1$, and $\gamma_0 < \frac{4\kappa}{C_2}$, Algorithm 2 converges to the minimum θ^* with asymptotic rate:*

$$\limsup_{n\to\infty} \mathbb{E}\left[(\theta_n - \theta^*)^2\right] n^{2/3} \leq \frac{\left(2C_0 + \sqrt{2}G_{\max}C_1^{1/2}\right)^2}{8\kappa^2} + \frac{G_{\max}^2}{2\kappa}.$$

Theorem 2 implies that the convergence rate of the sequence θ_n produced by Algorithm 2 is $O(n^{-2/3})$, provided that its input parameters are properly tuned, which is as good as the state-of-the-art convergence rate of classical KW algorithms [31]. The detailed proof is postponed to Appendix A.

IV. Application to Auto-scaling

In this section, we discuss the applicability of Theorems 1 and 2 to the auto-scaling problem modeled in Section II. The main point is to construct a cost function f satisfying Assumptions 1 whose minimum coincides with the minimum of the function c defined in (3).

This construction is not unique and several choices made in the following could certainly be changed, especially to improve the performance of the algorithm in practice (see Section IV-D).

A. Truncation/Extension: Construction of the Scale-up Rule

The first step is to construct a scale-up rule π_θ that maps any real parameter θ into a number of servers in $[0, N]$. The idea is as follows: see θ as the average amount of servers to turn on. A simple choice would be to sample π_θ from a binomial law with parameters $(N, \theta/N)$. This would be possible for $\theta \in (0, N)$. However, to comply with the optimization algorithm given in Section III, θ must live on the whole real space $\mathbb{R}$.

For that, we construct an explicit mapping from $\mathbb{R}$ to $[\varepsilon, M - \varepsilon]$, with $\varepsilon > 0$, and $M < N$. Here, ε must be seen as a very small parameter whose role is only to get a smooth transition from $\mathbb{R}$ to $[0, M]$. As for the choice of $M < N$, it will be explained in the next subsection.

First, define the following smooth step function $\psi_{a,b}$, for $a < b \in \mathbb{R}$:

$$\psi_{a,b} : x \in \mathbb{R} \mapsto \begin{cases} 0 & \text{if } x < a \\ \exp\left(-\frac{(b-x)^2}{x-a}\right) & \text{if } a \leq x \leq b \\ 1 & \text{if } b \leq x. \end{cases}$$

This function is equal to 0 on $(-\infty, a]$, equal to 1 on $[b, +\infty)$, and is smooth on $\mathbb{R}$. Using this function, for $\varepsilon > 0$, we define

$$\theta_{\varepsilon,M} : \theta \in \mathbb{R} \mapsto \begin{cases} h_-(\theta), & \text{if } \theta < 0 \\ h_-(\theta)\left(1 - \psi_{0,\varepsilon}(\theta)\right) + \theta\psi_{0,\varepsilon}(\theta), \\ \quad \text{if } 0 \leq \theta \leq \varepsilon \\ \theta, & \text{if } \varepsilon < \theta < M - \varepsilon \\ \theta\left(1 - \psi_{M-\varepsilon,M}(\theta)\right) + h_+(\theta)\psi_{M-\varepsilon,M}(\theta), \\ \quad \text{if } M - \varepsilon \leq \theta \leq M \\ h_+(\theta), & \text{if } M < \theta, \end{cases}$$

where $h_-(\theta) = \frac{\varepsilon}{3}\exp\left(\frac{\theta}{\varepsilon}\right)$ and $h_+(\theta) = M - \frac{\varepsilon}{3}\exp\left(-\frac{\theta - M}{\varepsilon}\right)$. We display a representation of this function for $M = 10, \varepsilon = 0.5$ in Figure 2. Here, $\theta_{\varepsilon,M}$ is adapted from θ and has the following properties:

- $\theta_{\varepsilon,M} \in [0, M]$,

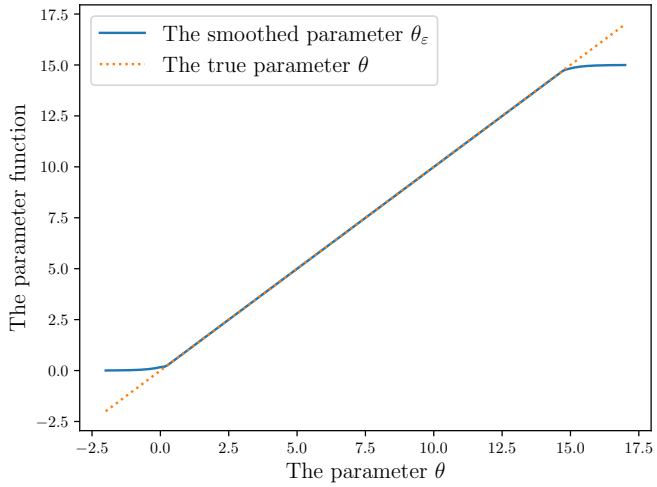

Figure 2. The smoothed parameter function $\theta_{\varepsilon,M}$ compared with θ

- For $\theta \in [\varepsilon, M-\varepsilon]$, $\theta_{\varepsilon,M} = \theta$,
- $\theta_{\varepsilon,M} \to 0$ as $\theta \to -\infty$,
- $\theta_{\varepsilon,M} \to M$ as $\theta \to +\infty$.

We will therefore be able to sample from a binomial law with parameters $(M, \theta_{\varepsilon,M}/M)$. At any state x, the number of servers to turn on $\pi(x)$ follows a binomial law with parameters $(M, \theta_{\varepsilon,M})$.

B. Smoothness

Since the mapping $\theta \to \theta_{\varepsilon,M} \to P_{\theta_{\varepsilon,M}}$ defined in the previous subsection only constructs irreducible matrices, the set of recurrent states of the Markov chain remains unchanged as θ gets updated. This implies that the rank of $P_{\theta_{\varepsilon,M}}$ is constant for all $\theta \in \mathbb{R}$.

By construction, we also have for all $\theta \in \mathbb{R}$, $\theta \mapsto P_{\theta_{\varepsilon,M}} \in \mathcal{C}^3$ (actually, it is in C^k for all $k > 0$).

These two properties imply that the stationary measure is also smooth, which we will prove by proving regularity properties of the pseudoinverse matrices related ot the transition matrices $P_{\theta_{\varepsilon,M}}$.

The Drazin inverse of a matrix is defined as follows.

Definition 1 ([12]). *Let A be a square real-valued matrix. The Drazin inverse exists and is the unique solution X to the following equations:*

$$\begin{cases} AX &= XA, \\ A^k &= A^{k+1}X \quad \textit{for some positive integer } k, \\ X &= X^2 A. \end{cases}$$

The smaller k for which these equations still hold is called the Drazin index of A. We will denote the Drazin inverse of A by $A^\#$.

We will be able to relate f to the Moore-Penrose pseudo inverse of a matrix, also called generalized inverse, which will be denoted by the superscript †. Regularity properties of this pseudo inverse will be proven using the following lemmas.

Lemma 1 (Theorem 4.3, [18]). *Let $\theta \mapsto A_\theta$ be a Fréchet differentiable square matrix function with local constant rank in $\mathbb{R}$. We denote by $\mathbf{D}A_\theta$ the Fréchet-derivative at θ and we write the following equation as a function from $\mathbb{R}$ to $\mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$. For any $\theta \in \mathbb{R}$:*

$$\mathbf{D}A_\theta^\dagger = -A_\theta^\dagger \mathbf{D}A_\theta A_\theta^\dagger + A_\theta^\dagger A_\theta^{\dagger\top} \mathbf{D}A_\theta^\top P_{A_\theta}^\perp + {}_{A_\theta}P^\perp \mathbf{D}A_\theta^\top A_\theta^{\dagger\top} A_\theta^\dagger, \tag{10}$$

where $P_{A_\theta}^\perp := I - A_\theta A_\theta^\dagger$ is the projector on the orthogonal complement of the space spanned by the columns of A_θ, and ${}_{A_\theta}P^\perp := I - A_\theta^\dagger A_\theta$ is similarly defined for the space spanned by the rows of A_θ.

