Cache allocation in multi-tenant edge computing: an online model-based reinforcement learning approach

Ayoub Ben Ameur, Andrea Araldo, Tijani Chahed, György Dán

Cache Allocation in Multi-tenant Edge Computing: An Online Model-based Reinforcement Learning Approach

Ayoub Ben-Ameur[1,2], Andrea Araldo[1], Tijani Chahed[1], and György Dán[2]

[1]SAMOVAR, Telecom SudParis, Institut Polytechnique de Paris
[2]Division of Network and Systems Engineering, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology

***Abstract*—We consider a Network Operator (NO) that owns Edge Computing (EC) resources, virtualizes them and lets third party Service Providers (SPs) run their services, using the allocated slice of resources. We focus on one specific resource, i.e., cache space, and on the problem of how to allocate it among several SPs in order to minimize the backhaul traffic. Due to confidentiality guarantees, the NO cannot observe the nature of the traffic of SPs, which is encrypted. Allocation decisions are thus challenging, since they must be taken solely based on observed monitoring information. Another challenge is that not all the traffic is cacheable. We propose a data-driven cache allocation strategy, based on Reinforcement Learning (RL). Unlike most RL applications, in which the decision policy is learned offline on a simulator, we assume no previous knowledge is available to build such a simulator. We thus apply RL in an *online* fashion, i.e., the model and the policy are learned by directly perturbing and monitoring the actual system. Since perturbations generate spurious traffic, we thus need to limit perturbations. This requires learning to be extremely efficient. To this aim, we devise a strategy that learns an approximation of the cost function, while interacting with the system. We then use such an approximation in a Model-Based RL (MB-RL) to speed up convergence. We prove analytically that our strategy brings cache allocation boundedly close to the optimum and stably remains in such an allocation. We show in simulations that such convergence is obtained within few minutes. We also study its fairness, its sensitivity to several scenario characteristics and compare it with a method from the state-of-the-art.**

***Index Terms*—Edge Computing, multi-tenant, cache allocation, online learning, model-based reinforcement learning.**

I. Introduction

MObile Edge Computing (MEC) consists in deploying computational capabilities, e.g., RAM, CPU, storage, GPUs, into nodes at the network's edge [1]. Such nodes could be co-located with (micro) base stations, access points, road side units, etc. By serving users directly at the network, Network Operators (NOs) can reduce latency and backhaul traffic, i.e., traffic to/from remote server locations. Backhaul traffic due to over-increasing user demand imposes high costs on the NOs, due to the required network infrastructure and the contracts with other Internet Providers to receive such data from the Internet.[1] MEC is thus strategic for NOs to reduce such costs. MEC is particular relevant for content delivery, which might represent 80% of the Internet traffic, and in particular video [6]. In this context, an extremely important resource is cache space, so as to store such content close to users. Big Service Provides (SPs), such as Netflix or Google, are already deploying MEC solutions to cache their content [7], [8]: they install their own hardware servers into the access networks of some NOs, and directly serve the most popular content to users from there. Such solutions have important limits. (i) Such solutions are costly and are only affordable to big SPs. (ii) It is impossible to install hardware servers in the edge nodes very close to end users, e.g., in wifi access points or base stations. Since such nodes are owned or controlled by the NO, the NO is the only one that can deploy computation resources therein. On the other hand, the services consumed by users, e.g., video streaming, are typically provided by third party SPs. And so the question is how to let SPs run their applications close to the users on resources owned by the NO. We position our work in the framework of multi-tenant MEC [9]: the NO virtualizes its computational resources at the edge (cache space in our case), allocates them to the SPs (tenants), and lets them use such resources to serve their respective users. The NO might ask a payment for the use of such resources [10]. However, we assume here that SPs can use MEC cache for free to serve their users directly at the edge, since the consequent reduction in backhaul related cost for the NO can compensates the cost of cache deployment [11].

Different from classic caching settings, we consider that traffic is encrypted by SPs, and so it is impossible for the NO to know which objects are requested, which ones are popular, nor whether they are cacheable (for instance video calls). The NO only allocates storage among SPs and lets each SP decide

[1]In some cases, the NO could be connected to the Internet via a Transit Internet Provider, which requires to be paid [2]. In other cases, the NO could have peering agreements with an Internet Provider, where traffic can be exchanged for free, but cannot anyways exceed certain limits specified in the agreement [3]. Another alternative for the NO is to join Internet Exchange Points (IXPs) [4]. In this case, the price the NO pays increases with the requested "port capacity", which increases with the backhaul traffic [5, Section VI.4].

what to cache within the allocated space (Fig. 1).[2] We study the problem of the NO to optimally allocate cache storage among several SPs, so as to minimize the backhaul traffic, i.e., the traffic from the Internet to the edge node, without knowing the requested video content.

Since all traffic is encrypted, the NO can make allocation decisions based on data-driven strategies, consisting in *trial and error*: the NO continuously perturbs cache allocation and observes induced variation on the backhaul traffic. We formulate this problem as a Markov Decision Process (MDP) and we use *online* model-based Reinforcement Learning (RL) to solve it: while RL is usually trained offline and then applied to a real system, **we instead train RL on the system while it is up and running**. Therefore, we are not only interested in finding a good allocation policy, but also in *how* to find it. Indeed, the only way for the NO to learn how to optimize the allocation is to continuously perturb it, but we need to keep the cost of such perturbations small. Our contributions can be summarized as follows:

1) We propose a model-based RL agent for cache allocation among third-party SPs. Starting from no knowledge of the system, the agent constructs a model of the backhaul traffic based on observations. Our RL agent includes the following features: memory replay, learning rate scheduling and decaying exploration probability (epsilon-greedy).
2) We analytically prove that the system converges to a state close to the optimum and stays there with probability 1 for an infinite horizon.
3) We show in simulation the performance of our method under different scenario parameters and show that it outperforms the state of the art Simultaneous Perturbation Stochastic Approximation (SPSA) [12]. We also show the benefits of using model-based RL over model-free RL.

The remainder of this paper is organized as follows. In Section II, we review recent related work. In Section III, we present our system model. In Section IV, we present the MDP formulation of our problem. Section V and Section VI discuss the theoretical properties and motivate the use of RL, respectively. In Section VII, we show our simulation results. Our simulation code is available as open source.[3] Section VIII contains our conclusion and some hints on future work. All the proofs are in the Appendix provided as supplemental material.

II. Related Work

We now review recent work from the literature pertaining to resource allocation and cache management (Section II-A), with a focus on data-driven approaches to solve such problems (Section II-B). We also include the case of multiple resources to evaluate whether restricting them to the case of one resource could relate to our problem and proposed solution (Section II-C).

[2]As in classic content caching, we assume the SPs do not pay the NO for the cache: cache is used by SPs for free, and the NO compensates the initial storage deployment cost with backhaul traffic reduction.

[3]https://github.com/Ressource-Allocation/Cache-Allocation-Project

A. Cache allocation

Authors of [13] propose a cache partitioning approach, where each cache can be partitioned into slices with each slice dedicated to a content provider. However, they need information about the system conditions in order to solve it (e.g., request rate for each content provider). We assume instead that no information is available and that optimization is done by observing the changes in backhaul traffic induced by perturbing the allocation.

In [14], authors formulate collaborative joint caching and processing problem as an Integer Linear Program that minimizes the back-haul network cost, subject to cache storage and processing capacity constraints for only one SP. However, in our work EC is a multi-tenant environment. Compared with single-tenancy, multi-tenancy enables higher utilization of the cache resources in cloud servers as well as in edge servers.

Authors of [15] study optimal content caching problem in EC, assuming that the NO decides what content to store in the cache available at the edge, while in our case, because of encryption, the NO cannot know the content of SPs; it can thus only allocate cache slots to SPs and let them decide what content to put in. Our assumption is more realistic, as today SPs generally require that their traffic stays encrypted and unknown to the NO.

In [16], the authors propose a resource pricing framework for one NO and several SPs, for several well-established resource allocations knowing the demand of the users. We instead do not know anything about the requests nature. Also, our focus is on resource allocation and not pricing, we assume that SPs do not pay for the resources, as we will discuss in the beginning of Section III.

In [17], the EC network is assumed to have multiple cache servers to assist SPs, each with its own set of users and acts as a rational selfish player, aiming to maximize its utility by making strategies of local self caching. In [11], authors also consider sharing cache between SPs, by applying coalitional game theory. They assume that the cache resources at the edge are owned by a central office equipped as a data center connecting multiple NOs and each NO pays for these resources to make them available for SPs. The NO also pays SPs to convince them to use the cache at the edge and lets them decide how much resources they get. This may seem counter-intuitive, as we expect that the NO will get paid when it provides certain resource. They justify this by the fact that the interest of the NO is to minimize its expenses: the authors show that if the NO does not pay SPs to use cache, he will end up paying more, in order to carry the big traffic from SPs. However, our allocation decision is centralized by the NO, who owns the resources, and we do not require any payment as shall be explained in the beginning of Section III.

Recently, authors of [18] considered the case where the NO is able to place application images at the edge, in a cache with limited capacity. The problem is solved via a Stackelberg game. The authors work under complete information assumption, i.e., the system parameters and utilities are assumed to be known by the NO, while in our work this information is unknown.

B. Data-driven methods for resource allocation

The main limit of most of the aforementioned works is that they assume that an exact characterization of the system is known, i.e., all the quantities and dependencies involved in the resource allocation problem are known in advance. This is not the case for our problem, where we do not know the expression of the cost function we want to minimize, e.g. , inter-domain bandwidth consumption. We thus need to resort to data-driven approaches, which can drive cost functions toward the minimum in the absence of a known characterization of the system, based solely on monitoring information.

RL has been used for resource allocation in the context of EC in, for instance, [19], [20], [21], [22], [23]. Joint management of the communication and computation resources using deep RL is considered in [19]. In [20], the authors consider a cluster with multiple resource types and use deep RL to choose one or more of the waiting jobs to schedule at each time step. In [21], the authors solve the problem of allocating GPU at the edge to run deep neural networks to maximize the quality of service of the prediction model. Authors of [22] use a RL approach to allocate CPU time, Virtual CPUs and memory, to Virtual Machines (VMs). In [23], authors propose deep RL for allocating resources in a network slicing scenario. Contrary to our approach, the authors of the works mentioned above pre-train the RL algorithm offline on a simulated system before using it on the actual one. Instead, we assume that the behavior of the system is not known in advance, which makes it impossible to build a simulator to model it. Our challenge is thus to train our algorithm online, acting directly on the hot and running system. This imposes on us a more parsimonious learning strategy as we need to ensure that the system is not heavily perturbed during training.

In [24], the authors present a RL algorithm for resource auto-scaling in clouds: resources are assumed to be unlimited, however the goal is to allocate to each SP an amount of resources that does not exceed its needs. In our case, resources are scarce, allocating resources to one SP means allocating less for another.

To the best of our knowledge, the only method we can compare against is SPSA [12], since the latter is the only work to propose a data-driven approach to partition a finite amount of cache among several SPs with encrypted content. The authors do so based on stochastic optimization. However, they need to continuously perturb the allocation, generating spurious backhaul traffic that may be non-negligible. We instead include traffic perturbation into the cost function, thus managing to keep it low, which allows us to outperform [12], as shown in Section VII.

A preliminary version of this work was presented in [25]. With respect to it, we added model-based RL, which outperforms the simple model-free RL of [25], and we thoroughly study its convergence analytically, whereas in [25] no theoretical results were available.

C. Multi-resource allocation at the Edge

In [26], the authors consider an EC system under network slicing in which the wireless devices generate latency sensitive computational tasks. The allocation of wireless and computing resources to a set of autonomous wireless devices in an EC system is considered in [27]. They model the interaction between the NO, which manages the allocation of wireless and computing resources, and devices. In order to minimize completion time, devices autonomously decide whether to use shared resources for offloading computing tasks so as to minimize their own completion times or to compute tasks locally. In [28], authors establish a software-defined networking based architecture for edge/cloud computing services in 5G heterogeneous networks, which can support efficient and on-demand computing resource management to optimize resource utilization and satisfy time-varying computational tasks. A dynamic provisioning of computing resources is considered in [29]. Computing resources are provisioned as VM instances on the fly. The authors propose two allocation and pricing mechanisms based on greedy algorithm and linear programming based approximation. A main common assumption of the papers above is that user devices submit tasks to the NO and such tasks consume resources to be executed and transmitted. Contention for resources is then modeled among user devices. However, we consider that these models are not appropriate for EC in our vision, since all traffic between devices and service providers is encrypted to maintain confidentiality and the NO does not have control over it. Therefore the contention for resources is, in our vision, between SPs and not between tasks submitted by users. In our assumption, the NO can only decide how to allocate resources among SPs and then users device interact directly with SPs, outside the control of the NO.

In [30], the authors assume that each SP explicitly requests a certain amount of resources (e.g., memory, CPU and link capacity) and consider the difference between the resources requested and the resources actually allocated to them. The decision maker (the NO) in this work aims to maximize the fairness of the resources allocation. In both [31] and [32], the authors optimize the offloading decision of end users and resource allocation strategy to minimize the system latency subject to dynamic cache capacities and computing resource constraints. Authors of [32] also aim to minimize energy consumption. We instead assume that the NO allocates its edge resources so as to satisfy its own goals (backhaul traffic minimization in our case), without requiring to receive explicit resource requests from SPs.

An explicit request of SPs of the amount of resources they are willing to consume is also required in [9] and [33]. The former relies on a heuristic for resolution and assumes that SPs are truthful and declare the resources they really need. The latter makes use of Monte-Carlo Tree Search, SPs pay proportionally to the resources they are granted. We do not need such assumptions in our work.

Similar to us, but in the context of network slicing, in [34] the NO jointly allocates CPU and bandwidth to several tenants, one per slice, in order to minimize an objective function relevant to the NO (energy, in their case). However, they assume the NO knows the expected load of requests of each tenant and the requirements of each request, while we do not require this assumption in this paper.

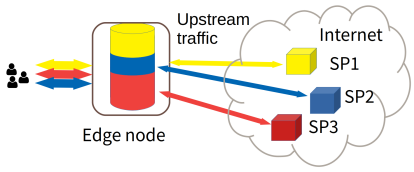


Fig. 1: Cache allocation and backhaul traffic (Origin servers → Edge) with multiple Service Providers (SPs).

In [35], the authors propose a new market-based framework for efficiently allocating resources of heterogeneous capacity-limited edge nodes to multiple competing SPs. Each SP is a player. Given a price vector of the resources, each SP aims to maximize its utility subject to a certain budget constraint. The authors assume that this utility function is known (see Section 4.2). However, in our case the utility (cost) function is unknown. Moreover, we consider the utility of the NO (not SPs), who owns the resources.

In [36], multiple edge servers and only one SP are considered. The resources on these edge servers are managed by multiple NOs. The SP has a budget to use resources at the edge. The end-users subscribed to this SP submit delay-sensitive jobs with the aim to be executed under delay constraints. An algorithm is proposed to allocate resources at each time-slot to the submitted jobs. While this work considers only one SP, we consider multiple SPs and one NO.

TABLE I: Table of notation

Notation	Description
P	Number of SPs (Section III)
K	Cache total capacity (Section III)
$\mathcal{N}_p$	Catalog of SP p (Section III-A)
N_p	Catalog size of SP p (Section III-A)
ζ_p	Cacheability of SP p (Section III-A)
ω	Exogenous conditions (Section III-B)
$\boldsymbol{\theta}$	Cache allocation (Section III)
$\mathbf{a}$	Action of the NO (Section III)
θ_p	Number of slots given to SP p (Section III)
$\boldsymbol{\theta}^*$	Optimal allocation (Section III-C)
Δ	Perturbation (Section IV-A)
$\boldsymbol{\theta}^{\text{prop}}$	Proportional allocation (Section VII-B)
λ	Total request rate (Section III-A)
f_p	Probability that a request is for SP p (Section III-A)
$\rho_{c,p}$	Popularity of object c of SP p (Section III-A)
$\lambda_{c,p}$	Request rate for object c of SP p (Section III-A)
$C_{\text{nom},p}(\theta_p,\omega)$	Nominal cost for SP p (Section III-B)
$C_{\text{nom}}(\boldsymbol{\theta},\omega)$	Nominal cost (Section III-B)
$C_{\text{pert}}(\mathbf{a})$	Perturbation cost (Section III-C)
$C^{(k)}$	Instantaneous cost (Section III-C)
$\mathcal{S}$	State space (Section IV-A)
$\mathcal{A}_{\boldsymbol{\theta}}$	Action space (Section IV-A)
$\alpha^{(k)}$	Learning rate at time slot k (12)
$\gamma^{(k)}$	Discount factor at time slot k (12)
$\epsilon^{(k)}$	Epsilon at time slot k (Section IV-C3)
$\hat{C}_{\text{nom},p}(\theta_p)$	Model of nominal cost for SP p (Section IV-B)
$\hat{C}_{\text{nom}}(\boldsymbol{\theta})$	Model of total nominal cost (Section IV-B)
$\mathcal{M}^{(k)}$	Memory at time slot k (Section IV-C2)
N_{memory}	Mini-batch size for memory (Section IV-C2)
N_{model}	Mini-batch size for model (Section IV-B)

III. System Model

We consider a system where NO owns cache space K at an edge node, for instance at the base station, and P SPs provide video streaming services for end users. Time is slotted. The NO can share its cache space among the P SPs, and we denote by $\theta_p^{(k)}$ the cache space allocated to SP p in time slot k. We consider that cache space is an integer K and each slot can store one object.