In the previous lemma, the Fréchet derivatives can be directly seen as matrix functions $\mathbb{R} \to \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$ rather than linear applications.

We can now state and prove the following regularity property on the cost function.

Proposition 1. *The function $\theta \mapsto m_{\theta_{\varepsilon,M}}(x)$ is C^∞.*

Proof. Let $x \in \mathcal{X}$ be any state. Consider a Markov reward process with rewards $r(x) = 1$ and $r(y) = 0$ for $y \neq x$, with the same transitions $P_{\theta_{\varepsilon,M}}$. This Markov chain is unichain and the gain at x is equal to $m_\theta(x)$, the stationary measure at x. It can be written with the Cesàro-limit:

$$m_\theta(x) = \mathbf{e}_x^\top P_{\theta_{\varepsilon,M}}^\infty \mathbf{e}_x,$$

where $P_{\theta_{\varepsilon,M}}^\infty := \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^T P_{\theta_{\varepsilon,M}}^t$ is the limiting matrix and $\mathbf{e}_x$ is the vector equal to 1 at x and 0 everywhere else. With [24, Theorem A.7], we relate the limiting matrix P_θ^∞ to the Drazin inverse $(I - P_{\theta_{\varepsilon,M}})^\#$ of $(I - P_{\theta_{\varepsilon,M}})$. Letting I denote the identity matrix:

$$P_{\theta_{\varepsilon,M}}^\infty = I - (I - P_{\theta_{\varepsilon,M}})(I - P_{\theta_{\varepsilon,M}})^\#.$$

Let us now call $A_\theta := I - P_{\theta_{\varepsilon,M}}$, and denote by k the Drazin index of A_θ, as defined in Definition 1. We use [12, Theorem 5] to relate the Drazin inverse to the generalized pseudoinverse in the following way:

$$A_\theta^\# = A_\theta^k (A_\theta^{2k+1})^\dagger A_\theta^k. \tag{11}$$

In order to prove the regularity of f, we therefore need to show that the function $\theta \mapsto A_\theta^\dagger$ is in C^3 itself. To prove it, recalling that P_θ and therefore A_θ is of constant rank over $\mathbb{R}$, we can use Lemma 1 to explicitly get the derivative of $\theta \mapsto A_\theta$, which itself is continuous, using the continuity of the generalized inverse ([27, Theorem 5.2]), as the rank of A_θ is constant. We can then use equation 10 to iterate the derivation process and prove that all the derivatives of $\theta \mapsto A_\theta^\dagger$ exist and are continuous. Using equation 11, we finally get that $\theta \mapsto A_\theta^\#$ is in C^∞, and therefore $\theta \mapsto m_{\theta_{\varepsilon,M}}(x)$ is C^∞. □

C. Truncation of the Parameter Domain and Penalty Function

We are now ready to construct the functions f from the autoscaling costs c and C. Let $M < N$ be any truncation of the parameter space.

Set $f_1(\theta) = \mathbb{E}C(X_\infty^{\theta_{\varepsilon,M}})$. We can thus rewrite this function as

$$f_1(\theta) = \sum_{x\in\mathcal{X}} C(x) m_{\theta_{\varepsilon,M}}(x) \tag{12}$$

This function is defined on $\mathbb{R}$ and is C^∞. In the following, we assume that there exists $M < N$ such that this smooth function has a unique minimum inside $(0, M)$ and is strongly convex in $(0, M)$. Such an interval is displayed in red in Figure 3 (left).

The last step is to construct a penalty function to extend the cost function unboundly outside $[0, M]$. Let

$$b(\theta) := (1-\psi_{0,\varepsilon}(\theta))(\theta-\varepsilon)^2+\psi_{M-\varepsilon,M}(\theta)(\theta-M+\varepsilon)^2. \quad (13)$$

We can now construct the function f as follows: $f(\theta) := f_1(\theta) + b(\theta)$.

- By construction of the smooth junction between f_1 and b, f is also C^∞ over $\mathbb{R}$ so it satisfies Assumption (1.a). By construction, the cost function f_1 is $\mathcal{C}^3$ on the compact $[0, M]$. This implies that f_1' is Lipschitz over $[0, M]$ and bounded over this interval. Since the penalty is quadratic outside $[0, M]$, f' is also Lipschitz over $\mathbb{R}$. (Assumption (1.b)) The decomposition of f into f_1 and b also implies Assumption (1.c) Assumption (1.d), as f_1' is bounded.
- As for Assumption 2, this is a direct consequence of the fact that $\theta_{\varepsilon,M}$ lives in the compact $[0, M]$ and P_θ^t as well as m_θ are continuous functions of θ.
- Finally, Assumptions 3 and 4 are true for f as soon as one can find a small enough interval $[0, M]$ where f_1 is strongly convex and has a unque minimum in the interior of this interval. This is checked numerically in the following section.

The final point is to notice that c and f coincide over $[\varepsilon, M - \varepsilon]$ and therefore have the same minimum θ^* in this interval.

D. Numerical Evaluation

By means of numerical simulations, we now evaluate the convergence properties of Algorithm 2 when applied to the proposed auto-scaling algorithm (Algorithm 1). Here, we use a simplified form (compared with Section IV-A) for the scale-up rule $\pi_\theta(x)$ and we set $\pi(x)$ to

$$\min\left\{(\lfloor\theta\rfloor + \mathbb{I}_{\{V<\theta-\lfloor\theta\rfloor\}} - x_3 + x_4)^+,\, N - x_2 - x_3 - 1\right\} \quad (14)$$

where V is an independent random variable uniformly distributed over [0,1]. Roughly speaking, since the number of init_0 servers (i.e., $x_3 - x_4$) is the number of servers that will be available to use in the short future, we let this number be $\theta - (x_3 - x_4)$. The randomization in V is used because the number of servers to activate must by an integer but we allow θ to be a real number because the optimal number of servers to initialize may not be an integer *in average*. Finally, the $\min$ and $(\cdot)^+ := \max\{\cdot, 0\}$ operators simply ensure that boundary conditions are satisfied.

Remark 3. *If $\theta = 0$, then no init_0 servers exist, which implies $x_4 = x_3$ at all times, and the Markov chain under investigation (defined by* (1)*) boils down to the Markov chain investigated in [22].*

In our simulations, we consider the following parametrization:

1) For the parameters that define the underlying Markov chain, we let $N = 50$, $\lambda \in \{0.15, 0.3\}$ (arrival rate), $\mu = 1$ (service rate), $\beta = 0.1$ (initialization rate) and $\gamma = 0.01$ (expiration rate). If a time unit is 10 milliseconds, these values are realistic for serverless applications [22], [13].
2) For the parameters that define the cost function (2), we let $w_{\text{rej}} = 10^3$, $w_1 = w_2 = 1$, $w_3 = 5$ and $w_4 = 100$. Note that $w_3 \geq \max\{w_1, w_2\}$ because initializing servers perform a batch of operations (connecting to database, loading libraries and data, etc.) and these are very expensive from the point of view of power consumption. Also, to make user-perceived performance and energy consumption comparable, we choose w_4 and w_{rej} to be significantly greater than w_3.
3) For the parameters that are used by Algorithm 2, in accordance with the assumptions in Theorems 1 and 2, we choose $\gamma_n = 10/n$, $\delta_n = n^{-2/3}$, $\tau = 10^6$, $\theta_0 \in \{1, 10\}$, $K = 2$ and $T = 10^8$.