We denote by $\boldsymbol{\theta}^{(k)} = (\theta_1^{(k)},...,\theta_P^{(k)})$ the allocation at time slot k and we define the set of feasible allocations

$$\mathcal{T} \triangleq \left\{ \boldsymbol{\theta} | \sum_{p=1}^{P} \theta_p \leq K, \theta_p \in \mathbb{Z}^+ \right\} \quad (1)$$

Table I summarizes the most frequently used notations in this paper.

A. Request pattern

We consider that each SP p has a catalog $\mathcal{N}_p$ of $N_p = |\mathcal{N}_p|$ cacheable objects. We use the tuple (c,p), $c = 1,2,\ldots,N_p$ to refer to object c of SP p. Requests for objects arrive with rate λ. We denote by f_p the probability that a given request is for an object offered by SP p, and hence the request arrival rate for objects of SP p is $\lambda \cdot f_p$. To capture the fact that not all objects of an SP may be cacheable (e.g., video calls), we denote by ζ_p the probability that a request to SP p is for a cacheable object, and we refer to this as its *cacheability*. For a cacheable object (c,p), we denote by $\rho_{c,p}$ its popularity, i.e., the probability that, among the requests for all cacheable objects of SP p, the request is for object c. Under this model, a cacheable object (c,p) receives requests at rate $\lambda_{c,p} = \lambda \cdot f_p \cdot \zeta_p \cdot \rho_{c,p}$.

We adopt the common assumption in the literature that all objects have the same size [11], [12], [13]. Objects may represent, for instance, chunks of videos. We consider that the arrival process is stationary. In practice, object popularity and request rate change smoothly over time, and as we will show in Section VII-B our algorithm can converge fast enough (15 minutes, see Figure 4a) to be able to consider the arrival process to be stationary.

B. Cost model

The cost of the NO is due to backhaul traffic, which could be incurred for two reasons. First, for a given cache partitioning $\boldsymbol{\theta}$, the cache misses cause backhaul traffic. We call the resulting cost the *nominal cost*. Observe that for any allocation $\boldsymbol{\theta}$, the nominal cost is a random variable parameterized by parameters that are unknown/not observable by the NO, i.e., the requests of users for video objects that could changes over time. We denote these unknown parameters, which represent exogenous conditions that are not under the control of the NO, by ω. We express this dependence, for any $\boldsymbol{\theta}$, by using the notation $C_{\text{nom},p}(\boldsymbol{\theta},\omega)$ for the nominal cost due to SP p:

$$\begin{aligned} C_{\text{nom},p} : &\mathbb{R}^P \times \mathcal{X} \to \mathbb{R} \\ &(\boldsymbol{\theta},\omega) \to \sum_{c \in \omega \cap \mathcal{N}_p} 1 - \mathbb{1}_{\theta_p}(c) \end{aligned}$$

where $\mathcal{X}$ is the topological space of requests and

$$\mathbb{1}_{\theta_p}(c) = \begin{cases} 1 & \text{if } c \text{ is in the } \theta_p \text{ cached objects in the edge} \\ 0 & \text{otherwise.} \end{cases}$$

The total nominal cost due to all SPs, at time slot k, is

$$C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega) = \sum_{p=1}^{P} C_{\text{nom},p}(\boldsymbol{\theta}^{(k)}, \omega) \tag{2}$$

The second source of backhaul traffic is changing the cache partitioning between the SPs. We refer this as *perturbation cost*. To express the perturbation cost, let us denote by $\mathbf{a}^{(k)} = \boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}$ the *perturbation vector*, i.e., the change in cache partitioning of the NO at time slot k. If $\theta_p^{(k+1)} > \theta_p^{(k)}$ then SP p will download $\theta_p^{(k+1)} - \theta_p^{(k)} > 0$ objects in its allocated storage. We can thus express the perturbation cost as:

$$C_{\text{pert}}(\mathbf{a}^{(k)}) = \sum_{p=1}^{P} [\theta_p^{(k+1)} - \theta_p^{(k)}]^+$$

The *instantaneous cost* $C^{(k)}$ at time slot k is then

$$C^{(k)} \triangleq C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}^{(k)}), \tag{3}$$

and the *cumulative cost* over Z time slots as:

$$C_{\text{cum}}(Z) = \sum_{k=1}^{Z} C^{(k)} \tag{4}$$

C. Problem formulation

Since for any $\boldsymbol{\theta}$, the total nominal cost $C_{\text{nom}}(\boldsymbol{\theta}, \omega)$ is a random variable, the NO aims to minimize its expected value:

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \mathcal{T}} \mathbb{E} C_{\text{nom}}(\boldsymbol{\theta}, \omega) \tag{5}$$

We emphasize that $\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}, \omega)$ is never observable directly, only a realization $C_{\text{nom}}(\boldsymbol{\theta}, \omega)$ is. The latter can be considered as a noisy observation of $\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$, i.e., $\forall \boldsymbol{\theta} \in \mathcal{T}$

$$C_{\text{nom}}(\boldsymbol{\theta}, \omega) = \mathbb{E} C_{\text{nom}}(\boldsymbol{\theta}, \omega) + \eta, \tag{6}$$

where η is a random variable.

The problem (5) is a contextual bandit problem, where ω is the context. Algorithms for solving contextual bandit problems rely on experimenting with different cache allocations, and hence they will involve perturbation cost that they do not take into account. Hence it is more reasonable to consider the following optimization problem:

$$\pi^* = \arg\min_{\pi \in \Pi} \lim_{Z \to \infty} \frac{1}{Z} \sum_{k=1}^{Z} \mathbb{E}[C^{(k)}] \tag{7}$$

where Π is is the set of causal allocation policies.

An allocation policy π is a function $\pi(\mathbf{a}|\boldsymbol{\theta})$ defining the decisions of the NO: whenever the NO observes state $\boldsymbol{\theta}$, it will choose an action $\mathbf{a}$ with probability $\pi(\mathbf{a}|\boldsymbol{\theta})$. During training, the NO starts with a certain policy $\pi^{(0)}(\cdot)$ and then adjusts it, based on the measured cost, in order to approach the optimal policy π^*.

Note that, despite the fact that the spurious traffic generated by perturbations adds to the cost, perturbations are the only way for the NO to discover how to optimize the "black-box" function $\boldsymbol{\theta} \to \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}, \omega)$. Indeed, by observing the effects of perturbations on the nominal cost, the NO can accumulate knowledge that can be used to drive the system close to the optimal allocation $\boldsymbol{\theta}^*$. Therefore, in our data-driven approach, rather than directly solving (5), which would be infeasible for the reasons stated above, our aim is to find a sequence of perturbations $\{\mathbf{a}^{(k)}\}$ in order to minimize the expected mean cumulative cost (4), i.e.,

Observe that the NO problem is a sequential decision making problem under uncertainty, where the uncertainty is due to the randomness of the users requests. In what follows we propose an allocation policy based on RL. We will show that, by doing so, we converge close to the optimal allocation (5). Note that, for any initial allocation $\boldsymbol{\theta}^{(0)}$, the sequence $\{\mathbf{a}^{(k)}\}$ deterministically induces a sequence of states $\{\boldsymbol{\theta}^{(k)}\}$:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \mathbf{a}^{(k)} \tag{8}$$

We refer to $\{\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}\}$ as a *state-action sequence*. Adopting the standard terminology from the literature [37, Section 4.1], the observations of the NO are based on a *bandit feedback model*, in that at every time-slot k the NO observes only the cost (3) of the state-action pair visited in that time-slot and not the others.

IV. Data-driven Optimization

A. MDP formulation

Our cache allocation problem can be formulated as a deterministic MDP. The set of **states** $\mathcal{S}$ consists of all the allocation vectors that we can visit. To prevent jumping from an allocation to another which involves a change of a large number of contents which would imply a high perturbation cost and which would be detrimental to the operation of an up and running system, as is our case, we adopt a discretization step $\Delta \in \mathbb{N}$, i.e., the amount by which we change the allocation, and define $\mathcal{S}$ as:

$$\mathcal{S} = \left\{ \boldsymbol{\theta} = (\theta_1, \ldots, \theta_P) | \sum_{p=1}^{P} \theta_p \leq K, \theta_p \text{ multiple of } \Delta \right\} \tag{9}$$

The discretization step Δ constitutes, indeed, a precision/complexity trade-off. A smaller value of Δ increases the precision of the allocation since it allows to converge to a discrete solution closer to the optimal one (Section VII-A); it however increases the complexity of the problem since it expands the space of states.

Observe that $\mathcal{S} \subset \mathcal{T}$ (1). When in state $\boldsymbol{\theta}$, the NO can pick an action from the following **action space**:

$$\mathcal{A}_{\boldsymbol{\theta}} = \{\mathbf{a} = \Delta \cdot (\mathbf{e}_p - \mathbf{e}_{p'}) | \boldsymbol{\theta} + \mathbf{a} \in \mathcal{S}, p, p' = 1, \ldots, P\} \tag{10}$$

where $\mathbf{e}_p$ is the p-th element of the standard basis of $\mathbb{R}^p$.

We will use the terms allocation/state and action/perturbation interchangeably. Therefore, an action $\mathbf{a}$ consists in the NO adding Δ units of storage to a certain

SP p and removing the same amount from another SP p'. The null action corresponds to not changing the allocation (which happens in (10) when $p = p'$). Thanks to (8), the transition from a state to another is deterministic.

Our objective function accounts for both nominal cost as well as perturbation cost and is given by:

$$C^{\gamma}_{\text{cum}} = \lim_{Z\to\infty} \mathbb{E}\left[\sum_{k=0}^{Z} \gamma^{(k)} \cdot \underbrace{\left(C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}^{(k)})\right)}_{\text{Instantaneous cost } C^{(k)}}\right] \tag{11}$$

where $0 < \gamma < 1$ is a hyper-parameter called *discount factor*.

B. Online model-based RL

To properly characterize Model-based RL, we first need individual definitions of *planning* and *Reinforcement Learning*. We can distinguish them based on their knowledge about the system: planning methods assume that a *model* of the system (nominal cost $\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$ vs. allocation $\boldsymbol{\theta}$, in our case) is available (a-priori or via learning), which allows the agent to repeatedly plan forward from any state $\boldsymbol{\theta}$. In contrast, for RL in its simplest form (i.e., model free RL) this model is not present, so the agent can have information about the instantaneous cost of a certain allocation, only by physically visiting it. *Model-based RL* [38, Section 8] combines both approaches. In model-based RL, "learning" happens at two locations in the decision algorithm: 1) to learn the model of the system, and 2) to learn the policy telling us which actions to take from any state.

Since we do not know the form of the function $\boldsymbol{\theta} \to \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$, the first step of model-based RL involves learning it from observed data. For that, we construct for each SP p a model $\hat{C}^{(k)}_{\text{nom,p}}(\theta_p)$ that will approximate the nominal cost $\mathbb{E}C_{\text{nom,p}}(\theta_p)$ for any θ_p. We thus obtain a model $\hat{C}^{(k)}_{\text{nom}}(\boldsymbol{\theta}) \triangleq \sum_{p=1}^{P} \hat{C}^{(k)}_{\text{nom,p}}(\theta_p)$ that approximates the nominal cost $\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$, at time slot k.

At each time slot k,we perform the classical update of Q-learning algorithm:

$$Q^{(k)}(\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}) = (1-\alpha^{(k)}) \cdot Q(\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}) + \alpha^{(k)} \cdot \left(C^{(k)} + \gamma^{(k)} \min_{\mathbf{a}\in\mathcal{A}_{\boldsymbol{\theta}^{(k+1)}}} Q^{(k)}(\boldsymbol{\theta}^{(k+1)}, \mathbf{a})\right) \tag{12}$$

Then, we take the last measured value $C_{\text{nom,p}}(\theta_p^{(k)})$ and do the following:

- We construct model $\hat{C}^{(k)}_{\text{nom}}(\boldsymbol{\theta})$.
- Then, we sample N_{model} random state-action pairs from the state space $\mathcal{S}$ and action space $\mathcal{A}$. Let us call $\{(\boldsymbol{\theta}^{[i]}, \boldsymbol{a}^{[i]}), \boldsymbol{\theta}^{[i]} \in \mathcal{S}, \boldsymbol{a}^{[i]} \in \mathcal{A}_{\boldsymbol{\theta}^{[i]}}\}_{i=1,\dots,N}$ such samples. For each i-th sample, we predict a cost $\hat{C}^{[i]} = \hat{C}_{\text{nom}}(\boldsymbol{\theta}^{[i]}) + C_{\text{pert}}(\boldsymbol{a}^{[i]})$. Note that in this way we are able to make predictions on state-action pairs that have not been visited yet, exploiting "similar" pairs observed in the past.
- We finally perform N_{model} Q-table updates, as in (12), using $\hat{C}^{[i]}$ calculated above in place of $C^{(k)}$ of formula (12).

In line 20 of Alg. 1, we estimate a model of the nominal cost from the collected observations. Such a model is then used to perform additional updates of the Q-table. It will be required in the proof of Theor. V-B.1[4] that this model has to be unbiased, in order to avoid driving Q-table updates in the wrong direction. A simple empirical average of the observed nominal costs would be an unbiased model, but we need to collect many samples before empirical averages are sufficiently close to the expected value of the nominal cost. Moreover, at the beginning of the learning process, a lot of allocations have not yet been visited, and thus empirical averages would not be all available. For this reason, in the first iterations, we fit a regression model on the collected observations. This model also gives estimates of the nominal cost in the allocations not yet visited, which allows to "accelerate" the updates of the Q-table at the beginning. To obtain unbiasedness, we then gradually abandon the regression model and we adopt the empirical averages.

We now formally describe how we estimate the cost model in Line 20 of Alg. 1. Up to a certain time slot K_{reg}, model $\hat{C}^{(k)}_{\text{nom},p}(\theta_p)$ is obtained by regression using $\mathcal{D}_p$ as dataset. Let us denote $\hat{C}^{(k)}_{\text{nom},p,reg}(\theta_p)$ the model obtained by regression. Then, we gradually replace $\hat{C}^{(k)}_{\text{nom},p,reg}(\theta_p)$ with the empirical mean:

$$\overline{C}^{(k)}_{\text{nom},p}(\theta_p) \triangleq \frac{1}{|\mathcal{K}^{(k)}_{p,\theta_p}|} \sum_{k'\in\mathcal{K}^{(k)}_{p,\theta_p}} C_{\text{nom},p}(\theta_p, \omega^{(k')}) \tag{13}$$

where $\mathcal{K}^{(k)}_{p,\theta_p}$ is the set of time-slots $k' \le k$, in which SP p has been allocated θ_p slots.

We define our model at time slot k for any $\theta_p \in [0, K]$ as:

$$\hat{C}^{(k)}_{\text{nom},p}(\theta_p) \triangleq \begin{cases} \hat{C}^{(k)}_{\text{nom},p,\text{reg}}(\theta_p) & \text{if } k \le K_{\text{reg}} \text{ or } \mathcal{K}^{(k)}_{p,\theta_p} = \emptyset \\ \frac{1}{|\mathcal{K}^{(k)}_{p,\theta_p}|}\hat{C}^{(k)}_{\text{nom},p,\text{reg}}(\theta_p) + \left(1 - \frac{1}{|\mathcal{K}^{(k)}_{p,\theta_p}|}\right)\bar{C}^{(k)}_{\text{nom},p}(\theta_p) & \text{otherwise} \end{cases} \tag{14}$$

Similar to (2), the cost estimated by the model is $\hat{C}^{(k)}_{\text{nom}}(\boldsymbol{\theta}) \triangleq \sum_{p=1}^{p} \hat{C}^{(k)}_{\text{nom},p}(\theta_p)$ and the empirical mean of the nominal cost measured is $\bar{C}^{(k)}_{\text{nom}}(\boldsymbol{\theta}) \triangleq \sum_{p=1}^{p} \bar{C}^{(k)}_{\text{nom},p}(\theta_p)$.

The following theorem ensures that our model is an unbiased estimator of the nominal cost.

Theorem IV-B.1. *For any SP p and allocated cache slots θ_p our model converges uniformly to the expected value of the nominal cost, i.e.,*

$$\lim_{k\to\infty}{}^{u} \hat{C}^{(k)}_{nom}(\cdot) = \mathbb{E}C_{nom}(\cdot) \tag{15}$$

where symbol $\lim^u$ indicates that

$$\lim_{k\to\infty} \|\hat{C}^{(k)}_{nom}(\cdot) - \mathbb{E}C_{nom}(\cdot)\|_{\infty} = 0 \tag{16}$$

[4]In particular in Theor. B-A.1, which is needed for Theor. V-B.1

In case a simulation model of the system is available, it can be used in lieu of the regression model. In this case, instead of fitting the regression model, we would use observations to calibrate the simulator, i.e., to find the simulation parameters that better match the observed costs.

C. Additional enhancements

We now report some enhancements that considerably improve the performance of our algorithm.

1) Learning rate scheduling: The hyper-parameter $\alpha^{(k)}$ in (12) tells the magnitude of step that is taken towards the solution. $\alpha^{(k)}$ should not be too big a number as it may continuously oscillate around the minima and it should not be too small of a number else it will take a lot of time and iterations to reach the minima. As in [12], we decrease it slowly, because initially when we are at a totally random point in solution space we need to take big leaps towards the solution and later when we come close to it, we make small jumps and hence small improvements to finally reach the minima.

$$\alpha^{(k)} = \alpha^{(k-1)} \cdot \left(1 - \frac{1}{1+M+k}\right)^{\frac{1}{2}+\xi} \tag{17}$$

where $M, \xi > 0$ are constants that tune the slope of decrease.

2) Experience replay: In the simplest implementation of Q-learning, the measurement made in a certain time-slot is used to update the Q-table in that time-slot only and is never used again. However, the set of previous measurements (i.e., the past "experience") could be further exploited to improve the Q-table update in future time-slots. To this aim, Experience Replay has been proposed [39]. At any time-slot k, in addition to using the measured instantaneous cost $C^{(k)}$ to update the Q-table in (12), we also store this measurement in the form of a triplet $\left(\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}, C^{(k)}\right)$, which we call *experience*. The set of experiences accumulated in this way is called *memory*. Formally, let us define the memory as:

$$\mathcal{M}^{(k)} = \{(\boldsymbol{\theta}^{(0)}, \mathbf{a}^{(0)}, C^{(0)}), \ldots, (\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}, C^{(k)})\}$$

Whenever we update the Q-table, additionally to performing (12) using the current observation, we also sample the memory randomly for a mini-batch of experiences of size N_{memory} and we use these random samples as entries for the Q-table by applying (12).

3) ϵ stretched exponential decay: The value of $\epsilon^{(k)}$ is the probability of taking a random action (exploration) instead of the best so far, at any time-slot k. We schedule $\epsilon^{(k)}$ as in Fig. 2:

$$\epsilon^{(k)} = \begin{cases} \epsilon_0 - \left[\frac{0.9\cdot\epsilon_0}{\cosh(e^{-\frac{k-A\cdot Z}{B\cdot Z}})} + \frac{k\cdot C}{Z}\right] & \text{if } k \le Z \\ \frac{\epsilon^{(Z)}}{k-Z} & \text{otherwise} \end{cases} \tag{18}$$

where ϵ_0 is the initial value of ϵ. We chose this particular shape of scheduling, since it induces an Exploration phase (high values of $\epsilon^{(k)}$) followed by an Exploitation phase, which can be controlled in a simple way. Hyperparameter A determines the length of the Exploration phase. Hyperparameter B determines the slope of the transition from Exploration to Exploitation. Hyperparameter C controls the steepness of left (Exploration) and right (Exploitation) tails of the curve of the $\epsilon^{(k)}$ evolution. This shape is inspired by common practice [40], but to preserve enough exploration, we introduced hyperparameter Z, which we call *time horizon*, after which the slope of decrease of $\epsilon^{(k)}$ diminishes. Overall, our chosen decay scheduling provides:

- sufficient time for exploration at the beginning.
- preference to exploitation (with respect to exploration) in the end (quasi-deterministic policy).
- smooth transition while switching from exploration to exploitation.

Alg. 1 describes how we combine model-based RL with the enhancements cited above.

V. Theoretical Properties

A. Memory complexity

Since the size of the state space $\mathcal{S}$ is $O((K/\Delta)^P)$ and of the action space $\mathcal{A}$ is $O(P^2)$, the memory required to store the Q-table is $O((K/\Delta)^P \cdot P^2)$. We assume that the number of SPs getting a cache slice is limited (3 or 4). The cubic dependence on P is a limited issue under this assumption. Observe that we can counter-fight the linear increase of memory complexity with respect to the cache size K by proportionally increasing the elementary allocation Δ.

B. Convergence

In our algorithm we make use of a discretization constant Δ, which may be larger than 1. The latter limits the state space $\mathcal{S}$ (9) and consequently prevents us from reaching the optimal allocation $\boldsymbol{\theta}^*$ defined by (5). We can thus only reach a *discretely optimal allocation*:

$$\hat{\boldsymbol{\theta}}^* \in \arg\min_{\boldsymbol{\theta}\in\mathcal{S}} \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}, \omega) \tag{19}$$

The following theorem, proved in the Appendix, shows that our allocation converges to the discretely optimal one.

Theorem V-B.1. *If the discount factor γ is sufficiently close to 1*

$$\lim_{k\to\infty} \boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^* \text{ with probability } 1.$$

Sketch of the proof. The main steps to prove the above theorem are:

- We prove that our Q-table $Q^{(k)}$ converges to the optimal Q-table Q^* with probability 1.
- We prove that the sequence of actions and states induced by Q^* has an absorbing state that is the discretely optimal state $\hat{\boldsymbol{\theta}}^*$.
- We prove that the sequence of actions and states induced by our Q-table $Q^{(k)}$ also follows Q^*.
- We prove that the sequence of actions and states that we take online converges with probability 1 to the sequence induced by our Q-table $Q^{(k)}$ (assuming no more exploration).

Algorithm 1: k-th step of Model-based RL

$\alpha^{(k)} \leftarrow$ calculate the value of α ; // formula (17)
$\epsilon^{(k)} \leftarrow$ calculate the value of ϵ ; // formula (18)
with probability $\epsilon^{(k)}$: $\mathbf{a}^{(k)} \leftarrow$ random action ; // ϵ-greedy policy
with probability $1-\epsilon^{(k)}$: $\mathbf{a}^{(k)} \leftarrow$ best action from $Q^{(k)}(\boldsymbol{\theta}, \mathbf{a})$;
$\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \mathbf{a}^{(k)}$;
$C^{(k)} \leftarrow C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}^{(k)})$;
$Q^{(k)}(\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}) \leftarrow (1-\alpha^{(k)}) \cdot Q^{(k)}(\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}) + \alpha^{(k)} \cdot \left(C^{(k)} + \gamma \min_{\mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}^{(k+1)}}} Q^{(k)}(\boldsymbol{\theta}^{(k+1)}, \mathbf{a})\right)$; // update $Q^{(k)}$
////////////////
/// Memory replay
$\mathcal{M}^{(k)} \leftarrow \mathcal{M}^{(k-1)} \cup \{(\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}, C_{\text{nom}}(\boldsymbol{\theta}^{(k)}))\}$;
for N_{memory} *times* ; // N_{memory} is the size of the memory mini batch
do
  $(\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}, C^{\text{rd}}_{\text{nom}}) \leftarrow$ random element from $\mathcal{M}^{(k)}$;
  $\boldsymbol{\theta}'^{\text{rd}} \leftarrow \boldsymbol{\theta}^{\text{rd}} + \mathbf{a}^{\text{rd}}$;
  $Q^{(k)}(\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}) \leftarrow (1-\alpha^{(k)}) \cdot Q^{(k)}(\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}) + \alpha^{(k)} \cdot \left(C^{\text{rd}}_{\text{nom}} + C_{\text{pert}}(\mathbf{a}^{\text{rd}}) + \gamma \min_{\mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}'^{\text{rd}}}} Q^{(k)}(\boldsymbol{\theta}'^{\text{rd}}, \mathbf{a})\right)$; // update $Q^{(k)}$
end
////////////////
/// Model training and inference
$\mathcal{D}_p^{(k)} \leftarrow \mathcal{D}_p^{(k-1)} \cup \{(\theta_p^{(k)}, C_{\text{nom,p}}(\theta_p^{(k)})\}$; // collect realization of $C_{\text{nom,p}}(\theta_p^{(k)})$ for each SP p
$\hat{C}^{(k)}_{\text{nom},p}(\theta_p) \leftarrow$ estimate model from $\mathcal{D}_p^{(k)}$ **for** N_{model} *times* ; // N_{model} is the size of the model mini batch
do
  $\boldsymbol{\theta}^{\text{rd}} \leftarrow$ random state from $\mathcal{S}$;
  $\mathbf{a}^{\text{rd}} \leftarrow$ random action from $\mathcal{A}_{\boldsymbol{\theta}^{\text{rd}}}$;
  $\boldsymbol{\theta}'^{\text{rd}} \leftarrow \boldsymbol{\theta}^{\text{rd}} + \mathbf{a}^{\text{rd}}$;
  Compute $\hat{C}^{(k)}_{\text{nom,p}}(\theta_p^{\text{rd}})$; // predict the nominal cost using the model
  $\hat{C} \leftarrow \sum_{\text{p=1}}^{P} \hat{C}_{\text{nom,p}}(\theta_p^{\text{rd}})) + C_{\text{pert}}(\mathbf{a}^{\text{rd}})$;
  $Q^{(k)}(\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}) \leftarrow (1-\alpha^{(k)}) \cdot Q^{(k)}(\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}) + \alpha^{(k)} \cdot \left(\hat{C} + \gamma \min_{\mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}'^{\text{rd}}}} Q^{(k)}(\boldsymbol{\theta}'^{\text{rd}}, \mathbf{a})\right)$; // update $Q^{(k)}$
end

- Finally, we show that this sequence converges with probability 1 to a sequence induced by Q^*.

□

Let us define the *discretization gap* G_Δ as the loss induced by the fact that we do not allocate cache slot by slot, but only in multiples of the discretization step Δ:

$$G_\Delta = \mathbb{E}C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}^*, \omega) \tag{20}$$

Intuitively, the larger the discretization step Δ, the larger the discretization gap G_Δ. The following bound (proved in the Appendix) shows this dependence between Δ and G_Δ.

Proposition V-B.2. $G_\Delta \leq \sum_{p=1}^{P} \sum_{c=1}^{\Delta} \lambda_{c,p}$.

A consequence of Theor. V-B.1, proved in the Appendix, is the following:

Corollary V-B.3. *Let us denote by* $C_{cum}(Z)$ *and* $C^*_{cum}(Z)$ *the cumulative cost* (4) *obtained via Q-learning and via the optimal theoretical allocation* $\boldsymbol{\theta}^*$*, respectively. The difference between the two converges in expectation to the discretization gap:*

$$\lim_{Z \to \infty} \frac{1}{Z} \mathbb{E}\left[C_{cum}(Z) - C^*_{cum}(Z)\right] = G_\Delta$$

Therefore, a trade-off emerges: one the one hand, discretization allows us to reduce the memory complexity, i.e., state space and action space (Section V-A), on the other hand it keeps allocations at a finite distance from the theoretical optimal cost. We will however see in the numerical results (Section VII) that this distance is in practice very small.

VI. Discussion on the use of RL

We now briefly discuss why we preferred our RL setting over other possible methodologies. First of all, we rule out all the static optimization techniques that require full information, due to the online and stochastic nature of the problem at hand.

Simultaneous Perturbation Stochastic Approximation (SPSA) has been applied in [41] for instance to optimize systems in a stochastic environment. However, the objective of such method is just to converge towards the optimum and no cost is considered for perturbation, the system is continuously perturbed along multiple directions. In real situations, perturbing a system can prevent it from working properly or engenders high cost (as in our problem). Our RL formulation allows to naturally include the cost of perturbation in the optimization problem and to consider the trade-off between converging fast toward the optimum and limiting the extent and frequencies of perturbations.

We could also interpret our allocation problem at hand as a "black-box optimization": we have an unknown system whose cost function is unknown ($C_{\text{nom}}(\boldsymbol{\theta}, \omega)$ in our case) and we aim to find the optimal solution $\boldsymbol{\theta}^*$. In such problems, Bayesian Optimization techniques [42] trade-off exploration and exploitation to pick the states to visit, observe the cost at those states and decide the next state to visit, up to finding the optimum. However, such techniques suffer from the same limitation of SPSA: they are meant for offline problems, where the objective is to retrieve the minimum of the cost function *at the end of the optimization* and the cost of jumping from one state to another is not quantified nor directly minimized. Our RL framework not only allows us to reach an allocation close to the optimum at the end, but also implicitly optimizes the path of states visited *during* the optimization.

Lyapunov Optimization (LO) has also been used for allocation problems [43], [44]. However, these works assume that the expression of the cost or reward function ($C_{\text{nom}}(\boldsymbol{\theta}, \omega)$ in our case) is known, the only unknown information is the sequence

of future realizations ω of the environment. We do not need to rely on such a strong assumption, as RL allows to optimize the system even if the expression of the cost function remains unknown.

The Markov Decision Process (MDP) underlying our RL method is a Deterministic MDP (DMDP), as the transition from one state to another is deterministic in our case. In [45], transitions are unknown (although deterministic) and authors use model free Q-learning to solve the MDP. However, we focus on an infinite-horizon deterministic control system with an approximation model for the reward (cost).

Our problem can be described as a MAB with switching cost (MAB-SC) [46], [47]. In that context, we would need to interpret each allocation vector as an arm and the perturbation cost as a switching cost. We indeed solve in our case this MAB-SC problem via model-based RL and as any randomized strategy, our algorithm is subject to the bounds derived in this context (Theor. 1 of [47]). We resort to model-based RL instead of other algorithms proposed in the MAB context, like EXP3 [48], as we can naturally embed in our algorithm the model $\boldsymbol{\theta} \rightarrow \hat{C}_{\text{nom}}(\boldsymbol{\theta})$ of the nominal cost, inferred via simple regression, which would not be as immediate to do using other MAB algorithms. Observe that, as in [49], our problem is not a contextual MAB, as the actions determine the states.

Our problem is not an adversarial MAB [50] either. Indeed, the focus of adversarial MAB is on worst-case analysis: performance bounds are provided under the assumptions that the environment (the value of ω in our case) or even the actual cost function expression $C_{\text{nom}}(\boldsymbol{\theta}, \omega)$ are chosen by an adversary in order to "hurt" the decision maker (the NO in our case) and to increase the cost of the system as much as possible. In such a setting, [51, Theor.3.1] shows that it is impossible to effectively optimize a DMDP such as ours. For our case, adversarial analysis is too pessimistic, as the worst case (in our case it would be for instance users generating a huge amount of requests only for unpopular non-cached objects) would have a negligible probability. We instead adopt a stochastic setting, where the realizations ω of the environment are taken from a probability distribution that is independent of the NO decisions and we study the "average" behavior of the system, i.e., the expected value of the cost and the probability to converge close to the optimum.

Online decision problems have been presented in an adversarial setting: relevant to our case is Smoothed Online Convex Optimization (SOCO) [52], [53], which aims to optimize the cost of a system by also taking into account the perturbation cost to jump from one state to another. However, such algorithms are studied in an adversarial setting, while we are in a stochastic setting, as mentioned above.

We have performed some performance analysis using R-learning instead of Q-learning. However, we observed worst achieved cost (which confirms previous findings from the literature [54]) and we omit the obtained results.

VII. Numerical Results

We now evaluate the performance of our RL allocation $\boldsymbol{\theta}^{(k)}$ through simulations developed in Python and compare it to two static allocations: (i) the theoretical optimal allocation $\boldsymbol{\theta}^*$, which would ideally be computed by an oracle who knows exactly the content popularity and thus the expression of function $\boldsymbol{\theta} \rightarrow \mathbb{E}C(\boldsymbol{\theta})$, (ii) the proportional allocation $\boldsymbol{\theta}^{\text{prop}}$ where θ_p is proportional to the rate of requests λ_p directed to SP p. We also compare it to two dynamic allocation strategies, namely (iii) the model-free RL version of our algorithm that we implemented in prior work [25] and (iv) SPSA [12].

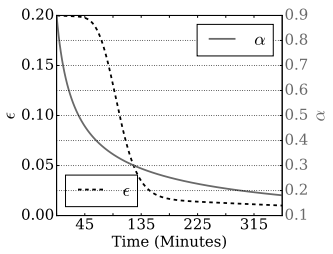


Fig. 2: Evolution of ϵ and α vs. time

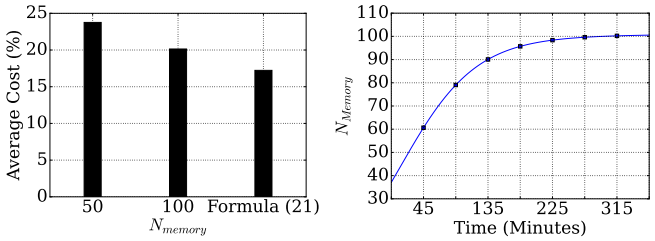
We consider a network with 3 SPs. We set the overall request arrival rate to $\lambda = 4 \cdot 10^3 req/s$ (the same order of magnitude of requests supported in one edge location of Amazon CloudFront [55]). Each of these requests is directed to SP 1, 2 or 3 with probabilities 0.75, 0.20 and 0.05, respectively. Note that it is the request rate, independent of the number of users that generate it, which determines the miss rate and, in turn, the cost that we want to minimize. We set the cacheability (Section III-A) of SP_1, SP_2 and SP_3 to ζ_1 = 0.4, ζ_2 = 0.9 and ζ_3 = 0.9, respectively. Each SP has a catalog of $N_1 = N_2 = N_3 = 10^7$ cacheable objects. Content popularity in each catalog follows Zipf's law with exponents β_1 = 1.2, β_2 = 0.4 and β_3 = 0.2, respectively. The total cache size is $K = 5 \cdot 10^6$ slots. Assuming that 1 slot corresponds to 15 MB (in the order of a chunk size), the total cache size in MB is $75 \cdot 10^6 = 75$ TB ≈ 3 times the storage of a Netflix Open Connect Appliance [56]. Even with such storage capacity deployed in the edge, we cannot cache the entire catalogs, i.e., one catalog size is 150 TB. Moreover, the same video could be stored in 20 different versions by Netflix, to adapt it to different codecs, devices and resolutions. The simulation time is set to Z = 6 hours. The length of a time-slot is 0.25 second.