Figure 3 (left) plots the cost function $c(\theta)$ (defined in (3)), which plays the role of $f(\theta)$ in Section III. It admits a unique minimum, θ^*, and it is strongly convex on a neighbourhood θ^*; here, the red vertical line is used to mark the convexity region postulated in Section IV-C. These properties have also been tested numerically over a wide range of parameters, and this is in agreement with the approach discussed in Section IV-C. For the plots in the figure, $\theta^* \in [5, 6]$, which means that initializing 6 or 7 servers is much better than initializing 1 server as in [22]; recall that θ is the number of *extra* server to initialize. From the figure, we observe the potential gain in activating the optimal surplus of servers amounts to 5-8%.

Figure 3 (right) plots the sequence θ_n produced by Algorithm 2 from different initial conditions. All trajectories converge to θ^*. Thus, the proposed algorithm improves with respect to the existing scale-up rule of no activating extra servers. Importantly, we notice that the trajectories corresponding to $\lambda = 0.15$ converge much faster than the trajectories corresponding to $\lambda = 0.3$. This quantifies the impact of the mixing time of the underlying Markov chain, which in our framework is captured by Assumption 2: the smaller the λ, the smaller the mixing time and therefore the easier the sampling close to stationarity.

E. Comparing with Fast θ-updates

In Algorithm 2, the reserve size parameter θ is updated only after the system has been observed for a sufficiently long period, ensuring that its dynamics approximate its stationary behavior. This observation time, linked to the mixing time of the underlying Markov chain, is controlled by the variable τ_n. An alternative approach could involve updating θ on the same timescale as the Markov chain, "without waiting for stationarity". In other words, θ is be updated after the system undergoes a fixed number of state transitions, such as 10^2 or 10^3, since the last update. As discussed in the Introduction, this stochastic-approximation approach has been already considered in the literature, e.g., [3], though we stress that the existing theoretical results do not apply to our case. Thus, we cannot expect that this alternative approach makes θ converge to the desired optimal point θ^*.

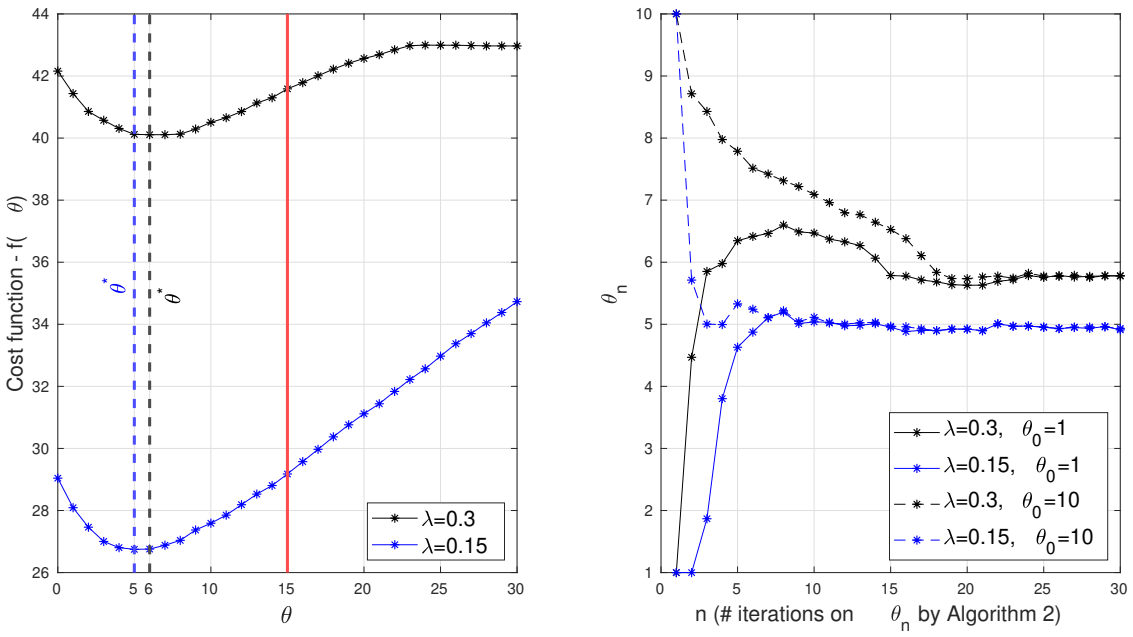

Figure 3. Plots of the cost function $f(\theta)$ (left) and of the sequences produced by Algorithm 2 (right). The red line corresponds to the truncation to the interval $[0, M]$ over which the function is convex.

The goal of this section is to evaluate this alternative approach within the same simulation setting described in Section IV-D and compare it with ours. Within the considered auto-scaling framework, we demonstrate that our approach provides improved performance.

In our simulations, we consider the following setup:

- *Scenario 1*: Identical parameter setting. Simulations are performed in exactly the same setting that produced Figure 3 (right) with the exception that θ is updated every 10^2 or 10^3 state transitions of the underlying Markov chain.
- *Scenario 2*: Corrected γ weights. As in Scenario 1 but θ is less sensitive to the gradient updates. Specifically, the weights γ_n are multiplied by the number of state transitions (10^2 or 10^3) and divided by $\tau = 10^6$ (used to generate Figure 3 (right)). In this manner, θ changes slower than in Scenario 1 and its variations are of the same order of the ones considered in the evaluation of our algorithm.

For Scenario 1, the corresponding sequences of θ's are plotted in Figures 4 and 5.

Remark 4. *Any point of the x-axis of Figure 4 has a corresponding simulation time of the underlying Markov chains, and the same holds true for the x-axis of Figure 3 (right). It is important to remark that such points coincide on a 1:1 scale. In other words, the curves in these figures describe the evolution of θ on the same time interval.*

Let us comment on the plots in these figures:

- As expected, the plots in Figure 4 are not fully visible because the fluctuations of θ are very sensitive to the contribution of the gradient values. For this reason, we reported the first 100 θ points in Figure 5.
- The curves in Figure 4 (left) indicate that θ tends to follow trajectories that attempt to escape the feasible region from above, which is constrained by $N = 50$ servers, implying $\theta \leq 50$ necessarily. These trajectories exhibit oscillations with maximum amplitude ($N = 50$), which is impractical and clearly suboptimal.
- In contrast, for $\lambda = 0.3$, Figure 4 (right) shows that θ tends to escape the feasible region from *below*. For $\lambda = 0.15$, θ oscillates within the interval 4 ± 0.5, while the optimal θ (between five and six, as shown in Figure 3) lies outside this range. While waiting for 1000 state transitions slightly improves results compared to 100 transitions, the outcomes remain unsatisfactory.

For Scenario 2, the corresponding sequences of θ's are plotted in Figure 6, where again the x-axis maps in simulation time 1:1 with the the x-axis in Figures 3 (right) and 4. As expected, the large oscillations encountered in Scenario 1 are smoothed out by scaling γ_n. We observe that the resulting curves are sensitive to the initial conditions and input parameters and do not converge to a point: They tend to move away from the true optimum θ^*, which is around 5 and 6 as shown in Figure 3 (left). Also, the black curves tend to behave as in Scenario 1 (in average).

In the considered auto-scaling context, the simulation results above indicate that "the fast θ-update approach" does not work.