We plot a normalized cost, i.e., the amount of objects downloaded from the Internet (either as a result of an edge cache miss or of an allocation perturbation) divided by the total amount of objects requested by the users. All curves are averaged with a sliding window of 10 min.

A. Pre-tuning of hyper-parameters

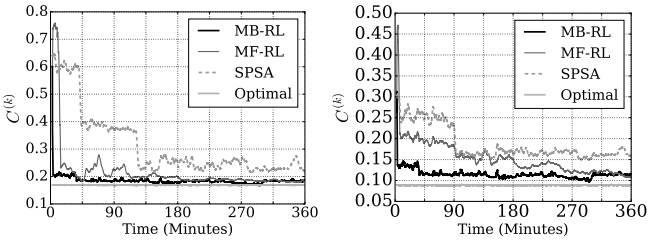
We now discuss some preliminary tuning that we performed in experimentation not shown in this paper on the features indicated in Section IV-C.

1) For the discretization step Δ, we found out that a good complexity vs. precision trade-off was to set it to $K/50$. To limit perturbations, we give a higher "weight" to the null action. Indeed, when we take a random action, we set the probability of choosing any non-null action to only



(a) N_{memory} effect (b) Variation of N_{memory} over time

Fig. 3: Choice of N_{memory}



(a) Default scenario (3 SPs) (b) Scenario with 4 SPs (§VII-D)

Fig. 4: Evolution of total instantaneous cost $C^{(k)}$.

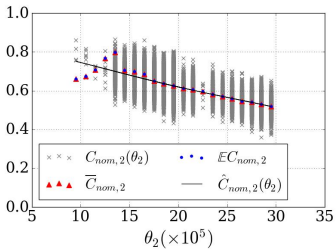


Fig. 5: Model $\hat{C}_{\text{nom,p}}(\theta_p), p = 2$

$1/P^2$ and all the remaining probability is for the null-action.

2) We set the discount factor γ to 0.99, i.e., very close to 1 to give importance to future rewards and prevent myopic decisions.
3) For α, the learning rate, we found that convergence was slow when it was fixed. Therefore, we adopt learning rate scheduling, which starts at 0.9 and decreases following (17) to 0.2, with $M = 3600$ and $\xi = 0.01$. The variation of α is illustrated in Fig. 2 (red curve).
4) We make ϵ decay as in (18) with $A = 0.3$, $B = 0.1$ and $C = 0.01$. Z = 6 hours. These hyperparameters have been chosen empirically after experimentation and provide a good compromise between exploration and exploitation. The black curve in Fig. 2 shows the decay of ϵ.
5) Regarding size N_{memory} of the mini-batch of experiences, we found that small fixed values were not allowing to exploit past experience, on the other hand, with large values past experience was excessively dominating the updates. We thus built a scheduling function that is (i) increasing, so as to collect sufficient past experience before giving it importance, (ii) concave and with a plateau, so as to avoid excessive dominance of past experience over current observations. After experimenting with scheduling curves having the aforementioned characteristics, we obtained the best performance with

$$N_{\text{memory}}^{(k)} = \frac{N_{\max}}{\cosh(e^{-\frac{k-A\cdot Z}{B\cdot Z}})} + \frac{k\cdot C}{Z} \tag{21}$$

where $N_{\max}$ = 100, A = 0.15, B = 0.3, C = 0.7, Z = 6 hours. The choice of N is illustrated in Fig. 3.
6) For N_{model}, we take at each time-slot $N_{\text{model}} = 50$ samples on which we will apply the model $\hat{C}_{\text{nom}}(\boldsymbol{\theta})$. We rely here on empirically tuning the number of samples N_{model}, striking a balance between computational cost and sample effectiveness.

B. Convergence close to the optimum

The behavior of our algorithm is well illustrated by Fig. 4a: Our MB-RL algorithm learns the system in only 15 minutes and converges to a cost close to the theoretical optimum. This rapid convergence is achieved thanks to the accurate model we obtain by regression (Fig. 5) and that we use to estimate the nominal cost for any given allocation (even if the state is not visited). Although this paper focuses on a stationary scenario, the results suggest that the fast convergence of our MB-RL algorithm could allow us to adapt even if the system parameters change over time. For instance, even if the request pattern is non stationary, our model can adapt to any new request pattern in 15 minutes. For model-free Reinforcement Learning (MF-RL), instead, we do not construct a model while learning the system behavior and we update the Q-table solely by perturbing the system by taking relatively many suboptimal actions in a first phase, in order to learn it. For this reason, perturbation cost is high up to 135 minutes. (see Fig. 7a) After that, we start to exploit the collected knowledge and we limit perturbation.

Furthermore, our RL algorithm outperforms SPSA used in [12], which converges to the optimal allocation in 45 minutes but never reaches the optimum due to the continuous perturbations it has to apply to estimate the sub-gradient of the objective function.

We now compare the cost $C^{(k)}$ induced by our policy with the cost of the static proportional allocation $\boldsymbol{\theta}^{\text{prop}}$ (proportional to the rate of requests directed to each SP). Note that while our method deals with both nominal and perturbation costs (3), the static $\boldsymbol{\theta}^{\text{prop}}$ does not apply any perturbation to the system. We define the gain of our policy with respect to $\boldsymbol{\theta}^{\text{prop}}$ as:

$$G_{\text{prop}}^{(k)} = \frac{C_{\text{nom}}(\boldsymbol{\theta}_{\text{prop}}, \omega) - C^{(k)}}{C_{\text{nom}}(\boldsymbol{\theta}_{\text{prop}}, \omega^{(k)})} \tag{22}$$

Fig. 6 shows that our solution reaches a gain of 60% in less than 45 minutes with respect to $\boldsymbol{\theta}^{\text{prop}}$.

To confirm our findings, we plot in Fig. 7a the perturbation cost $C_{\text{pert}}(\mathbf{a}^{(k)})$ for the model-based and model-free versions of our algorithm and for SPSA. Results confirm that after 15 minutes, $C_{\text{pert}}(\mathbf{a}^{(k)})$ is practically null on an average 10 minute window for the model-based RL which means that we no longer drastically change the allocation. In the case of model-free RL, instead, we have to wait about 3 hours for the system to stabilize and the perturbations to be negligible.

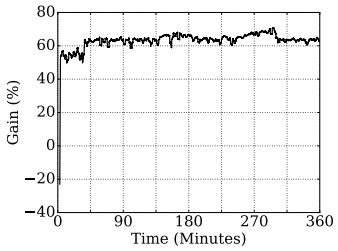
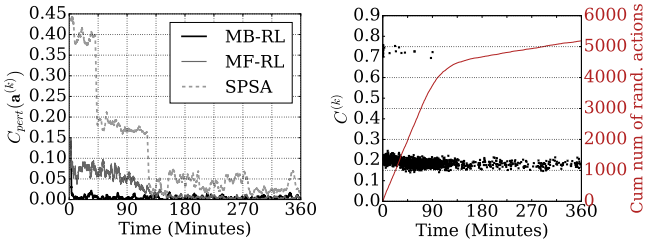
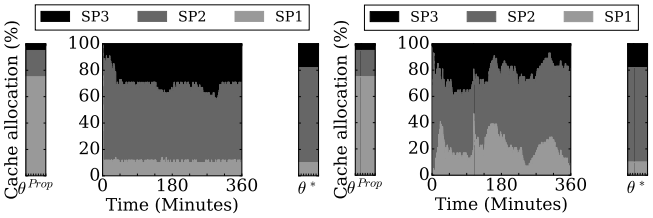


Fig. 6: Gain of MB-RL with respect to proportional allocation



(a) Perturbation Cost $C_{\text{pert}}(\mathbf{a}^{(k)})$ (b) Random actions

Fig. 7: System perturbation



(a) MB-RL (b) MF-RL

Fig. 8: Evolution of the allocation over time

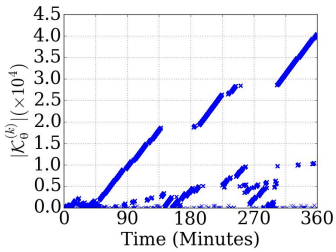


Fig. 9: Number of times current $\boldsymbol{\theta}$ is visited

This validates two findings: (i) our algorithm converges to a cache configuration, in both versions, and moves from there with a small probability, according to Theor. V-B.1 and (ii) model-based RL converges 10 times faster than model-free RL. Observe that for SPSA, $C_{\text{pert}}(\mathbf{a}^{(k)})$ is always higher than in our RL algorithms in both versions, which is expected since SPSA consists in continuously perturbing the allocation.

We map in Fig. 7b each value of $C^{(k)}$ with the nature of the action taken at the time-slot k: each black point represents a random action which means the agent is exploring the environment. The large number of black points in the beginning confirms that the first phase is an exploration phase and it corresponds to the high value of ϵ, then the number of perturbations starts to decrease as the value of ϵ decreases in order to limit the exploration as stated in Section IV-C3. We plot in the same figure the cumulative number of random actions: we observe that it increases rapidly in the first 90 minutes and then starts to stabilize in the rest of the simulation.

We plot in Fig. 8a and in Fig. 8b the evolution of the allocation $\boldsymbol{\theta}^{(k)}$ of the MB-RL and MF-RL algorithms, respectively, over time (center of each figure), the proportional allocation $\boldsymbol{\theta}^{\text{prop}}$ (left) and the optimal allocation $\boldsymbol{\theta}^{*}$ (right). Note that we start by $\boldsymbol{\theta}^{0} = \boldsymbol{\theta}^{\text{prop}}$. The results show that we converge to an allocation $\hat{\boldsymbol{\theta}}^{*}$ close to $\boldsymbol{\theta}^{*}$ and we almost stay in this allocation, which matches our theoretical result stated in Theor. V-B.1. MB-RL algorithm clearly exhibits greater stability and convergence speed compared to MF-RL, as it leverages an explicit model of the environment.

In Fig. 9, we plot, per each timeslot k, how many times the current state $\boldsymbol{\theta}^{(k)}$ has been visited in the past. We see that between 0 and 15 minutes a lot of states are visited few times, which corresponds to the exploration phase: the agent keeps jumping from one state to another to learn the optimal policy. After 15 minutes, the time in which our algorithm converges, we observe an almost linear behaviour: the agent keeps visiting the same allocation $\hat{\boldsymbol{\theta}}^{*}$ (the absorbing state). The few points that we observe out of the linear behaviour represent the few random actions taken by the agent because $\epsilon > 0$. As long as the value of ϵ is non zero, the agent will choose a random action at some point, even after reaching $\hat{\boldsymbol{\theta}}^{*}$. The figure shows that our RL agent exploits more and more the good states, but never completely ceases visiting other states for exploration purposes.

C. Fairness

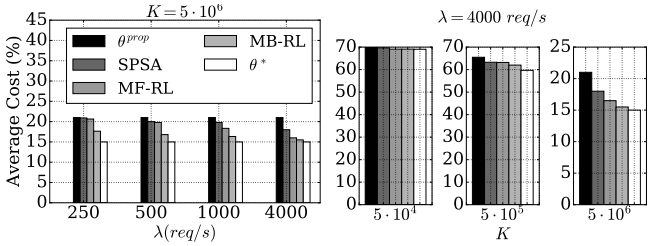
Let $x_p = \frac{\theta_p}{\zeta_p \cdot \lambda \cdot f_p}$ denote the slots given to SP p, normalized to its amount of cacheable requests. We compute the fairness of the system with the Jain's fairness index [57]:

$$\mathcal{J}(x_1, \ldots, x_P) \triangleq \frac{(\sum_{p=1}^{P} x_p)^2}{P \cdot \sum_{p=1}^{P} x_p^2} \tag{23}$$

Our results show that cache sharing strategy with our MB-RL allocation (0.7 fairness) is much fairer than the optimal allocation $\boldsymbol{\theta}^{*}$ (0.36 fairness), at almost the same total cost. It is also close to that of the proportional allocation $\boldsymbol{\theta}^{\text{prop}}$ (0.85 fairness) albeit being much better in terms of cost. Note that we are also close to the ideal maximum fairness achieved by the proportional allocation not taking into account cacheability, i.e., if all contents were cacheable (i.e., $\zeta_p = 1, p = 1, .., P$). The latter is 1, by construction, as it is proportional to the rate of requests directed to each SP; on the other hand, it is an artificial measure, as it ignores cacheability.

D. Sensitivity analysis

We next study how the performance of our solution is affected by the request rate λ and the cache capacity K. In



(a) To request rate λ (b) To cache size K

Fig. 10: Sensitivity of the system

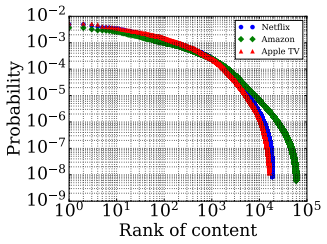


Fig. 11: Content popularity

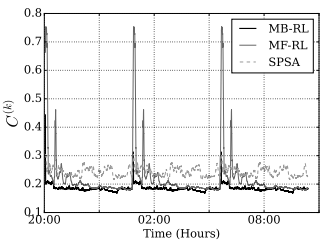


Fig. 12: $C^{(k)}$ for real dataset

Fig. 10 we plot the average cost $\frac{1}{Z}C_{\text{cum}}(Z)$ (4) of MB-RL and MF-RL algorithm, after $Z = 6$ hours, and compare it to the static proportional and optimal allocations.

Let us first focus on the request rate λ. A small λ implies that only few requests are observed in each time slot, which may result in a high noise, as defined in (6), and ultimately affects the accuracy of the update of the Q-table and slows down the convergence. We thus expect any data-driven approach to perform best with large λ. This is evident for MF-RL and SPSA (Fig. 10a), whose cost is far from the optimum for $\lambda \leq 1000$ req/s. It is however interesting to observe that MB-RL performs relatively well even with few requests per second. This shows that embedding the inferred model into the RL agent increases the sample efficiency of our method.

Fig. 10b shows the average cost measured over $Z = 6$ hours for various cache sizes $K \in \{5{\cdot}10^4, 5{\cdot}10^5, 5{\cdot}10^6\}$ and a fixed request rate $\lambda = 4 \cdot 10^3 req/s$. It confirms that the gains of our MB-RL algorithm hold for different cache sizes, and shows that gain increases for larger caches. Indeed, for a small cache size there is not much to optimize: the cost is high with both proportional and optimal allocations, so even if MB-RL and MF-RL position themselves between the two, the improvement in cost is negligible.

Compared to SPSA, we observe in Fig. 10 that our algorithm performs better in any configuration of the system.

We finally verify that the good performance of MB-RL is maintained when increasing the number of SPs. We simulate a scenario in the same conditions as in Section VII-B but we change the number of SPs to $P = 4$. Each of the requests is directed to SP 1, 2, 3 or 4 with probabilities 0.60, 0.20, 0.10 and 0.10, respectively. We set the cacheability of SP_1, SP_2, SP_3 and SP_4 to ζ_1 = 0.5, ζ_2 = 0.7, ζ_3 = 0.9 and ζ_4 = 0.9, respectively. Each SP has a catalog of $N_1 = N_2 = N_3 = N_4 = 10^7$ cacheable objects. Content popularity in each catalog follows Zipf's law with exponents β_1 = 2.2, β_2 = 0.4, β_3 = 0.1 and β_4 = 0.1, respectively. The total cache size is $K = 5 \cdot 10^6$. The simulation time is maintained at Z = 6 hours. The length of a time-slot is maintained at 0.25 second and total request rate at $\lambda = 4 \cdot 10^3 req/s$.

As in Section VII-B, we plot in Fig. 4b the total cost $C^{(k)}$ of our MB-RL algorithm, the optimal allocation $\boldsymbol{\theta}^*$, MF-RL and SPSA. The results show that our algorithm rapidly converges close to optimal cost, outperforming SPSA and MF-RL.

Such small P values remain reasonable as the number of video streaming SPs using EC (as described in our vision) is still limited to big players such as Netflix [56].

E. Results with real-world datasets

To test our optimization in a more realistic scenario, we use datasets from Netflix [58], Amazon [59], and Apple TV [60]. To estimate the relative popularity of movies within each of the 3 SPs, we normalize the number of times a movie was voted to obtain the probability distributions of Fig. 11. Request rate λ is non-stationary and is extracted from the Shanghai Telecom dataset [61], including ∼7 million requests from 9481 edge devices to simulate the overall request arrival rate. We make normalized probability f_p that a request is for SP p=Netflix, Amazon, Appel TV vary with time as follows [62]:

	Netflix	Amazon	Apple TV
01 am - 06 am	0.1198	0.3575	0.5228
06 am - 11 am	0.0425	0.1002	0.8572
08 pm - 01 am	0.1325	0.3042	0.5633

Fig. 12 shows the superior ability of MB-RL to adapt rapidly to the temporal changes of the load, thanks to the model that reduces the reliance on blind trial-and-error.

VIII. Conclusion and Future Work

We proposed a model-based RL algorithm for online edge cache allocation among several SPs, assuming encrypted, not all cacheable content: a major challenge of in-network caching. Our aim is not only to minimize cost, in terms of miss rate, but also to optimize the way to achieve that, through minimizing perturbations. We proved that our algorithm converges to an absorbing discrete optimal state with probability 1 for an infinite horizon. Simulations in several scenarios, including synthetic and real traces of user arrivals, showed that our algorithm converges to a configuration close to the optimal one, much faster than the compared allocation strategies. This allows to drastically reduce overall system cost.

References

[1] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE TPDS*, 2021.
[2] (2020) Internet IP transit provider. [Online]. Available: https://www.thousandeyes.com/learning/techtorials/transit-provider
[3] G. Çakmak and H. Suomi, "A comparison of ISP and MNO interconnection models," in *ICT*, 2014.
[4] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger, "Anatomy of a large european IXP," in *ACM SIGCOMM*, 2012.
[5] V. Giotsas *et al.*, "O Peer, Where Art Thou? Uncovering Remote Peering Interconnections at IXPs," *IEEE/ACM ToN*, 2021.
[6] Cisco, "White paper," *Cisco Visual Networking Index: Forecast and Trends*, 2017–2022.

[7] (2018) Netflix titus. [Online]. Available: https://netflix.github.io/titus/
[8] T. V. Doan, L. Pajevic, V. Bajpai, and J. Ott, "Tracing the Path to YouTube: A Quantification of Path Lengths and Latencies Toward Content Caches," *IEEE Communications Magazine*, 2019.
[9] A. Araldo *et al.*, "Resource allocation for edge computing with multiple tenant configurations," *ACM/SIGAPP SAC*, 2020.
[10] F. Tütüncüoğlu, A. Ben-Ameur, G. Dán, A. Araldo, and T. Chahed, "Dynamic time-of-use pricing for serverless edge computing with generalized hidden parameter markov decision processes," in *IEEE ICDCS*, 2024, pp. 668–679.
[11] M. Ahmadi *et al.*, "Cache subsidies for an optimal memory for bandwidth tradeoff in the access network," *JSAC*, 2020.
[12] A. Araldo *et al.*, "Caching encrypted content via stochastic cache partitioning," *IEEE/ACM ToN*, 2018.
[13] W. Chu *et al.*, "Joint cache resource allocation and request routing for in-network caching services," *Computer Networks*, 2018.
[14] S. Yang *et al.*, "Online orchestration of collaborative caching for multi-bitrate videos in edge computing," *IEEE TPDS*, 2022.
[15] P. Blasco and D. Gündüz, "Multi-armed bandit optimization of cache content in wireless infostation networks," in *IEEE International Symposium on Information Theory*, 2014.
[16] S. Hoteita *et al.*, "On fair network cache allocation to content providers," *Computer Networks*, 2016.
[17] G. Zheng and V. Friderikos, "Fair cache sharing management for multi-tenant based mobile edge networks," *MobiArch*, 2020.
[18] F. Tütüncüoğlu and G. Dán, "Optimal pricing for service caching and task offloading in edge computing," in *IEEE/IFIP WONS*, 2022.
[19] W. Xiaofei *et al.*, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, 2019.
[20] H. Mao *et al.*, "Resource management with deep RL," *HotNets*, 2016.
[21] Z. Fang *et al.*, "Qos-aware scheduling of heterogeneous servers for inference in deep neural networks," *CIKM*, 2017.
[22] J. Rao *et al.*, "VCONF: a RL approach to VMs auto-configuration," *ACM ICAC*, 2009.
[23] F. Mason, G. Nencioni, and A. Zanella, "Using distributed reinforcement learning for resource orchestration in a network slicing scenario," *IEEE/ACM ToN*, 2022.
[24] J. Yuang *et al.*, "Fast reinforcement learning algorithms for resource allocation in data centers," *IFIP*, 2020.
[25] A. Ben-Ameur, A. Araldo, and T. Chahed, "Cache allocation in multi-tenant edge computing via online reinforcement learning," *IEEE ICC*, 2022.
[26] S. Jošilo and G. Dán, "Joint wireless and edge computing resource management with dynamic network slice selection," *IEEE/ACM ToN*, 2022.
[27] ——, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *IEEE INFOCOM*, 2019.
[28] J. Du *et al.*, "SDN-based resource allocation in edge and cloud computing systems: An evolutionary stackelberg differential game approach," *IEEE/ACM ToN*, 2022.
[29] T. Bahreini *et al.*, "Mechanisms for resource allocation and pricing in mobile edge computing systems," *IEEE TPDS*, 2021.
[30] F. Fossati *et al.*, "Multi-resource allocation for network slicing," *IEEE/ACM ToN*, 2020.
[31] D. Wang, Y. Bai, G. Huang, B. Song, and F. R. Yu, "Cache-aided mec for iot: Resource allocation using deep graph reinforcement learning," *IEEE Internet of Things Journal*, 2023.
[32] S. Yang, J. Liu, F. Zhang, F. Li, X. Chen, and X. Fu, "Caching-enabled computation offloading in multi-region mec network via deep reinforcement learning," *IEEE Internet of Things Journal*, 2022.
[33] M. Elkael *et al.*, "Monkey business: Reinforcement learning meets neighborhood search for virtual network embedding," *Com.Net.*, 2022.
[34] W. Huang *et al.*, "Dimensioning resources of Network Slices for energy-performance trade-off," in *IEEE SCC*, 2022.
[35] D. T. Nguyen, L. B. Le, and V. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Trans. Cloud Comp.*, 2021.
[36] S. Zhang, Y. Liang, J. Ge, M. Xiao, and J. Wu, "Provably efficient resource allocation for edge service entities using hermes," *IEEE/ACM ToN*, 2020.
[37] A. Krause *et al.*, "Submodular function maximization," *Tractability*, 2011.
[38] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *The MIT Press*, 2018.
[39] W. Fedus *et al.*, "Revisiting fundamentals of experience replay," *ICML*, 2020.
[40] S. Natarajan. (2020) Stretched exponential decay function for epsilon greedy algorithm. [Online]. Available: https://medium.com/analytics-vidhya/stretched-exponential-decay-function-for-epsilon-greedy-algorithm
[41] O. Granichin *et al.*, "Simultaneous perturbation stochastic approximation for tracking under unknown but bounded disturbances," *IEEE Trans. Aut. Contr.*, 2015.
[42] B. Shahriari *et al.*, "Taking the human out of the loop : A review of bayesian optimization," *Proceedings of the IEEE*, 2016.
[43] C.-F. Liu *et al.*, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," *IEEE Globecom*, 2017.
[44] X. Lyu *et al.*, "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE JSAC*, 2017.
[45] L. F. Yang *et al.*, "Learning to control in metric space with optimal regret," in *IEEE Annual Allerton Conference on Communication*, 2019.
[46] R. Ortner, "Online regret bounds for markov decision processes with deterministic transitions," *Theoretical Computer Science*, 2010.
[47] O. Dekel, J. Ding, T. Koren, and Y. Peres, "Bandits with switching costs: T 2/3 regret," *ACM Symposium on Theory of computing*, 2014.
[48] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The non-stochastic multiarmed bandit problem," *Society for Industrial and Applied Mathematics*, 2003.
[49] R. Warlop *et al.*, "Fighting boredom in recommender systems with linear reinforcement learning," *Advances in NIPS*, 2018.
[50] P. Alatur *et al.*, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *ACM Sigmetrics*, 2019.
[51] O. Dekel and E. Hazan, "Better rates for any adversarial deterministic MDP," *ICML*, 2013.
[52] G. Goel *et al.*, "Beyond Online Balanced Descent: An Optimal Algorithm for Smoothed Online Optimization," *Advances in NIPS*, 2019.
[53] M. Lin *et al.*, "Online optimization with switching cost," *Performance Evaluation Review*, 2012.
[54] W. W. Cohen and H. Hirsh, "To discount or not to discount in reinforcement learning: A case study comparing r learning and q learning," in *Machine Learning Proceedings*, 1994.
[55] (2022) Amazon cloud front. [Online]. Available: https://aws.amazon.com/fr/cloudfront/
[56] (2023) Netflix open connect. [Online]. Available: https://openconnect.netflix.com/
[57] R. K. Jain *et al.*, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation*, 1998.
[58] OctopusTeam. (2024) Full Netflix Dataset. [Online]. Available: https://www.kaggle.com/datasets/octopusteam/full-netflix-dataset
[59] ——. (2024) Full Amazon Prime Video Dataset. [Online]. Available: https://www.kaggle.com/datasets/octopusteam/full-amazon-prime-dataset
[60] ——. (2024) Full Apple TV Dataset. [Online]. Available: https://www.kaggle.com/datasets/octopusteam/full-apple-tv-dataset
[61] mexwell. (2023) Telecom Shanghai Dataset. [Online]. Available: https://www.kaggle.com/datasets/mexwell/telecom-shanghai-dataset
[62] Sandvine. (2023) THE GLOBAL INTERNET PHENOMENA REPORT JANUARY 2023. [Online]. Available: https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/2023/reports/Sandvine%20GIPR%202023.pdf
[63] Z. Liu, C. Hu, R. Li, T. Xiang, X. Li, J. Yu, and H. Xia, "A privacy-preserving outsourcing computing scheme based on secure trusted environment," *IEEE Trans. Cloud Comp.*, vol. 11, no. 3, pp. 2325–2336, 2022.
[64] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Machine learning*, 1994.
[65] M. Walker, "Convergence of sequences of functions: Some additional notes lecture notes," *Economics 519: Mathematics for Economists, University of Arizona*, 2017.
[66] M. T. Regehr and A. Ayoub, "An elementary proof that Q-learning converges almost surely," 2021.
[67] F. S. Melo, "Convergence of q-learning: A simple proof," *Institute Of Systems and Robotics, Tech. Rep*, 2001.
[68] M. L. Littman and C. Szepesvári, "A generalized reinforcement-learning model: Convergence and applications," in *ICML*, 1996.
[69] T. S. Jaakkola *et al.*, "On the convergence of stochastic iterative dynamic programming algorithms," *Neural Computing*, 1994.
[70] M. G. Bellemare, G. Ostrovski, A. Guez, P. S. Thomas, and R. Munos, "Increasing the action gap: New operators for reinforcement learning," in *AAAI*, 2016.
[71] D. S. Hochbaum, "Lower and Upper Bounds for the Allocation Problem and Other Nonlinear Optimization Problems," *Math. Op. Res.*, 1994.

LIST OF FIGURES

LIST OF TABLES

photos/Ayoub-Ben-Ameur.jpeg

Ayoub Ben-Ameur is an applied researcher at NetMicroscope Inc. (Chicago, IL). He received the national engineering diploma in Telecommunication systems from the higher school of communications of Tunis, Tunisia in 2020. He was awarded a Ph.D. in Computer Science, Data, and Artificial Intelligence from the Institut Polytechnique de Paris - Telecom SudParis, France in 2023 where he worked on data-driven strategies for resource allocation and pricing in edge computing systems. After securing funding from the French embassy in Sweden and from ERASMUS+, he was a visiting researcher at KTH Royal Institute of Technology in Stockholm, Sweden in 2023.

photos/andrea.jpeg

Andrea Araldo is Assoc. Prof. at the Institut Polytechnique de Paris. His research interests are Network Optimization and Smart Mobility. He holds a PhD in Computer Science from Un. Paris Saclay (France - 2016). He was visiting researcher at KTH Royal Institute of Technology (Sweden - 2016) and Postdoc at the Massachusetts Institute of Technology (MIT - US - 2017-2018). In 2018 he was awarded the IEEE ComSoc best paper and in 2022 the French national Young Researcher grant (ANR JCJC).

photos/tijani.jpeg

Tijani Chahed received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University, Turkey, and the Ph.D. and Habilitation a Diriger des Recherches (HDR) degrees in computer science from the University of Versailles and the University of Paris 6, France, respectively. He is currently a Professor with the Networks and Services Department, Telecom SudParis, France. His research interests include resource allocation and quality of service, notably in wireless networks.

photos/gyuri.jpeg

György Dán is professor in teletraffic systems at KTH Royal Institute of Technology, Stockholm, Sweden. He received the M.Sc. degree in computer engineering from the Budapest University of Technology and Economics, Hungary in 1999, the M.Sc. degree in business administration from the Corvinus University of Budapest, Hungary in 2003, and the Ph.D. in Telecommunications from KTH in 2006. He worked as a consultant in the field of access networks, streaming media and videoconferencing 1999-2001. He was a visiting researcher at the Swedish Institute of Computer Science in 2008, a Fulbright research scholar at the Information Trust Institute at University of Illinois Urbana-Champaign in 2012-2013, and an invited professor at EPFL in 2014-2015. His research interests include the design and analysis of content management and computing systems, game theoretical models of networked systems, and cyber-physical system security in power systems.

APPENDIX A
IMPLEMENTATION CONSIDERATIONS

The system described can be implemented under the following assumptions. Similar to Content Delivery Networks (CDNs), each SP runs its own infrastructure over the Internet. As part of this infrastructure, a software *SP edge server* (implemented as a set of microservices) runs on the edge. All requests from users for the content of a specific SP are redirected to the respective SP edge server, e.g., via DNS redirection, which is handled by the NO (it would also be possible that each SP redirects its own requests). All requests of a certain SP go through its edge server. If the object requested is not on the edge, the edge server will forward the request to another server, over the Internet, of its infrastructure. Although the edge servers of all SPs run in the physical edge node owned by the NO, the NO cannot access their data and their processing, as they are protected via state-of-the art memory encryption technologies, e.g., Intel SGX [63, §3.2]. The connection between a user and an SP edge server is encrypted via, we assume for clarity, Transport Layer Security (TLS). The NO can observe and count TLS records (e.g., packets) carrying HTTP messages, either requests or replies with chunks of videos. Such records are denoted by the TLS field `type=application data (23)`. Although the NO cannot observe the identifiers of the objects requested and sent (as they are encrypted into the body of the TLS records), the NO can count the aforementioned records, thus obtaining a sufficiently precise estimation of the requests sent from users to a certain SP, of how many of them are served directly from the SP edge server (hit rate) and of how many necessitated a download from another server outside the edge (miss rate).

One could envisage a collaboration between SPs and the NO, where SPs accept to provide additional information to the NO to help it cache content. While this collaboration could be fruitful and simplify operations, it is not straightforward to realize. First, with current technologies, all traffic is encrypted into TLS sessions, including possible metadata that could guide the NO in caching. Second, the information about what content is consumed by which user is a business asset: user profiling information can be monetized by the SP, via improving service, customizing advertisements (and thus selling advertisements at a higher price) or selling it to third parties. Therefore, it is reasonable to assume SPs are reluctant to share it with NOs, even when this could help optimize caching. For these reasons, in this paper we take a conservative assumption, and we assume no such collaboration exists. This justifies why we resort to a data-driven approach.

APPENDIX B
PROOF OF THE CONVERGENCE THEOREM V-B.1

In what follows, we aim to prove that if the discount factor γ is sufficiently close to 1, then

$$\lim_{k\to\infty} \boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^* \text{ with probability } 1.$$

This means that the system will converge to a discretely optimal state (Section V-B) and will stay in that state with probability 1 as time goes to infinity.

Definition B-.1. *Given a Q-table $Q(\boldsymbol{\theta}, \mathbf{a})$, $\forall(\boldsymbol{\theta}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}_{\boldsymbol{\theta}}$, we say that a sequence of states and actions is induced by Q, if and only if,*

$$\mathbf{a}^{(k)} \in \arg\min_{\mathbf{a}} Q(\boldsymbol{\theta}^{(k)}, \mathbf{a}), \forall k > 0$$

i.e., if and only if it greedily follows Q.

Definition B-.2. *Given a sequence $Q^{(k)}$ of Q-tables , we say that a sequence of state and actions is induced by a sequence of Q-tables $Q^{(k)}$, if and only if,*

$$\mathbf{a}^{(k)} \in \arg\min_{\mathbf{a}} Q^{(k)}(\boldsymbol{\theta}^{(k)}, \mathbf{a}), \forall k > 0$$

i.e., if and only if it greedily follows $Q^{(k)}$.

We decompose the proof as follows: In Section B-A we deeply describe the process of updating the Q-table $Q^{(k)}$. In Section B-B we prove that our Q-table $Q^{(k)}$ converges to the optimal Q-table Q^* with probability 1. In Section B-C we prove that the sequence of actions and states induced by Q^* has an absorbing state that is the discretely optimal state $\hat{\boldsymbol{\theta}}^*$. In Section B-D we prove that the sequence of actions and states induced by our Q-table $Q^{(k)}$ (assuming no more exploration) is also induced by Q^*. In Section B-E we prove that the sequence of states and actions $\{\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}\}$ that we take online (thus including ϵ-greedy exploration) converges with probability 1 to the sequence induced by our Q-table $Q^{(k)}$ if we were not doing any exploration. Finally, we show that $\{\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}\}$, taken online, converges with probability 1 to the sequence induced by Q^*.