APPENDIX

In this appendix, we provide proofs for our main results, i.e., Theorems 1 and 2. To do so, it is convenient to rewrite

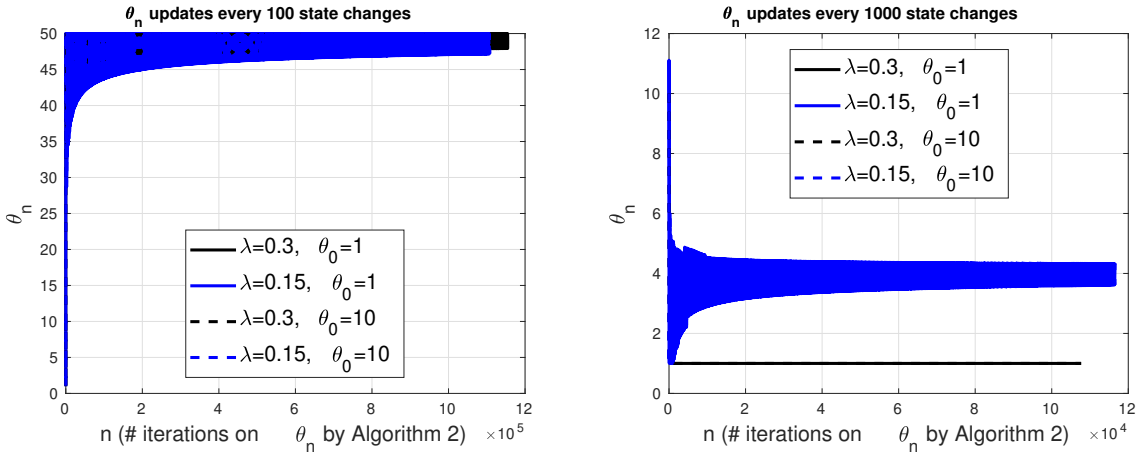

Figure 4. Scenario 1: Plots of the sequences produced by Algorithm 2 when the underlying Markov chain is simulated for τ_n =100 (left) and τ_n =1000 (right) steps. In simulation time, the x-axis corresponds exactly to the one in Figure 3.

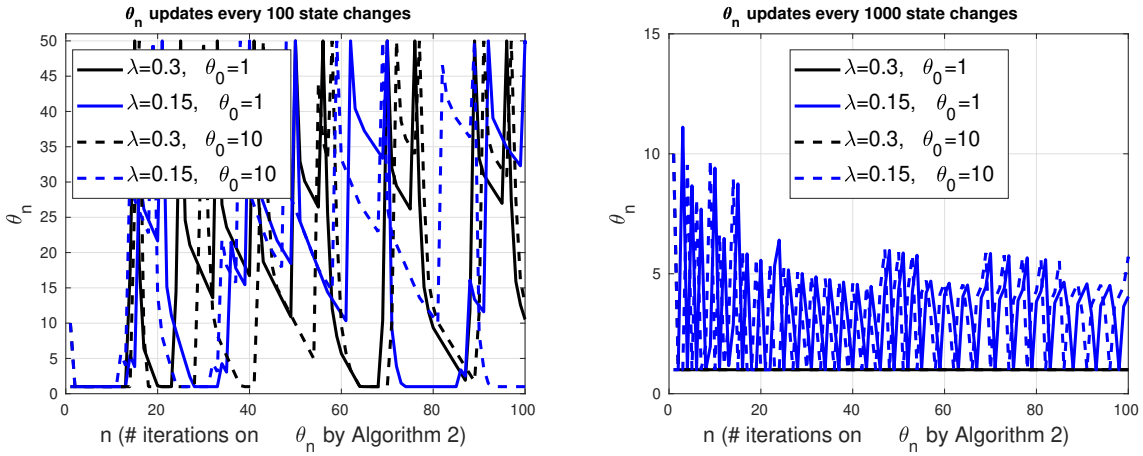

Figure 5. A zoom of the plots of Figure 4 obtained by truncation of the x-axis.

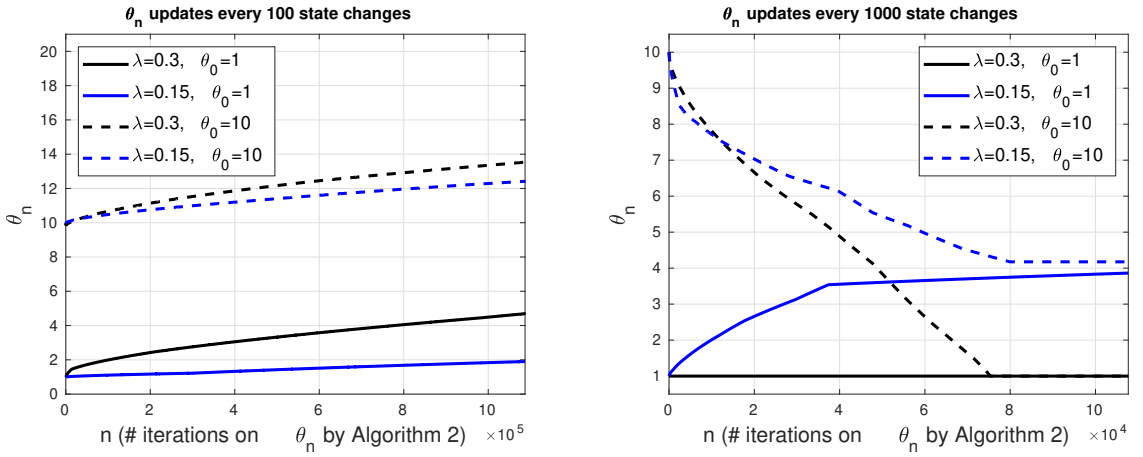

Figure 6. Scenario 2: Plots of the sequences produced by Algorithm 2 when the underlying Markov chain is simulated for τ_n =100 (left) and τ_n =1000 (right) steps. In simulation time, the x-axis corresponds exactly to the one in Figure 3.

the parameter update rule as follows

$$\theta_{n+1} = \theta_n - \gamma_n \left(f'(\theta_n) + \Delta_{\text{diff},n} + \Delta_{\text{mart},n} + \Delta_{\text{mix},n} \right), \tag{15}$$

where we define

$$\begin{aligned}
\Delta_{\text{mart},n} :=& \frac{F(\theta_n + \delta_n, X_\infty^{\theta_n+\delta_n}) - F(\theta_n - \delta_n, X_\infty^{\theta_n-\delta_n})}{2\delta_n} \\
&- \frac{f(\theta_n + \delta_n) - f(\theta_n - \delta_n)}{2\delta_n}, \\
\Delta_{\text{diff},n} :=& \frac{f(\theta_n + \delta_n) - f(\theta_n - \delta_n)}{2\delta_n} - f'(\theta_n), \\
\Delta_{\text{mix},n} :=& \frac{\hat{f}_n(\theta_n + \delta_n) - \hat{f}_n(\theta_n - \delta_n)}{2\delta_n} \\
&- \frac{F(\theta_n + \delta_n, X_\infty^{\theta_n+\delta_n}) - F(\theta_n - \delta_n, X_\infty^{\theta_n-\delta_n})}{2\delta_n}.
\end{aligned}$$

The rewriting (15) will facilitate the control of some difference terms.

We consider specific samples $X_\infty^{\theta_n+\delta_n}$ and $X_\infty^{\theta_n-\delta_n}$ from the stationary distribution of the Markov chains with parameters $\theta_n + \delta_n$ and $\theta_n - \delta_n$ respectively, so that for $i = 0, \dots, K-1$, the couplings $\left(X^{\theta_n+\delta_n}_{T_n+i\tau_n, T_n+(i+1)\tau_n}, X_\infty^{\theta_n+\delta_n}\right)$ and $\left(X^{\theta_n-\delta_n}_{T_n+(K+i)\tau_n, T_n+(K+i+1)\tau_n}, X_\infty^{\theta_n-\delta_n}\right)$ are optimal, as defined in [21, Remark 4.8]. More precisely, for $i = 0$ for example, this means that $X^{\theta_n+\delta_n}_{T_n, T_n+\tau_n}$ and $X_\infty^{\theta_n+\delta_n}$ are different with probability $\left\| P^{\tau_n}_{\theta_n+\delta_n}(x, \cdot) - m_{\theta_n+\delta_n} \right\|_1$, with x denoting the starting state for the Markov chain simulation between time-steps T_n and $T_n + \tau_n$; in the remainder, $\| \cdot \|_1$ denotes the L_1 norm.