Definition B-.3. *We say that $Q^{(k)}$ converges with probability 1 to Q^*, if and only if [64]*

$$\mathbb{P}(\lim_{k\to\infty} |Q^{(k)}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a})| < \epsilon) = 1, \forall \epsilon > 0, \forall(\boldsymbol{\theta}, \mathbf{a})$$

A. Consistency Q-table updates

Observe that in Algorithm 1, we update at every time-slot k the Q-table $N_u = 1 + N_{\text{memory}} + N_{model}$ times. For simplicity of notation, up to now we have only referred to Q-table $Q^{(k)}(\cdot,\cdot)$, but actually the Q-table has changed N_u times in one single iteration of Algorithm 1. In this proof, we need to distinguish all these different versions. Let us denote with $Q^{\{j\}}(\cdot,\cdot)$ the j-th version of the Q-table. In time-slot k, versions $Q^{\{j\}}(\cdot,\cdot)$, for $j = k \cdot N_u, \ldots, (k+1) \cdot N_u - 1$ are created. Updates of lines 6, 14 and 26 of Algorithm 1 can be described in a unified way: when computing version $Q^{\{j+1\}}(\cdot,\cdot)$, a state action pair $(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}})$ is chosen, and we apply an update rule of the following form:

$$\begin{aligned} Q^{\{j+1\}}(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}}) = (1-\alpha^{\{j\}}) \cdot Q^{\{j\}}(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}}) \\ +\alpha^{\{j\}}(C_{\text{nom}}^{\{j\}} + C_{\text{pert}}(\mathbf{a}^{\{j\}}) + \gamma^{\{j\}} \min_{\mathbf{a}\in\mathcal{A}_{\boldsymbol{\theta}'^{\{j\}}}} Q^{\{j\}}(\boldsymbol{\theta}'^{\{j\}}, \mathbf{a})) \end{aligned} \tag{24}$$

where $\boldsymbol{\theta}'^{\{j\}} = \boldsymbol{\theta}^{\{j\}} + \mathbf{a}^{\{j\}}$ and $\alpha^{\{j\}} = \alpha^{(k)}, \gamma^{\{j\}} = \gamma^{(k)}$, for any $j = k \cdot N_u, \ldots, (k+1) \cdot N_u - 1$. As for the value of $(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}}, C_{\text{nom}}^{\{j\}})$, it depends on whether the j-th update is

obtained using an observed sample (Line 6 of Algorithm 1), experience replay (Line 14) or the model (Line 26):

$$(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}}, C_{\text{nom}}^{\{j\}}) = \begin{cases} (\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}, C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega)) \\ \text{if } j = k \cdot N_u \\ \text{(use observed nominal cost)} \\ \\ (\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}, C_{\text{nom}}^{\text{rd}}) \in \mathcal{M}^{(k)} \\ \text{if } j = k \cdot N_u + j', j' = 1, \dots, N_{\text{memory}} \\ \text{(use nominal cost from memory)} \\ \\ (\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}, \hat{C}_{\text{nom}}^{(k)}(\boldsymbol{\theta}^{\text{rd}})) \\ \text{if } j = k \cdot N_u + N_{\text{memory}} + j', j' = 1, \dots, N_{\text{model}} \\ \text{for randomly chosen} \\ \boldsymbol{\theta}^{\text{rd}} \in \mathcal{S}, \mathbf{a}^{\text{rd}} \in \mathcal{A}_{\boldsymbol{\theta}^{\text{rd}}} \\ \text{(use estimation of nominal cost from the model)} \end{cases} \tag{24bis}$$

For $j = 1, 2, \dots,$ we also define function $\hat{C}_{\text{nom}}^{\{j\}}(\cdot) : \mathcal{S} \to \mathbb{R}$ as

$$\hat{C}_{\text{nom}}^{\{0\}}(\boldsymbol{\theta}) = 0, \forall \boldsymbol{\theta} \in \mathcal{S}$$
$$\hat{C}_{\text{nom}}^{\{j\}}(\boldsymbol{\theta}) \triangleq \begin{cases} C_{\text{nom}}^{\{j\}} & \text{if } \boldsymbol{\theta} = \boldsymbol{\theta}^{\{j\}} \text{(see (24bis))} \\ \hat{C}_{\text{nom}}^{\{j-1\}}(\boldsymbol{\theta}) & \text{otherwise} \end{cases} \tag{24tris}$$

Definition (24tris) allows rewriting update (24) as follows:

$$Q^{\{j+1\}}(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}}) = (1 - \alpha^{\{j\}}) \cdot Q^{\{j\}}(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}}) + \alpha^{\{j\}}(\hat{C}_{\text{nom}}^{\{j\}} + C_{\text{pert}}(\mathbf{a}^{\{j\}}) + \gamma^{\{j\}} \min_{\mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}'^{\{j\}}}} Q^{\{j\}}(\boldsymbol{\theta}'^{\{j\}}, \mathbf{a})) \tag{24quadris}$$

In practice, we have just replaced $C_{\text{nom}}^{\{j\}}$ with $\hat{C}_{\text{nom}}^{\{j\}}$. Thanks to this minor change, we get rid of sequence of numbers $\{C_{\text{nom}}^{\{j\}}\}_j$, replacing it with sequence of functions $\{\boldsymbol{\theta} \to \hat{C}_{\text{nom}}^{\{j\}}(\boldsymbol{\theta})\}_j$, which is important as the proof of Theorem B-B.3 (in particular (35)) requires a sequence of functions. We call (24quadris) the functional form of the Q-table update.

Observe that at step j, the Q-table is updated only in correspondence to pairs $(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}})$, while it remains unchanged in all other values:

$$Q^{\{j+1\}}(\boldsymbol{\theta}, \mathbf{a}) = Q^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) \forall (\boldsymbol{\theta}, \mathbf{a}) \neq (\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}})$$

Similarly, by construction of (24tris), for any $\boldsymbol{\theta} \in \mathcal{S}$, the value $\hat{C}_{\text{nom}}^{\{j\}}(\boldsymbol{\theta})$ only changes at step j for which $\boldsymbol{\theta} = \boldsymbol{\theta}^{\{j\}}$. In all the other steps, the value is inherited from the previous steps. The following theorem characterizes function $\hat{C}_{\text{nom}}^{\{j\}}(\cdot)$ used in the functional form of the Q-table updates.

Theorem B-A.1. *Function $\hat{C}_{nom}^{\{j\}}(\cdot)$ converges uniformly in expectation to the nominal cost, i.e.,*

$$\lim_{j \to \infty}^{u} \mathbb{E}\left[\hat{C}_{nom}^{\{j\}}(\cdot) | \mathcal{F}^{\{j\}}\right] = \mathbb{E}C_{nom}(\cdot). \tag{25}$$

where $\lim^u$ has the same meaning as in Theorem IV-B.1 and $\mathcal{F}^{\{j\}} = \{Q^{\{j\}}, Q^{\{j-1\}}, ...\}$ stands for the past at step j.

Proof. We first consider any $\boldsymbol{\theta} \in \mathcal{S}$ and prove pointwise convergence, i.e. that

$$\lim_{j \to \infty} \mathbb{E}\left[\hat{C}_{\text{nom}}^{\{j\}}(\boldsymbol{\theta}) | \mathcal{F}^{\{j\}}\right] = \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}) \tag{26}$$

To this aim, exploiting the construction of $\hat{C}_{\text{nom}}^{\{j\}}(\cdot)$, it suffices to show that

$$\lim_{z \to \infty} \mathbb{E}\left[\hat{C}_{\text{nom}}^{\{j_z\}}(\boldsymbol{\theta}) | \mathcal{F}^{\{j_z\}}\right] = \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}), \tag{27}$$

where $j_1, j_2, \dots$ is the subsequence of indices js such that $\boldsymbol{\theta}^{\{j_z\}} = \boldsymbol{\theta}$. We denote with $k\{j\}$ the timeslot in which version $Q^{\{j\}}(\cdot, \cdot)$ of the Q-table is calculated, i.e., $k\{j\} = k$ for $j = k \cdot N_u, \dots, (k+1) \cdot N_u - 1$. Indices $\{j_z\}_{z \in \mathbb{N}}$ can be divided in three subsequences of indices:

1) Indices $\{j_{z_w}\}_{w \in \mathbb{N}}$ such that $j_{z_w} = k\{j_{z_w}\} \cdot N_u$. In this case, thanks to (24bis),
$$\mathbb{E}\left[\hat{C}_{\text{nom}}^{\{j_{z_w}\}}(\boldsymbol{\theta}) | \mathcal{F}^{\{j_z\}}\right] = \mathbb{E}\left[C_{\text{nom}}(\boldsymbol{\theta}, \omega^{(k\{j_{z_w}\})}) | \mathcal{F}^{\{j_{z_w}\}}\right] = \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$$
2) Indices $\{j_{z_w}\}_{w \in \mathbb{N}}$ such that $j_{z_w} = k\{j_{z_w}\} \cdot N_u + j', j' = 1, \dots, N_{\text{memory}}$. In this case, thanks to (24tris), $\hat{C}_{\text{nom}}^{\{j_{z_w}\}}(\boldsymbol{\theta})$ is a past observation of nominal cost when state $\boldsymbol{\theta}$ was visited. By construction, its expected value is $\mathbb{E}\left[\hat{C}_{\text{nom}}^{\{j_{z_w}\}}(\boldsymbol{\theta}) | \mathcal{F}^{\{j_z\}}\right] = \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$.
3) Indices $\{j_{z_w}\}_{w \in \mathbb{N}}$ such that $j_{z_w} = k\{j_{z_w}\} \cdot N_u + N_{\text{memory}} + j', j' = 1, \dots, N_{\text{model}}$. In this case, thanks to (24tris), $C_{\text{nom}}^{\{j_{z_w}\}}(\boldsymbol{\theta})$ is obtained via the model explained in Section IV-B. By exploiting Theorem IV-B.1, we have that $\lim_{w \to \infty} \mathbb{E}\left[C_{\text{nom}}^{\{j_{z_w}\}}(\boldsymbol{\theta})\right] = \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$.

Since sequence $\mathbb{E}\left[\hat{C}_{\text{nom}}^{\{j_z\}}(\boldsymbol{\theta}) | \mathcal{F}^{\{j_z\}}\right]$ is the union of the three subsequences above, each of which converges to $\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$, we obtain the theorem.

Since pointwise convergence 26 holds for all $\boldsymbol{\theta} \in \mathcal{S}$ and $\mathcal{S}$ is finite, then [65, Proposition 1] convergence is also uniform. □

B. Convergence of the Q-table

In this section, we will prove that our Q-table $Q^{(k)}$, updated following (24bis), converges to the optimal Q-table Q^* with probability 1. As we combine Q-Learning with a model that approximates the expected nominal cost $\mathbb{E}_\omega[C_{\text{nom}}(\boldsymbol{\theta}, \omega)]$ (Section IV-B) and with other enhancements mentioned in Section IV-C, we cannot simply rely on the property of convergence of classical Q-learning (Theorem 2 of [66] or [67]). It is worth noting that many works claim that model-based RL in all its forms converges [68], however no explicit proof is provided for the specific form of combining Q-table updates with a model of the reward (cost in our case).

In what follows, we will prove that the process defined by (24bis) converges to the optimal Q-table Q^* with probability 1. We start with two general results Lemma B-B.1 and Lemma B-B.2, which we obtain by extending a previously known result.

Lemma B-B.1. *Consider a random iterative process* $\delta^{\{j\}} : \mathcal{X} \to \mathbb{R}^n$ *defined as*

$$\delta^{\{j+1\}}(x) = (1-\alpha^{\{j\}}) \cdot \delta^{\{j\}}(x) + \alpha^{\{j\}} \cdot F^{\{j\}}(x) \quad (28)$$

where $F^{\{j\}} : \mathcal{X} \to \mathbb{R}^n$ *is a random function, at each step* j*. Process* $\delta^{\{j\}}$ *converges with probability* 1 *to* 0*, i.e.,* $\mathbb{P}\left[\lim_{j\to\infty} \delta^{\{j\}}(x) = 0\right] = 1, \forall x \in \mathcal{X}$*, under the following assumptions:*

1) $\mathcal{X}$ *is finite*
2) $0 \leq \alpha^{\{j\}} \leq 1, \sum_k \alpha^{\{j\}} = \infty$ *and* $\sum_k (\alpha^{\{j\}})^2 < \infty$
3) $\exists 0 < \nu < 1$:
$$\|\mathbb{E}[F^{\{j\}}(x) - \eta^{\{j\}}(x) | \mathcal{F}^{\{j\}}]\|_\infty \leq \nu \|\delta^{\{j\}}\|_\infty,$$
where $\eta^{\{j\}}(\cdot)$ *is a sequence of functions such that* $\lim^u_{j\to\infty} \eta^{\{j\}}(\cdot) = 0$.
4) $\exists M > 0, \textbf{\textit{var}}[F^{\{j\}}(x) | \mathcal{F}^{\{j\}}] \leq M(1 + \|\delta^{\{j\}}\|^2_\infty)$

Here $\mathcal{F}^{\{j\}} = \{\delta^{\{j\}}, \delta^{\{j-1\}}, .., F^{\{j-1\}}, .., \alpha^{\{j-1\}}, ..\}$ *stands for the past at step* j*.*

Proof. The original version of this theorem is [69, Theorem 1]. The only difference is assumption 3, which in [69] is

$$\exists 0 < \nu < 1, \|\mathbb{E}[F^{\{j\}}(x) | \mathcal{F}^{\{j\}}]\|_\infty \leq \nu \|\delta^{\{j\}}\|_\infty.$$

We need to add the additional term $\mathbb{E}[\eta^{\{j\}}(x) | \mathcal{F}^{\{j\}}]$ to account for the error of our model in approximating the expected value of the cost. Let us define another random process in the following way:

$$\begin{aligned} F_\prime^{\{j\}}(x) &= F^{\{j\}}(x) - \eta^{\{j\}}(x) \\ \delta_\prime^{\{j+1\}}(x) &= (1-\alpha^{\{j\}}) \cdot \delta_\prime^{\{j\}}(x) + \alpha^{\{j\}} \cdot F_\prime^{\{j\}}(x) \end{aligned} \quad (29)$$

Process $\delta_\prime^{\{j\}}(x)$ respects the assumptions of the original [69, Theorem 1] and thus

$$\lim_{j\to\infty} \delta_\prime^{\{j\}}(x) = 0, \forall x \in \mathcal{X}, \text{ with probability 1.} \quad (30)$$

Let us now define an additional random process

$$\delta^{\{j\}}_{\text{diff}}(x) = \delta_\prime^{\{j\}}(x) - \delta^{\{j\}}(x) \quad (31)$$

and observe that

$$\begin{aligned} \delta^{\{j+1\}}_{\text{diff}}(x) &= \delta_\prime^{\{j+1\}}(x) - \delta^{\{j+1\}}(x) \\ &= \left[(1-\alpha^{\{j\}}) \cdot \delta_\prime^{\{j\}}(x) + \alpha^{\{j\}} \cdot F_\prime^{\{j\}}(x)\right] \\ &\quad - \left[(1-\alpha^{\{j\}}) \cdot \delta^{\{j\}}(x) + \alpha^{\{j\}} \cdot F^{\{j\}}(x)\right] \\ &= (1-\alpha^{\{j\}}) \cdot (\delta_\prime^{\{j\}}(x) - \delta^{\{j\}}(x)) \\ &\quad {}^{+}\alpha^{\{j\}} \cdot (-\eta^{\{j\}}(x)) \\ &= (1-\alpha^{\{j\}}) \cdot \delta^{\{j\}}_{\text{diff}}(x) + \alpha^{\{j\}} \cdot (-\eta^{\{j\}}(x)) \end{aligned}$$

The iterative process above respects the assumptions of [69, Lemma 1], which states that

$$\lim_{j\to\infty} \delta^{\{j\}}_{\text{diff}}(x) = 0, \forall x \in \mathcal{X}, \text{ with probability 1.} \quad (32)$$

Thanks to (30) and (31), process $\delta^{\{j\}}(x)$ must converge to 0 with probability 1. □

Lemma B-B.2. *Consider a random iterative process* $\delta^{\{j\}} : \mathcal{X} \to \mathbb{R}^n$ *such that, for each* $x \in \mathcal{X}$*, there is an infinite subsequence of indices* $\mathcal{I}_x = \{j_1, j_2, ..., j_z, ...\}$ *such that*

$$\delta^{\{j_z+1\}}(x) = (1-\alpha^{\{j_z\}}) \cdot \delta^{\{j_z\}}(x) + \alpha^{\{j_z\}} \cdot F^{\{j_z\}}(x), \forall j_z \in \mathcal{I}_x$$

and

$$\delta^{\{j_z+1\}}(x) = \delta^{\{j_z\}}(x), \forall j_z \in \mathbb{N} \setminus \mathcal{I}_x.^5$$

Assume that the same assumptions of Lemma B-B.1 hold:

1) $\mathcal{X}$ *is finite*
2) $0 \leq \alpha^{\{j\}} \leq 1, \sum_k \alpha^{\{j\}} = \infty$ *and* $\sum_k (\alpha^{\{j\}})^2 < \infty$
3) $\exists 0 < \nu < 1$:
$$\|\mathbb{E}[F^{\{j\}}(x) - \eta^{\{j\}}(x) | \mathcal{F}^{\{j\}}]\|_\infty \leq \nu \|\delta^{\{j\}}\|_\infty,$$
where $\eta^{\{j\}}(\cdot)$ *is a sequence of functions such that* $\lim^u_{j\to\infty} \eta^{\{j\}}(\cdot) = 0$.
4) $\exists M > 0, \textbf{\textit{var}}[F^{\{j\}}(x) | \mathcal{F}^{\{j\}}] \leq M(1 + \|\delta^{\{j\}}\|^2_\infty)$

Then, process $\delta^{\{j\}}$ *converges with probability* 1 *to* 0*, i.e.,* $\mathbb{P}\left[\lim_{j\to\infty} \delta^{\{j\}}(x) = 0\right] = 1, \forall\, x \in \mathcal{X}$*.*

Proof. Let us fix any $x \in \mathcal{X}$. By applying Lemma B-B.1 on the sequence $\delta^{\{j_1\}}(x), \delta^{\{j_2\}}(x), ...$ (which is subsequence of $\delta^{\{1\}}(x), \delta^{\{2\}}(x), ...$) we obtain that,

$$\lim_{z\to\infty} \delta^{\{j_z\}}(x) = 0, \forall x \in \mathcal{X}, \text{ with probability 1.}$$

Observe that by construction, for any $z \in \mathbb{N}$,

$$\delta^{\{j\}}(x) = \delta^{\{j_z+1\}}(x), \text{ for } j = j_z + 1, j_z + 2, ..., j_{z+1}.$$

This implies that,

$$\lim_{j\to\infty} \delta^{\{j\}}(x) = 0, \forall x \in \mathcal{X}, \text{ with probability 1.}$$

We have thus proved point-wise convergence. Thanks to [65, Proposition 1], since $\mathcal{X}$ is finite, this also implies uniform convergence. □

Theorem B-B.3. *If* $Q^{\{j\}}$ *is updated by update* (24quadris)*, then* $Q^{\{j\}}$ *converges to* Q^* *with probability* 1*.*

Proof. We will prove the theorem using Lemma B-B.1. We define $\mathcal{X} = \mathcal{S} \times \mathcal{A}$ and

$$\begin{aligned} F^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) &= \hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) + C_{\text{pert}}(\mathbf{a}) \\ &\quad + \gamma^{\{j\}} \min_{\mathbf{a}' \in \mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} Q^{\{j\}}(\boldsymbol{\theta} + \mathbf{a}, \mathbf{a}') - Q^*(\boldsymbol{\theta}, \mathbf{a}) \end{aligned} \quad (33)$$

Observe that, by fixing the past $\mathcal{F}^{\{j\}}$, i.e., all the observed rewards, actions, states and extractions from the memory and from the model, up to before update j, table $Q^{\{j\}}$ is univocally determined. The only stochastic term in (33) is thus $\hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta})$, while all the others are deterministic. Let us define $\delta^{\{j\}}(\boldsymbol{\theta}, \mathbf{a})$ as follows:

$$\delta^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) = Q^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a}), \forall (\boldsymbol{\theta}, \mathbf{a}) \in \mathcal{X} \quad (34)$$

[5] Observe that the subsequence of indices $\mathcal{I}_x$ changes for every considered x.

If $(\boldsymbol{\theta},\mathbf{a})=(\boldsymbol{\theta}^{\{j\}},\mathbf{a}^{\{j\}})$, we obtain

$$\begin{aligned}
\delta^{\{j+1\}}(\boldsymbol{\theta},\mathbf{a}) &= Q^{\{j+1\}}(\boldsymbol{\theta},\mathbf{a}) - Q^*(\boldsymbol{\theta},\mathbf{a}) \\
&\overset{(24\text{quadris})}{=} (1-\alpha^{\{j\}})\cdot Q^{\{j\}}(\boldsymbol{\theta},\mathbf{a}) \\
&\quad + \alpha^{\{j\}}\Bigg(\hat{C}^{\{j\}}_{\text{nom}} + C_{\text{pert}}(\mathbf{a}) \\
&\quad + \gamma^{\{j\}} \min_{\mathbf{a}'\in\mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} Q^{\{j\}}(\boldsymbol{\theta}+\mathbf{a},\mathbf{a}')\Bigg) \\
&\quad - Q^*(\boldsymbol{\theta},\mathbf{a}) + \alpha^{\{j\}}Q^*(\boldsymbol{\theta},\mathbf{a}) - \alpha^{\{j\}}Q^*(\boldsymbol{\theta},\mathbf{a}) \\
&= (1-\alpha^{\{j\}})\cdot\Bigg(Q^{\{j\}}(\boldsymbol{\theta},\mathbf{a}) - Q^*(\boldsymbol{\theta},\mathbf{a})\Bigg) \\
&\quad + \alpha^{\{j\}}\cdot\Bigg(\hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) + C_{\text{pert}}(\mathbf{a}) \\
&\quad + \gamma^{\{j\}} \min_{\mathbf{a}'\in\mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} Q^{\{j\}}(\boldsymbol{\theta}+\mathbf{a},\mathbf{a}') - Q^*(\boldsymbol{\theta},\mathbf{a})\Bigg) \\
&= (1-\alpha^{\{j\}})\cdot\delta^{\{j\}}(\boldsymbol{\theta},\mathbf{a}) + \alpha^{\{j\}}\cdot F^{\{j\}}(\boldsymbol{\theta},\mathbf{a})
\end{aligned} \tag{35}$$

Instead, for $(\boldsymbol{\theta},\mathbf{a})\neq(\boldsymbol{\theta}^{\{j\}},\mathbf{a}^{\{j\}})$,

$$\begin{aligned}
\delta^{\{j+1\}}(\boldsymbol{\theta},\mathbf{a}) &= Q^{\{j+1\}}(\boldsymbol{\theta},\mathbf{a}) - Q^*(\boldsymbol{\theta},\mathbf{a}) \\
&\overset{(24\text{tris})}{=} Q^{\{j\}}(\boldsymbol{\theta},\mathbf{a}) - Q^*(\boldsymbol{\theta},\mathbf{a}) \\
&= \delta^{\{j\}}(\boldsymbol{\theta},\mathbf{a})
\end{aligned} \tag{35bis}$$

Observe that for given past $\mathcal{F}^{\{j\}}$, $\delta^{\{j\}}(\boldsymbol{\theta},\mathbf{a})$ is deterministic. Since the exploration never ends (ϵ is always greater than 0), state-action pair $(\boldsymbol{\theta},\mathbf{a})$ is visited infinitely many times. Therefore, there is an infinite subsequences of indices for which (35) holds instead of (35bis). We are thus in the case of Lemma B-B.2.

The first condition of Lemma B-B.2 holds by definition of the state and action spaces. Moreover, the learning rate scheduling (Section IV-C1) obeys the second condition of Lemma B-B.2. The last condition holds because we define the cost function to be bounded.[6] This means that we only have to show that the third condition of Lemma B-B.2 holds to prove convergence of $Q^{\{j\}}$ to Q^*.

The optimal Q-table is a fixed point of a contraction operator $\mathbf{H}$ (see Section 1 of [70]), defined for function $m:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ as:

$$(\mathbf{H}m)(\boldsymbol{\theta},\mathbf{a}) = \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}) + C_{\text{pert}}(\mathbf{a}) + \nu\cdot\min_{\mathbf{a}'\in\mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} m(\boldsymbol{\theta}+\mathbf{a},\mathbf{a}') \tag{36}$$

This operator is a contraction in the sup-norm (Section 1 of [67]), i.e.,

$$\begin{aligned}
\|\mathbf{H}m_1 - \mathbf{H}m_2\|_\infty &\leq \nu\cdot\|m_1-m_2\|_\infty \\
&= \nu\cdot\sup_{(\boldsymbol{\theta},\mathbf{a})\in\mathcal{S}\times\mathcal{A}}|m_1(\boldsymbol{\theta},\mathbf{a}) - m_2(\boldsymbol{\theta},\mathbf{a})|
\end{aligned} \tag{37}$$

We now prove that the third condition of Lemma B-B.2 holds for our $F^{\{j\}}(\boldsymbol{\theta},\mathbf{a}) - \eta^{\{j\}}(\boldsymbol{\theta})$, where $F^{\{j\}}(.,.)$ is defined as in (33) and $\eta^{\{j\}}(\boldsymbol{\theta}):\mathcal{S}\to\mathbb{R}$ is defined as $\eta^{\{j\}}(\boldsymbol{\theta}) = \hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$. Theorem B-A.1 shows that $\lim^u_{j\to\infty}\eta^{\{j\}}(\cdot)=0$. Therefore, via (33):

$$\begin{aligned}
&\forall\boldsymbol{\theta}\in\mathcal{S},\mathbf{a}\in\mathcal{A}_{\boldsymbol{\theta}}, F^{\{j\}}(\boldsymbol{\theta},\mathbf{a}) - \eta^{\{j\}}(\boldsymbol{\theta}) \\
&= \hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) + C_{\text{pert}}(\mathbf{a}) \\
&\quad + \gamma^{\{j\}}\cdot\min_{\mathbf{a}'\in\mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} Q^{\{j\}}(\boldsymbol{\theta}+\mathbf{a},\mathbf{a}') \\
&\quad - Q^*(\boldsymbol{\theta},\mathbf{a}) - \Bigg(\hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})\Bigg) \\
&= \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}) + C_{\text{pert}}(\mathbf{a}) \\
&\quad + \gamma^{\{j\}}\cdot\min_{\mathbf{a}'\in\mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} Q^{\{j\}}(\boldsymbol{\theta}+\mathbf{a},\mathbf{a}') - Q^*(\boldsymbol{\theta},\mathbf{a}) \\
&\overset{(36)}{=} \mathbf{H}Q^{\{j\}}(\boldsymbol{\theta},\mathbf{a}) - Q^*(\boldsymbol{\theta},\mathbf{a}) \\
&\overset{(\text{Since } Q^*=\mathbf{H}Q^*)}{=} \mathbf{H}Q^{\{j\}}(\boldsymbol{\theta},\mathbf{a}) - \mathbf{H}Q^*(\boldsymbol{\theta},\mathbf{a}).
\end{aligned}$$

In the norm-sup, using (37), we obtain:

$$\begin{aligned}
\|F^{\{j\}}(\cdot,\cdot) - \eta^{\{j\}}(\boldsymbol{\theta})\|_\infty &\leq \gamma^{\{j\}}\|Q^{\{j\}}(\cdot,\cdot) - Q^*(\cdot,\cdot)\|_\infty \\
&= \gamma^{\{j\}}\|\delta^{\{j\}}(\cdot,\cdot)\|_\infty
\end{aligned}$$

Applying expectation given past $\mathcal{F}^{\{j\}}$ (and considering that $\delta^{\{j\}}(\cdot,\cdot)$ is deterministic, as we wrote right after (35bis)), we obtain

$$\mathbb{E}\left[\|F^{\{j\}}(\cdot,\cdot) - \eta^{\{j\}}(\cdot)\|_\infty|\mathcal{F}^{\{j\}}\right] \leq \gamma^{\{j\}}\|\delta^{\{j\}}(\cdot,\cdot)\|_\infty$$

which proves that the third condition in Lemma B-B.1 holds. Then, by Lemma B-B.2, $\delta^{\{j\}}$ converges to 0 with probability 1, which implies, via (34), that $Q^{\{j\}}$ converges to Q^* with probability 1. □

C. Absorbing state

In this section, we will prove that the sequence of actions and states induced by Q^* has an absorbing state that is the discretely optimal state $\hat{\boldsymbol{\theta}}^*$. In what follows, we will use the concept of sequences defined as follows:

Definition B-C.1. *A sequence $s=\{\boldsymbol{\theta}^{(k)},\mathbf{a}^{(k)}\}_k$ is a sequence of states and actions such that*

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \mathbf{a}^{(k)}$$

Let us denote with $C^\gamma_{\text{cum}}(s)$ the cumulative discounted reward (11) of any sequence s.

Definition B-C.2. *Sequence $s'=\{\boldsymbol{\theta}'^{(k)},\mathbf{a}'^{(k)}\}_k$ is optimal if, for any other sequence s'' having the same initial state as s^*, we have $C^\gamma_{cum}(s'')\geq C^\gamma_{cum}(s')$.*

Definition B-C.3. *Q-table Q is optimal if, for any initial state, any sequence s'' induced by Q, is an optimal sequence.*

We will use the notation $Q^{(k)}$ to refer to the Q-table of Algorithm 1. We will refer to the following sequences:

- s: state and action sequence induced by sequence $Q^{(k)}$ of Q-tables obtained with our Algorithm 1. We call it "offline sequence".
- s_ϵ: sequence induced by the online policy: such a sequence follows $Q^{(k)}$ with probability $1-\epsilon$ and takes a random action with probability ϵ. This is the sequence that comes out of the actions chosen in lines 3 and 4 of Algorithm 1. We call it "online sequence".

[6] For instance, one could consider that the cost measured at each time-slot cannot exceed the backhaul link capacity.

- s^*: sequence induced by the optimal Q-table Q^*.

It is worth emphasizing that when applying Algorithm 1, we do not traverse sequence s, as we do not take actions induced by $Q^{(k)}$. Indeed, we explore from time to time. In this sense, s is a theoretical sequence, that we use as a reference in our proofs, but that we never follow in reality. What we really follow is s_ϵ.

Lemma B-C.4. *Any sequence s^* induced by Q^* has an absorbing state, i.e.,*

$$\exists \boldsymbol{\theta}_{abs} \in \mathcal{S}, k' > 0 : \boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}_{abs}, \quad k \geq k'$$

Proof. Suppose by contradiction that $s^* = \{\boldsymbol{\theta}'^{(k)}, \mathbf{a}'^{(k)}\}_k$ does not have an absorbing state. If that were the case, we could construct a modified version s'' of s^* as follows. We take the best of the allocations visited, i.e., $\boldsymbol{\theta}_{\text{best}} \in \arg\min_{k=0}^{\infty} \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}^{(k)})$. Suppose k_1 is the first timeslot in which such allocation is visited. Sequence $s'' = \{\boldsymbol{\theta}''^{(k)}, \mathbf{a}''^{(k)}\}_k$ is as follows:

$$\boldsymbol{\theta}''^{(k)} = \begin{cases} \boldsymbol{\theta}'^{(k)} & \text{if } k \leq k_1 \\ \boldsymbol{\theta}_{\text{best}} & \text{otherwise} \end{cases} \tag{38}$$

$$\mathbf{a}''^{(k)} = \begin{cases} \mathbf{a}'^{(k)} & \text{if } k \leq k_1 \\ \mathbf{0} \text{ (null action)} & \text{otherwise} \end{cases} \tag{39}$$

The difference of cumulative discounted cost (11) induced by the two sequences s^* and s'' is

$$\begin{aligned}
&C^{\gamma}_{\text{cum}}(s^*) - C^{\gamma}_{\text{cum}}(s'') \\
&= \lim_{T\to\infty} \mathbb{E}\Bigg[\sum_{k=k_1+1}^{T} \gamma^{(k)} \Big(C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}'^{(k)}) \\
&\quad - C_{\text{nom}}(\boldsymbol{\theta}''^{(k_1)}, \omega) \Big) \Bigg] \\
&= \lim_{T\to\infty} \Bigg[\sum_{k=k_1+1}^{T} \gamma^{(k)} \Big(\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{best}}, \omega) \\
&\quad + C_{\text{pert}}(\mathbf{a}'^{(k)}) \Big) \Bigg]
\end{aligned} \tag{40}$$

For any k, by construction of $\boldsymbol{\theta}_{\text{best}}$, we have

$$\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{best}}, \omega) \geq 0.$$

Moreover, at least one action $\mathbf{a}'^{(k)}$ is non-null, as we have assumed that s^* does not have any absorbing state. Therefore, (40) is positive, which is absurd as it violates Definitions B-C.2 and B-C.3. □

Lemma B-C.5. *If discount factor γ is sufficiently close to* 1*, the absorbing state of sequence s^* is a discretely optimal allocation* (19)*.*

Proof. Let us define an undirected graph $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ where each node $\boldsymbol{\theta} \in \mathcal{S}$ is a state and each edge $\mathbf{a} \in \mathcal{A}$ is an action. Such an edge connects state $\boldsymbol{\theta}$ with state $\boldsymbol{\theta} + \mathbf{a}$ and has weight $\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}, \omega) + C_{\text{pert}}(\mathbf{a})$. Let us denote with $s(\boldsymbol{\theta}, \boldsymbol{\theta'})$ shortest path on such a graph between nodes $\boldsymbol{\theta}$ and $\boldsymbol{\theta'}$, where the cost of the path is the sum of the cost on the arcs. If such a path is $\boldsymbol{\theta} = \boldsymbol{\theta}^{[0]} \xrightarrow{a^{[0]}} \boldsymbol{\theta}^{[1]} \dots \xrightarrow{a^{[n-1]}} \boldsymbol{\theta}^{[n]} = \boldsymbol{\theta'}$, the discounted cost accumulated over this path is:

$$C^{\gamma}_{\text{cum}}(s(\boldsymbol{\theta}, \boldsymbol{\theta'})) = \sum_{j=0}^{n-1} \gamma^j \cdot \Big(\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}^{[j]}, \omega) + C_{\text{pert}}(\mathbf{a}^{[j]}) \Big) \tag{41}$$

Let us define:

$$M \triangleq \max_{\boldsymbol{\theta}, \boldsymbol{\theta'} \in \mathcal{S}} C^{\gamma}_{\text{cum}}(s(\boldsymbol{\theta}, \boldsymbol{\theta'})) \tag{42}$$

Let us take a discretely optimal state $\hat{\boldsymbol{\theta}}^*$. Thanks to Lemma B-C.4, we know that s^* goes to an absorbing state $\boldsymbol{\theta}_{\text{abs}}$ at a certain timeslot k' and does change state anymore. Let us suppose by contradiction that $\boldsymbol{\theta}_{\text{abs}}$ is not discretely optimal and define quantity

$$\delta C = \mathbb{E}C(\boldsymbol{\theta}_{\text{abs}}, \omega) - \mathbb{E}C(\hat{\boldsymbol{\theta}}^*, \omega). \tag{43}$$

By construction, $\delta C > 0$, otherwise $\boldsymbol{\theta}_{\text{abs}}$ would be the discretely optimal.