1) Proof of Theorem 1: Before delving into the proof, let us first show a preliminary lemma.

Lemma 2. *Asumming 1, the variance conditioned on θ is finite:*

$$\sup_\theta \text{Var}_\theta(F(\theta, X_\infty^\theta)) < \infty.$$

Proof. Using Assumption 1.c, we can write:

$$\begin{aligned}
\text{Var}_\theta(F(\theta, X_\infty^\theta)) &= \mathbb{E}\left[\left(F(\theta, X_\infty^\theta) - \mathbb{E}\left[F(\theta, X_\infty^\theta) \right] \right)^2 \mid \theta \right] \\
&\leq \mathbb{E}\left[\left(G(\theta, X_\infty^\theta) - \mathbb{E}\left[G(\theta, X_\infty^\theta) \right] \right)^2 \mid \theta \right] \\
&\leq 4 G_{\max}^2.
\end{aligned}$$

□

The proof of Theorem 2 is divided in the following steps:

1) We introduce the continuous-time interpolation $\bar{\theta}$ of the discrete process θ.
2) We show that $\bar{\theta}$ is an APT (asymptotic pseudotrajectory, see below) for the flow induced by f', meaning that it remains "close" to a trajectory with flow f'.
3) We then show that $\bar{\theta}$ is a precompact APT, i.e., $f(\bar{\theta}_t)$ remains bounded.
4) We deduce that $\bar{\theta}$ and therefore θ share the same equilibrium set as the trajectories induced by the flow f', which we assumed to be a single point: the minimum of f.

Proof. We first remind the definition of an asymptotic pseudotrajectory (APT) in our setup.

Definition 2 (Asymptotic pseudotrajectory, [7])**.** *Let the continuous map*

$$\begin{aligned}
\Phi : \mathbb{R}_+ \times \mathbb{R} &\to \mathbb{R} \\
(t, \theta) &\mapsto \Phi(t, \theta) = \Phi_t(\theta)
\end{aligned}$$

be a semiflow, so that Φ_0 is the identity and $\Phi_{t+s} = \Phi_t \circ \Phi_s$.

A continuous function X is an asymptotic pseudotrajectory for the semiflow Φ if for any $T > 0$:

$$\lim_{t\to\infty} \sup_{0 \leq h \leq T} |X(t+h) - \Phi_h(X(t))| = 0.$$

Moreover, in $\mathbb{R}$, the asymptotic pseudotrajectory X is said to be precompact if its image is bounded.

Rather than dealing with the discrete trajectory $(\theta_n)_{n\in\mathbb{N}}$, we consider its piecewise linear interpolated counterpart in continuous time, $(\bar{\theta}_t)_{t\in\mathbb{R}_+}$. This is defined as follows

$$\begin{cases} t_n = \sum_{i=1}^n \gamma_i, \\ \bar{\theta}_{t_n+s} = \theta_n + s\frac{\theta_{n+1}-\theta_n}{\gamma_{n+1}} \quad \text{for } 0 \leq s < \gamma_{n+1}. \end{cases} \tag{16}$$

Then, the goal is to use the following proposition from [7].

Proposition 2 (Proposition 4.1 in [7])**.** *Assume that f' is Lipschitz, and that with probability 1, for all $T > 0$:*

$$\lim_{n\to\infty} \sup_{k\in E_n} \left\{ \left| \sum_{i=n+1}^k \gamma_i (\Delta_{\text{diff},i} + \Delta_{\text{mix},i} + \Delta_{\text{mart},i}) \right| \right\} = 0, \tag{17}$$

where $E_n := \{k \mid t_n < t_k \leq t_n + T\}$ is the set of discrete-time timesteps k such that the t_k are within a timeframe T of t_n. Then the interpolated process $(\bar{\theta}_t)_{t\in\mathbb{R}}$ is an asymptotic pseudotrajectory for the flow induced by f'.

Let us first show Equation (17). We deal with each of error term in (17) separately. For any $T > 0$ and any $k \in E_n$:

$$\begin{aligned}
\left| \sum_{i=n+1}^k \gamma_i \Delta_{\text{diff},i} \right| &\leq \sum_{i\in E_n} \gamma_i \left| \Delta_{\text{diff},i} \right| \\
&\leq C_0 \sum_{i\in E_n} \gamma_i \delta_i^2 \\
&\leq C_0 \delta_n^2 \sum_{i\in E_n} \gamma_i \leq C_0 T \delta_n^2 \underset{n\to\infty}{\longrightarrow} 0
\end{aligned}$$

where in the second inequality we have used the bound (27), and in the third and last inequalities, we have used the definition of E_n and that $(\delta_n)_n$ is decreasing from the parametrization properties (7)-(8).

Now, let us consider the second term. Let us recall that $X_\infty^{\theta_n+\delta_n}$ and $X_\infty^{\theta_n-\delta_n}$ build optimal couplings. Using Assumption 2, these quantities are different with probability at most $C_1\rho^{\tau_n} = C_1 n^{-\alpha}$. Choosing $\alpha > 1$, and using a Borel-Cantelli lemma, we have that almost surely, for n large enough: $X^{\theta_n+\delta_n}_{T_n, T_n+\tau_n} = X_\infty^{\theta_n+\delta_n}$. Proceeding similarly for $i = 0, \dots, K-1$ and $\theta_n - \delta_n$, and using a union bound, we obtain (18), i.e., $\Delta_{\text{mix},n} = 0$ almost surely. Therefore, with probability 1,

$$\left| \sum_{i=n+1}^k \gamma_i \Delta_{\text{mix},i} \right| \underset{n\to\infty}{\longrightarrow} 0.$$

$$\begin{aligned}\mathbb{P}\left(\Delta_{\text{mix},n}=0\right) &\geq \mathbb{P}\left(\bigcap_{i=0}^{K-1}\left(X^{\theta_n+\delta_n}_{T_n+i\tau_n,T_n+(i+1)\tau_n}=X^{\theta_n+\delta_n}_{\infty}\cap X^{\theta_n-\delta_n}_{T_n+(K+i)\tau_n,T_n+(K+i+1)\tau_n}=X^{\theta_n-\delta_n}_{\infty}\right)\right)\\ &=1-\mathbb{P}\left(\bigcup_{i=0}^{K-1}\left(X^{\theta_n+\delta_n}_{T_n+i\tau_n,T_n+(i+1)\tau_n}\neq X^{\theta_n+\delta_n}_{\infty}\cup X^{\theta_n-\delta_n}_{T_n+(K+i)\tau_n,T_n+(K+i+1)\tau_n}\neq X^{\theta_n-\delta_n}_{\infty}\right)\right)\\ &\geq 1-\sum_{i=0}^{K-1}\left(\mathbb{P}\left(X^{\theta_n+\delta_n}_{T_n+i\tau_n,T_n+(i+1)\tau_n}\neq X^{\theta_n+\delta_n}_{\infty}\right)+\mathbb{P}\left(X^{\theta_n-\delta_n}_{T_n+(K+i)\tau_n,T_n+(K+i+1)\tau_n}\neq X^{\theta_n-\delta_n}_{\infty}\right)\right)\\ &\geq 1 \text{ for } n \text{ large enough,}\end{aligned} \tag{18}$$

Finally, let us consider the last term. First, we need to show that $\delta_n^2\mathbb{E}\left[\Delta^2_{\text{mart},n}\mid\theta_n\right]<\infty$, so that with the tower rule:

$$\sup_n \delta_n^2\mathbb{E}\left[\Delta^2_{\text{mart},n}\right]<\infty. \tag{19}$$

Let us calculate

$$\begin{aligned}\mathbb{E}\left[\left(F(\theta_n+\delta_n,X_\infty^{\theta_n+\delta_n})-f(\theta_n+\delta_n)\right)^2\mid\theta_n\right]\\ =\text{Var}_{\theta_n+\delta_n}\left(F(X_\infty^{\theta_n+\delta_n})\right),\end{aligned}$$

so that, with the same reasoning for $\theta_n-\delta_n$, using that for any $a,b\in\mathbb{R}$, $(a+b)^2\leq 2a^2+2b^2$, we get

$$\begin{aligned}\sup_n 4\delta_n^2\mathbb{E}\left[\Delta^2_{\text{mart},n}\mid\theta_n\right]&\leq 2\sup_n\delta_n^2\text{Var}_{\theta_n+\delta_n}\left(F(X_\infty^{\theta_n+\delta_n})\right)\\ &\quad+2\sup_n\delta_n^2\text{Var}_{\theta_n-\delta_n}\left(F(X_\infty^{\theta_n-\delta_n})\right)<\infty,\end{aligned}$$

where in the final step we have used Lemma 2.

Using the parametrization property (7) and [33, Theorem 12.1] on the convergence of martingales in $\mathcal{L}^2$, as $M_n:=\sum_{i=1}^n\gamma_i\Delta_{\text{mart},i}$ defines a martingale M with

$$\sum_i\gamma_i^2\mathbb{E}\left[\Delta^2_{\text{mart},i}\right]\leq\sup_n\delta_n^2\mathbb{E}\left[\Delta^2_{\text{mart},n}\mid\theta_n\right]\sum_i\gamma_i^2\delta_i^{-2}<\infty,$$

then almost surely, $M_n\to M_\infty$. Then, for any $n\in\mathbb{N}$ and $k\in E_n$, with probability 1:

$$\begin{aligned}\left|\sum_{i=n+1}^k\gamma_i\Delta_{\text{mart},i}\right|&\leq\left|\sum_{i\geq n}\gamma_i\Delta_{\text{mart},i}\right|+\left|\sum_{i>\sup E_n}\gamma_i\Delta_{\text{mart},i}\right|\\ &\underset{n\to\infty}{\longrightarrow}0.\end{aligned}$$

Overall, summing the $\Delta_{\text{diff},i}$, $\Delta_{\text{mix},i}$ and $\Delta_{\text{mart},i}$, we have proved that (17) holds with probability 1. Now, we show that $\sup_n|\theta_n|<\infty$.

For n large enough, with probability 1, $|\Delta_{\text{diff},n}+\Delta_{\text{mart},n}+\Delta_{\text{mix},n}|<r/2$, with r defined in Assumption 1.d, so that if $|\theta_n|>L$, then equation (15) gives $|\theta_{n+1}|<|\theta_n|-\gamma_n\frac{r}{2}$. Otherwise, if $|\theta_n|<L$, then $|f'(\theta_n)|\leq\sup_{\theta\in[-L,L]}|f'(\theta)|$ (by Assumption 1.b) and with equation (15) $|\theta_{n+1}|\leq|\theta_n|+\gamma_n(\sup_{\theta\in[-L,L]}|f'(\theta)|+r/2)\leq L+\gamma_0(\sup_{\theta\in[-L,L]}|f'(\theta)|+r/2)$, as (γ_n) is decreasing (by the parametrization property (8)).

Therefore, with probability 1, $(\theta_n)_n$ remains bounded and we can apply [7][Proposition 4.1], so that $\bar\theta$ is an asymptotic pseudotrajectory of the flow Φ induced by f'. It is precompact as f is continuous and $\{\theta_n,n\in\mathbb{N}\}$ is bounded with probability 1.

We remark that the flow Φ satisfies $\frac{d\Phi_t(\theta)}{dt}=-f'(\Phi_t(\theta))$. Since

$$\frac{d}{dt}\left[f(\Phi_t(\theta))\right]=\frac{d\Phi_t(\theta)}{dt}\times f'(\Phi_t(\theta))=-f'(\Phi_t(\theta))^2<0,$$

then f is a Lyapounov function of the flow Φ. Using [7, Corollary 6.6] and the uniqueness Assumption 3, we get that $\bar\theta_t\overset{a.s.}{\to}\theta^*$ as it is the only minimum, and thus $\theta_n\overset{a.s.}{\to}\theta^*$ as desired. □

2) Proof of Theorem 2: Our approach consists in decomposing the term of interest in multiple terms and then in bounding each term individually. In the decomposition, the main innovation and difficulty will come from the non-stationary term. More specifically, letting $\xi_n:=\mathbb{E}\left[(\theta_n-\theta^*)^2\right]$, we write

$$\begin{aligned}\xi_{n+1}&=\xi_n+2\mathbb{E}\left[(\theta_{n+1}-\theta_n)(\theta_n-\theta^*)\right]+\mathbb{E}\left[(\theta_{n+1}-\theta_n)^2\right]\\ &\leq\xi_n\\ &\quad-2\gamma_n\mathbb{E}\left[f'(\theta_n)(\theta_n-\theta^*)\right] &(20)\\ &\quad-2\gamma_n\mathbb{E}\left[\Delta_{\text{diff},n}(\theta_n-\theta^*)\right] &(21)\\ &\quad-2\gamma_n\mathbb{E}\left[\Delta_{\text{mart},n}(\theta_n-\theta^*)\right] &(22)\\ &\quad-2\gamma_n\mathbb{E}\left[\Delta_{\text{mix},n}(\theta_n-\theta^*)\right] &(23)\\ &\quad+\mathbb{E}\left[(\theta_{n+1}-\theta_n)^2\right]. &(24)\end{aligned}$$

In the remainder, we bound the five terms above following these lines:

- To bound (20), we use Assumption 4 as in classical gradient descent algorithms. With the strong convexity from Assumption 4, we obtain

$$\begin{aligned}-2\gamma_n\mathbb{E}\left[f'(\theta_n)(\theta_n-\theta^*)\right]&\leq-2\gamma_n\kappa\mathbb{E}\left[(\theta_n-\theta^*)^2\right]\\ &\leq-2\gamma_n\kappa\xi_n.\end{aligned} \tag{25}$$

- To bound (21), using Assumption 1 and the Cauchy-Schwarz inequality, we obtain

$$\begin{aligned}&-2a_n\mathbb{E}\left[\Delta_{\text{diff},n}(\theta_n-\theta^*)\right]\\ &\leq 2\gamma_n\mathbb{E}\left[\Delta^2_{\text{diff},n}\right]^{1/2}\mathbb{E}\left[(\theta_n-\theta^*)^2\right]^{1/2}\\ &\leq 2C_0\gamma_n\delta_n^2\xi_n^{1/2},\end{aligned}$$

where we used that $\Delta_{\text{diff},n}\leq C_0\delta_n^2$ by assumption on f and by its Euler discretization around any parameter.

More precisely, we can write the Taylor expansion, for $\varepsilon = 1$ or $\varepsilon = -1$, as

$$f(\theta_n + \varepsilon\delta_n) = f(\theta_n) + f'(\theta_n)\varepsilon\delta_n + f''(\theta_\varepsilon)\frac{\delta_n^2}{2}, \quad (26)$$

for some $\theta_\varepsilon \in [\theta_n - \delta_n, \theta_n + \delta_n]$, so that:

$$\Delta_{\mathrm{diff},n} \leq \delta_n \frac{|f''(\theta_{+1}) - f''(\theta_{-1})|}{2} \leq \delta_n^2 \sup_{\theta\in\Theta} f^{(3)}(\theta). \quad (27)$$

- To bound (22), we notice that $\mathbb{E}\left[\Delta_{\mathrm{mart},n} \mid \theta_n\right] = 0$. Taking the expectation, we get $-2\gamma_n \times \mathbb{E}\left[\Delta_{\mathrm{mart},n}\left(\theta_n - \theta^*\right)\right] = 0$.
- To bound (23), using the ergodicity structure in Assumption 2, the Assumption 1 and first with a Cauchy-Schwarz inequality:

$$\begin{aligned}
&-2\gamma_n \mathbb{E}\left[\Delta_{\mathrm{mix},n}(\theta_n - \theta^*)\right] \\
&\leq 2\gamma_n \mathbb{E}\left[\Delta_{\mathrm{mix},n}^2\right]^{1/2} \mathbb{E}\left[\left(\theta_n - \theta^*\right)^2\right]^{1/2} \\
&= 2\gamma_n \mathbb{E}\left[\left(\theta_n - \theta^*\right)^2\right]^{1/2} \mathbb{E}\left[\mathbb{E}\left[\Delta_{\mathrm{mix},n}^2 \mid \theta_n\right]\right]^{1/2}.
\end{aligned}$$

With i) Assumption 1.c on the instant cost function, ii) using that $(\sum_{i=0}^{K-1} a_i)^2 \leq K \sum_{i=0}^{K-1} a_i^2$ for any $a_i \in \mathbb{R}$, which follows by a linear algebra argument, and iii) letting in our case $a_i := G(\theta_n + \delta_n, X_{T_n+i\tau_n, T_n+(i+1)\tau_n}) - G(\theta_n + \delta_n, X_\infty^{\theta_n+\delta_n})$, we obtain

$$\begin{aligned}
&\mathbb{E}\left[\left(\hat{f}_n(\theta_n + \delta_n) - F(\theta_n + \delta_n, X_\infty^{\theta_n+\delta_n})\right)^2 \mid \theta_n\right] \\
&= \frac{1}{K^2}\mathbb{E}\left[\left(\sum_{i=0}^{K-1} a_i\right)^2 \mid \theta_n\right] \leq \frac{1}{K}\sum_{i=0}^{K-1} \mathbb{E}\left[a_i^2 \mid \theta_n\right] \\
&\leq \frac{G_{\max}^2}{K} \sum_{i=0}^{K-1} \left\| P_{\theta_n+\delta_n}^{\tau_n}(x_{\theta_n+\delta_n}^{(i,+)}, \cdot) - m_{\theta_n+\delta_n} \right\|_1,
\end{aligned}$$

where $x_{\theta_n+\delta_n}^{(i,+)}$ denotes the initial state for the $i-$th simulation of the Markov chain with parameter $\theta_n + \delta_n$ at episode n. Applying the same argument, a similar bound is obtained for the Markov chain with parameter $\theta_n - \delta_n$. Now, using that $(a+b)^2 \leq 2a^2 + 2b^2$ for any $a, b \in \mathbb{R}$, for the conditional expectation we obtain

$$\begin{aligned}
&4\delta_n^2 \mathbb{E}\left[\Delta_{\mathrm{mix},n}^2 \mid \theta_n\right] \\
&\leq 2\mathbb{E}\Big[\left(\hat{f}_n(\theta_n + \delta_n) - F(\theta_n + \delta_n, X_\infty^{\theta_n+\delta_n})\right)^2 \\
&\quad + \left(\hat{f}_n(\theta_n - \delta_n) - F(\theta_n - \delta_n, X_\infty^{\theta_n-\delta_n})\right)^2 \mid \theta_n\Big] \\
&\leq \frac{2G_{\max}^2}{K} \sum_{i=0}^{K-1} \left\| P_{\theta_n+\delta_n}^{\tau_n}(x_{\theta_n+\delta_n}^{(i,+)}, \cdot) - m_{\theta_n+\delta_n} \right\|_1 \\
&\quad + \left\| P_{\theta_n-\delta_n}^{\tau_n}(x_{\theta_n-\delta_n}^{(i,-)}, \cdot) - m_{\theta_n-\delta_n} \right\|_1 \\
&\leq 2G_{\max}^2 C_1 \rho^{\tau_n},
\end{aligned}$$

where in the last inequality we have used ergodicity (Assumption 2). We continue the computations of the bound on (23):

$$-2\gamma_n \mathbb{E}\left[\Delta_{\mathrm{mix},n}(\theta_n - \theta^*)\right] \leq \sqrt{2}\frac{\gamma_n}{\delta_n}\xi_n^{1/2} G_{\max} C_1^{1/2} \rho^{\tau_n/2}.$$

Choosing $T_n := 2\alpha \frac{\log n}{\log 1/\rho}$, we finally obtain that

$$-2\gamma_n \mathbb{E}\left[\Delta_{\mathrm{mix},n}(\theta_n - \theta^*)\right] \leq \sqrt{2}\frac{\gamma_n}{\delta_n}\xi_n^{1/2} G_{\max} C_1^{1/2} n^{-\alpha}.$$

- To bound (24), we let

$$\begin{aligned}
a_i := &\, G(\theta_n + \delta_n, X_{T_n+i\tau_n, T_n+(i+1)\tau_n}) \\
&- G(\theta_n - \delta_n, X_{T_n+(K+i)\tau_n, T_n+(K+i+1)\tau_n})
\end{aligned}$$

to obtain

$$\begin{aligned}
&\mathbb{E}\left[\left(\theta_{n+1} - \theta_n\right)^2\right] \\
&= \frac{\gamma_n^2}{4\delta_n^2}\mathbb{E}\left[\left(\hat{f}_n(\theta_n + \delta_n) - \hat{f}_n(\theta_n - \delta_n)\right)^2\right] \\
&= \frac{\gamma_n^2}{4\delta_n^2}\mathbb{E}\left[\left(\frac{1}{K}\sum_{i=0}^{K-1} a_i + b(\theta_n + \delta_n) - b(\theta_n - \delta_n)\right)^2\right] \\
&\leq \frac{\gamma_n^2}{2\delta_n^2}\mathbb{E}\left[\left(\frac{1}{K}\sum_{i=0}^{K-1} a_i\right)^2 + \left(b(\theta_n + \delta_n) - b(\theta_n - \delta_n)\right)^2\right] \\
&\leq \frac{\gamma_n^2}{2K\delta_n^2}\mathbb{E}\left[\sum_{i=0}^{K-1} a_i^2\right] + \frac{C_2\gamma_n^2}{2}\mathbb{E}\left[(\theta_n - \theta^*)^2\right] \\
&\leq \frac{2\gamma_n^2 G_{\max}^2}{\delta_n^2} + \frac{C_2\gamma_n^2}{2}\xi_n
\end{aligned}$$

where the second inequality follows by Assumption 1.c.

Finally, summing the previous terms, we obtain

$$\xi_{n+1} \leq (1 - q_n)\xi_n + u_n \xi_n^{1/2} + v_n,$$

where $q_n := \gamma_n\left(2\kappa - \frac{C_2\gamma_n}{2}\right)$, $u_n = 2C_0\gamma_n\delta_n^2 + \sqrt{2}G_{\max}C_1^{1/2}\frac{\gamma_n}{\delta_n}n^{-\alpha}$ and $v_n = 2\frac{G_{\max}^2\gamma_n^2}{\delta_n^2}$.

We can already choose α such that $\frac{n^{-\alpha}}{\delta_n} \leq \delta_n^2$. In this case, $u_n \leq C_u\delta_n^2\gamma_n$ with $C_u := 2C_0 + \sqrt{2}G_{\max}C_1^{1/2}$. Now, let us consider the following lemma from [31], which we state here for convenience. This lemma is purely algebraic and will let us identify the "correct" scaling for the input parameters of Algorithm 2.

Lemma 3 ([31])**.** *Let ξ_n be a positive sequence, A_n be a sequence, and B_n, C_n be positive non-increasing sequences such that*

$$\xi_{n+1} \leq \xi_n(1 - A_n) + A_n B_n \xi_n^{1/2} + C_n A_n.$$

Then,

$$\limsup_{n\to\infty} \frac{\xi_n}{B_n^2 + C_n} \leq \frac{1}{2}.$$

Since $\gamma_n < \frac{4\kappa}{C_2}$, then q_n is positive and Lemma 3 gives the scaling:

$$\limsup_{n\to\infty} \frac{\xi_n}{B_n^2 + C_n} \leq \frac{1}{2},$$

with $B_n = \frac{2C_u\delta_n^2}{4\kappa - C_2\gamma_n}$ and $C_n = \frac{4G_{\max}^2\gamma_n}{\delta_n^2(4\kappa - C_2\gamma_n)}$. Therefore, to get comparable scalings, we choose the parameters such that δ_n^4 and $\gamma_n\delta_n^{-2}$ are of the same order. A valid parameter choice is

obtained when $\gamma_n = n^{-1}$, $\delta_n = n^{-1/6}$ and $\alpha > 1/2$. Within these parameters, we get

$$\limsup_{n\to\infty} \xi_n n^{2/3} \leq \frac{\left(2C_0 + \sqrt{2}G_{\max}C_1^{1/2}\right)^2}{8\kappa^2} + \frac{G_{\max}^2}{2\kappa}. \quad (28)$$

This concludes the proof.

REFERENCES

[1] Siddharth Agarwal, Maria A. Rodriguez, and Rajkumar Buyya. A reinforcement learning approach to reduce serverless function cold start frequency. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 797–803, 2021.

[2] Siddharth Agarwal, Maria A. Rodriguez, and Rajkumar Buyya. A Deep Recurrent-Reinforcement Learning Method for Intelligent AutoScaling of Serverless Functions . *IEEE Transactions on Services Computing*, 17(05):1899–1910, September 2024.

[3] Sebastian Allmeier and Nicolas Gast. Computing the bias of constant-step stochastic approximation with markovian noise. 2024.

[4] Jonatha Anselmi. Asynchronous load balancing and auto-scaling: Mean-field limit and optimal design. *IEEE/ACM Transactions on Networking*, 32(4):2960–2971, 2024.

[5] Mohammad Sadegh Aslanpour, Adel N. Toosi, Muhammad Aamir Cheema, Mohan Baruwal Chhetri, and Mohsen Amini Salehi. Load balancing for heterogeneous serverless edge computing: A performance-driven and empirical approach. *Future Generation Computer Systems*, 154:266–280, 2024.

[6] Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

[7] Michel Benaïm. Dynamics of stochastic approximation algorithms. 1999.

[8] Priscilla Benedetti, M. Femminella, G. Reali, and Kris Steenhaut. Reinforcement learning applicability for resource-based auto-scaling in serverless edge applications. In *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 674–679, 2022.

[9] V.S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.

[10] Siddharth Chandak, Vivek S. Borkar, and Parth Dodhia. Concentration of contractive stochastic approximation and reinforcement learning. *Stochastic Systems*, 12(4):411–430, 2022.

[11] Behzad Chitsaz, Ahmad Khonsari, Masoumeh Moradian, Aresh Dadlani, and Mohammad Sadegh Talebi. Scaling power management in cloud data centers: A multi-level continuous-time mdp approach. *IEEE Transactions on Services Computing*, pages 1–12, 2024.

[12] Randall E. Cline. Inverses of rank invariant powers of a matrix. *SIAM Journal on Numerical Analysis*, 5(1):182–197, 1968.

[13] Jeffrey Dean and Luiz André Barroso. The tail at scale. *Commun. ACM*, 56(2):74–80, February 2013.

[14] Javad Dogani and Farshad Khunjush. Proactive auto-scaling technique for web applications in container-based edge computing using federated learning model. *Journal of Parallel and Distributed Computing*, 187:104837, 2024.

[15] John C. Duchi, Alekh Agarwal, Mikael Johansson, and Michael I. Jordan. Ergodic mirror descent. *SIAM Journal on Optimization*, 22(4):1549–1578, 2012.

[16] Anshul Gandhi, Sherwin Doroudi, Mor Harchol-Balter, and Alan Scheller-Wolf. Exact analysis of the m/m/k/setup class of markov chains via recursive renewal reward. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '13, page 153–166, New York, NY, USA, 2013. Association for Computing Machinery.

[17] Diego Goldsztajn, Andres Ferragut, Fernando Paganini, and Matthieu Jonckheere. Controlling the number of active instances in a cloud environment. *SIGMETRICS Perform. Eval. Rev.*, 45(3):15–20, March 2018.

[18] G. H. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on Numerical Analysis*, 10(2):413–432, 1973.

[19] Bjorn Johansson, Maben Rabi, and Mikael Johansson. A simple peer-to-peer algorithm for distributed optimization in sensor networks. In *2007 46th IEEE Conference on Decision and Control*, pages 4705–4710, 2007.

[20] Björn Johansson, Maben Rabi, and Mikael Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 20(3):1157–1170, 2010.

[21] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2008.

[22] Nima Mahmoudi and Hamzeh Khazaei. Performance modeling of serverless computing platforms. *IEEE Transactions on Cloud Computing*, 10(4):2834–2847, 2022.

[23] G. Ch. Pflug. On-line optimization of simulated markovian processes. *Mathematics of Operations Research*, 15(3):381–395, 1990.

[24] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1 edition, April 1994.

[25] S. Sundhar Ram, A. Nedić, and V. V. Veeravalli. Incremental stochastic subgradient algorithms for convex optimization. *SIAM Journal on Optimization*, 20(2):691–717, 2009.

[26] Miklós Rásonyi and Kinga Tikosi. Convergence of the kiefer–wolfowitz algorithm in the presence of discontinuities. *Advances in Applied Probability*, 55(2):382–406, 2023.

[27] G. W. Stewart. On the continuity of the generalized inverse. *SIAM Journal on Applied Mathematics*, 17(1):33–45, 1969.

[28] Tao Sun, Yuejiao Sun, and Wotao Yin. On markov chain gradient descent. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9918–9927, Red Hook, NY, USA, 2018. Curran Associates Inc.

[29] Thomas Tournaire, Hind Castel-Taleb, and Emmanuel Hyon. Efficient computation of optimal thresholds in cloud auto-scaling systems. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 8(4), jul 2023.

[30] Erwin van Eyk, Alexandru Iosup, Cristina L. Abad, Johannes Grohmann, and Simon Eismann. A spec rg cloud group's vision on the performance challenges of faas cloud architectures. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, ICPE '18, page 21–24, New York, NY, USA, 2018. ACM.

[31] Walton N. Zero-order stochastic optimization: Kiefer-wolfowitz. https://appliedprobability.blog/2022/10/28/zero-order-stochastic-optimization-keifer-wolfowitz/, 2022.

[32] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. Peeking behind the curtains of serverless platforms. In *Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '18, page 133–145, USA, 2018. USENIX Association.

[33] David Williams. *Probability with Martingales*. Cambridge University Press, 1991.

Jonatha Anselmi is a tenured researcher at the French National Institute for Research in Digital Science and Technology (Inria), since 2014. Prior to this, he was a full-time researcher at the Basque Center for Applied Mathematics and a postdoctoral researcher at Inria. He received his PhD in computer engineering at Politecnico di Milano (Italy) in 2009. At the intersection of applied mathematics, computer science and engineering, his research interests focus on decision making under uncertainty, with particular emphasis on the development of highly-scalable algorithms that minimize congestion and operational costs of large-scale distributed systems.

Bruno Gaujal is an Inria researcher. Till Dec. 2015, he has been the head of the large-scale computing team in Inria Grenoble-Alpes. He has held several positions in AT&T Bell Labs, Loria and École Normale Supérieure of Lyon. He obtained his PhD from University of Nice in 1994. He is a founding partner of a start-up company, RTaW, since 2007. His main interests are in performance evaluation, optimization and control of large discrete event dynamic systems with applications to telecommunication and large computing infrastructures.

Louis-Sebastien Rebuffi defended his PhD thesis in 2023 at Université Grenoble Alpes under the supervision of Bruno Gaujal and Jonatha Anselmi. His research interests focus on reinforcement learning algorithms applied to controlled queuing systems, viewing them as Markov decision processes.