Let us construct another sequence s'' such that it is the same as s^* up to time slot k'. Then, while s^* stays in $\boldsymbol{\theta}_{\text{abs}}$, s'' takes non-null actions and follows the shortest path $s(\boldsymbol{\theta}_{\text{abs}}, \hat{\boldsymbol{\theta}}^*)$ and then it stays in $\hat{\boldsymbol{\theta}}^*$ and does not change state anymore. Suppose that n is the length of such a shortest path. Let us compute the difference between the cumulative discounted cost of $s^* = \{\boldsymbol{\theta}'^{(k)}, \mathbf{a}'^{(k)}\}$ and $s'' = \{\boldsymbol{\theta}''^{(k)}, \mathbf{a}''^{(k)}\}$:

$$\begin{aligned}
&C^{\gamma}_{\text{cum}}(s'') - C^{\gamma}_{\text{cum}}(s^*) \\
&= \mathbb{E}\Bigg[\sum_{k=0}^{\infty} \gamma^{(k)} \cdot \Big(C_{\text{nom}}(\boldsymbol{\theta}''^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}''^{(k)}) \Big) \\
&\quad - \gamma^{(k)} \cdot \Big(C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}'^{(k)}) \Big) \Bigg] \\
&= \mathbb{E}\Bigg[\sum_{k=0}^{k'+n-1} \gamma^{(k)} \cdot \Big(C_{\text{nom}}(\boldsymbol{\theta}''^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}''^{(k)}) \\
&\quad - C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) - C_{\text{pert}}(\mathbf{a}'^{(k)}) \Big) \\
&\quad + \sum_{k=k'+n}^{\infty} \gamma^{(k)} \cdot \Big(C_{\text{nom}}(\boldsymbol{\theta}''^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}''^{(k)}) \\
&\quad - C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) - C_{\text{pert}}(\mathbf{a}'^{(k)}) \Big) \Bigg] \\
&= \gamma^{(k')} \cdot C^{\gamma}_{\text{cum}}(s(\boldsymbol{\theta}_{\text{abs}}, \hat{\boldsymbol{\theta}}^*)) + \sum_{k=k'+n}^{\infty} \gamma^{(k)} \cdot \mathbb{E}C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega) \\
&\quad - \sum_{k=k'}^{\infty} \gamma^{(k)} \cdot \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{abs}}, \omega) \\
&= \gamma^{(k')} \cdot C^{\gamma}_{\text{cum}}(s(\boldsymbol{\theta}_{\text{abs}}, \hat{\boldsymbol{\theta}}^*)) - \sum_{k=k'}^{k'+n-1} \gamma^{(k)} \cdot \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{abs}}, \omega) \\
&\quad + \sum_{k=k'+n}^{\infty} \gamma^{(k)} \cdot \Big(\mathbb{E}C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{abs}}, \omega) \Big)
\end{aligned} \tag{44}$$

Via (43) and elementary calculus (for the summation of truncated geometric series), we obtain:

$$\sum_{k=k'+n}^{\infty} \gamma^{(k)} \cdot \Big(\mathbb{E}C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{abs}}, \omega)\Big)$$
$$= -\delta C \cdot \underbrace{\sum_{k=k'+n}^{\infty} \gamma^{(k)}}_{\text{trunc. geom. series}} = -\delta C \cdot \frac{\gamma^{k'+n}}{1-\gamma}. \tag{45}$$

By replacing (42) and (45) into (44), we get:

$$C^{\gamma}_{\text{cum}}(s'') - C^{\gamma}_{\text{cum}}(s^*)$$
$$\leq \underbrace{\gamma^{(k')} \cdot M}_{a} - \underbrace{\sum_{k=k'}^{k'+n-1} \gamma^{(k)} \cdot \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{abs}}, \omega)}_{b} - \underbrace{\delta C \cdot \frac{\gamma^{k'+n}}{1-\gamma}}_{c} \tag{46}$$

For $\gamma \to 1$, terms a and b tends to a constant, while term c tends to infinity. Therefore, $\lim_{\gamma \to 1} (C^{\gamma}_{\text{cum}}(s'') - C^{\gamma}_{\text{cum}}(s^*)) = -\infty$. This means that if γ is sufficiently close to 1, then $C^{\gamma}_{\text{cum}}(s'') < C^{\gamma}_{\text{cum}}(s^*)$, which is absurd as it violates Definitions B-C.2 and B-C.3. □

D. Convergence of the offline sequence

In this section, we will prove that sequence s of actions induced by $Q^{(k)}$ converges to the sequence of actions induced by Q^*.

Definition B-D.1. *Let s_1 and s_2 be two sequences of actions and states. We denote the difference of average expected cost induced by sequences s_1 and s_2 by:*

$$D(s_1, s_2) \triangleq \lim_{T \to \infty} \frac{1}{T} \mathbb{E}\left[\sum_{k=0}^{T} \left(C_1^{(k)} - C_2^{(k)}\right)\right] \tag{47}$$

where $C^{(k)}$ is the instantaneous cost defined by (3)*.*

Definition B-D.2. *We say that a sequence s_1 converges to a sequence s_2 if and only if*

$$D(s_1, s_2) = 0.$$

Proposition B-D.3. *Let $Q_1^{(k)}$ and $Q_2^{(k)}$ be two sequences of Q-tables such that $\arg\min_{\mathbf{a}} Q_1^{(k)}(\boldsymbol{\theta}, \mathbf{a}) = \arg\min_{\mathbf{a}} Q_2^{(k)}(\boldsymbol{\theta}, \mathbf{a}), \forall k > k', \forall \boldsymbol{\theta} \in \mathcal{S}$. Then, if a sequence s_1 is induced by $Q_1^{(k)}$, it must be induced by $Q_2^{(k)}$ starting from k'.*

The following lemma proves that our offline sequence s approaches the optimal sequence of states and actions.

Lemma B-D.4. *If s is a sequence induced by Q-tables $Q^{(k)}$, obtained with our Algorithm 1, then*

$$\mathbb{P}\Big(\exists K > 0, s \text{ is induced by } Q^* \text{ starting from } K\Big) = 1$$

Proof. In Theorem B-B.3, we proved that $Q^{(k)}$ converges with probability 1 to Q^*. Then, by definition, we have

$$\mathbb{P}\Big(\lim_{k \to \infty} |Q^{(k)}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a})| < e\Big) = 1, \forall e > 0, \forall (\boldsymbol{\theta}, \mathbf{a}) \tag{47bis}$$

Let us denote by ϵ_{min} the minimum difference between two distinct Q-values of Q^*:

$$\epsilon_{\text{min}} = \min \left\{ |Q^*(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}', \mathbf{a}')| \;\middle|\; \begin{array}{l} \boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{S}, \\ \mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}}, \mathbf{a}' \in \mathcal{A}_{\boldsymbol{\theta}'}, \\ Q^*(\boldsymbol{\theta}, \mathbf{a}) \neq Q^*(\boldsymbol{\theta}', \mathbf{a}') \end{array} \right\}$$

Formula (47bis) implies that

$$\mathbb{P}\Big(\lim_{k \to \infty} |Q^{(k)}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a})| < \frac{\epsilon_{\text{min}}}{2}\Big) = 1, \forall (\boldsymbol{\theta}, \mathbf{a}).$$

This implies that, $\forall (\boldsymbol{\theta}, \mathbf{a})$,

$$\mathbb{P}\Big(\exists K_{\boldsymbol{\theta},\mathbf{a}} > 0, \forall k > K_{\boldsymbol{\theta},\mathbf{a}}, |Q^{(k)}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a})| < \epsilon_{\text{min}}\Big) = 1. \tag{47tris}$$

Let us call $E_{\boldsymbol{\theta},\mathbf{a}}$ the following event:

$$\exists K_{\boldsymbol{\theta},\mathbf{a}} > 0, \forall k > K_{\boldsymbol{\theta},\mathbf{a}}, |Q^{(k)}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a})| < \epsilon_{\text{min}}$$

Formula (47tris) implies that $\mathbb{P}(E_{\boldsymbol{\theta},\mathbf{a}}) = 1, \forall (\boldsymbol{\theta}, \mathbf{a})$. Hence $\mathbb{P}(\bar{E}_{\boldsymbol{\theta},\mathbf{a}}) = 0, \forall (\boldsymbol{\theta}, \mathbf{a})$. By taking $K = \max_{\boldsymbol{\theta},\mathbf{a}} K_{\boldsymbol{\theta},\mathbf{a}}$ (that exists since $\mathcal{S}$ and $\mathcal{A}$ are finite), we can write:

$$\mathbb{P}\left(\begin{array}{c} \exists K > 0, \forall k > K, \forall \boldsymbol{\theta} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}} \\ \Rightarrow |Q^{(k)}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a})| < \epsilon_{\text{min}} \end{array}\right) = 1$$

and thus

$$\mathbb{P}\left(\begin{array}{c} \exists K > 0, \forall k > K, \forall \boldsymbol{\theta} \in \mathcal{S} \\ \Rightarrow \arg\min_{\mathbf{a}} Q^{(k)}(\boldsymbol{\theta}, \mathbf{a}) = \arg\min_{\mathbf{a}} Q^*(\boldsymbol{\theta}, \mathbf{a}) \end{array}\right) = 1$$

Thanks to Proposition B-D.3, we will have that s is induced by Q^* starting from K, hence the result. □

Corollary B-D.5. *Sequence s converges to sequence s^*.*

Proof. Thanks to Lemma B-C.5, we can claim that $\exists K^* > 0$, starting from which s^* has arrived to a discretely optimal state $\hat{\boldsymbol{\theta}}^*$. Lemma B-D.4 allows us to write:

$$\mathbb{P}\Big(\exists K > 0, \forall k > K, s \text{ takes actions induced by } Q^*\Big) = 1$$

Let $K_{\text{max}} = \max(K^*, K)$ and let $\mathcal{E}$ be the following event:

$$\exists K_s^* \geq K_{\text{max}}, \text{ starting from which } s \text{ has arrived to } \hat{\boldsymbol{\theta}}^*.$$

Thanks to Lemma B-C.5 and Lemma B-D.4, we can write:

$$\mathbb{P}(\mathcal{E}) = 1$$

Let us now compute $D(s, s^*)$ by applying the law of total expectations:

$$D(s, s^*) = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}\left[\sum_{k=0}^{T} \left(C^{(k)} - C^{*(k)}\right)\right]$$
$$= \lim_{T \to \infty} \Bigg(\mathbb{P}(\mathcal{E}) \cdot \frac{1}{T} \mathbb{E}\left[\sum_{k=0}^{T} \left(C^{(k)} - C^{*(k)}\right) | \mathcal{E}\right]$$
$$+ (1 - \mathbb{P}(\mathcal{E})) \cdot \frac{1}{T} \mathbb{E}\left[\sum_{k}^{T} \left(C^{(k)} - C^{*(k)}\right) | \bar{\mathcal{E}}\right]\Bigg) \quad \text{(the bracket} \to 0\text{)}$$
$$= \lim_{T \to \infty} \frac{1}{T} \sum_{k=0}^{T} \Big[\mathbb{E}[C^{(k)} | \mathcal{E}] - \mathbb{E}[C^{*(k)} | \mathcal{E}]\Big]$$

After $K_c = \max(K_s^*, K^*)$ both s and s^* will be in the same state $\hat{\boldsymbol{\theta}}^*$, if event $\mathcal{E}$ is verified. Therefore, $\mathbb{E}[C^{(k)}|\mathcal{E}] = \mathbb{E}[C^{*(k)}|\mathcal{E}], \forall k \geq K_c$ and thus

$$D(s, s^*) = \lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{K_c} \Big[\mathbb{E}[C^{(k)}|\mathcal{E}] - \mathbb{E}[C^{*(k)}|\mathcal{E}]\Big] = 0$$

□

E. Convergence of the online sequence

In this section, we will prove that the sequence s_ϵ of states and actions visited by Algorithm 1 online, converges to $\hat{\boldsymbol{\theta}}^*$, which represents the main result of our work.

To do so, we need to show that

$$\exists\ \hat{\boldsymbol{\theta}}^*,\ \mathbb{P}\bigg(\lim_{k\to\infty}\ ||\ \boldsymbol{\theta}^{(k)} - \hat{\boldsymbol{\theta}}^*||\ > 0\ \bigg) = 0.$$

We will prove a stronger property: $\exists$ discretely optimal state $\hat{\boldsymbol{\theta}}^* \in \mathcal{S}$, such that

$$\lim_{k\to\infty} \mathbb{P}(\boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^*) = 1. \tag{48}$$

Let us denote with $\mathcal{B}(k, d)$ the event that from time-slot $k - d$ to $k - 1$ there have been no random actions taken. The probability of this event is:

$$\mathbb{P}(\mathcal{B}(k, d)) = \prod_{k'=k-d}^{k} (1 - \epsilon^{(k')}) \tag{49}$$

where $\epsilon^{(k')}$ follows (18). Since $\lim_{k\to\infty} \epsilon^{(k)} = 0$, then $\forall d > 0, \lim_{k\to\infty} \mathbb{P}(\mathcal{B}(k, d)) = 1$. If $\mathcal{B}(k, d)$ is verified, the actions taken by our algorithm are those induced by $Q^{(k'')}, \forall k'' = k - d, .., k$. Therefore, Lemma B-D.4 applies and if we take $d > K$ (where K is the one indicated by Lemma B-D.4), we know that if $\mathcal{B}(k, d)$ is verified, then the actions taken by our algorithm in time-slots $k'' = k-d+K, .., k$ are those suggested by Q^*. Hence, Lemma B-C.4 applies and if we take $d > K+k'$ (where K is the one indicated by Lemma B-C.4), we know that the state in which our algorithm brings the system in slots $k'' = k - d + K + k', .., k$, is an absorbing state.

Thanks to Lemma B-C.5, we know that this absorbing state is $\hat{\boldsymbol{\theta}}^*$, if γ is sufficiently large. Therefore, if $\mathcal{B}(k, d)$ is verified, then

$$\boldsymbol{\theta}^{(k'')} = \hat{\boldsymbol{\theta}}^*, \forall k'' = k - d + K + k', .., k.$$

In other words, if $\mathcal{B}(k, d)$ is verified for a sufficiently large γ, then $\exists$ discretely optimal state $\hat{\boldsymbol{\theta}}^*$ such that $\boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^*$. Therefore, $\mathbb{P}(\boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^*) \geq \mathbb{P}(\mathcal{B}(k, d))$. Since the second term tends to 1 when $k \to \infty$ (49), then (48) is verified.

APPENDIX C
PROOF OF THE COROLLARY V-B.3

Proof. We have seen that our allocations have an absorbing state $\hat{\boldsymbol{\theta}}^*$ that is discretely optimal, i.e., there exists $k' > 0$ such that $\boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^*$ for $k \geq k'$ with probability 1. We can compute

$$\begin{aligned}
&\lim_{Z\to\infty} \frac{1}{Z}\mathbb{E}\left[C_{\text{cum}}(Z) - C^*_{\text{cum}}(Z)\right] \\
&= \lim_{Z\to\infty} \frac{1}{Z}\mathbb{E}\left[C_{\text{cum}}(k') - C^*_{\text{cum}}(k')\right] \\
&+ \lim_{Z\to\infty} \frac{1}{Z}\mathbb{E}\Big[C_{\text{cum}}(Z) - C_{\text{cum}}(k') \\
&- \Big(C^*_{\text{cum}}(Z) - C^*_{\text{cum}}(k')\Big)\Big] \\
&= \lim_{Z\to\infty} \frac{1}{Z} \sum_{k=k'+1}^{Z} \Big(\mathbb{E}\Big[(C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}^{(k)}) \\
&- C_{\text{nom}}(\boldsymbol{\theta}^*, \omega)\Big]\Big) \\
&= \lim_{Z\to\infty} \frac{Z - (k'+1)}{Z} \cdot \Big(\mathbb{E}_\omega[C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega)] \\
&- \mathbb{E}_\omega[C_{\text{nom}}(\boldsymbol{\theta}^*, \omega)]\Big) \\
&= \mathbb{E}_\omega[C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega)] - \mathbb{E}_\omega[C_{\text{nom}}(\boldsymbol{\theta}^*, \omega)] = G_\Delta
\end{aligned}$$

□

APPENDIX D
PROOF OF PROPOSITION V-B.2

Suppose an oracle that knows exactly (i) the probability f_p that a request is for SP p, (ii) the cacheability ζ_p and (iii) the popularity of each object (c, p) within the catalog of each SP. Such an oracle can compute the rate of requests $\lambda_{c,p} = \lambda \cdot f_p \cdot \zeta_p \cdot \rho_{c,p}$ for each object. The expected value of the nominal cost when the set of cached objects is $\mathcal{K}$ is

$$\mathbb{E}C_{\text{nom}}(\mathcal{K}) = \lambda - \sum_{(c,p)\in\mathcal{K}} \lambda_{c,p}$$

The following property will be useful later.

Lemma D-.1. *The set function $\mathcal{K} \to \mathbb{E}C_{nom}(\mathcal{K})$ is monotonically increasing, i.e., if $\mathcal{K} \subseteq \mathcal{K}'$, then $\mathbb{E}C_{nom}(\mathcal{K}) \leq \mathbb{E}C_{nom}(\mathcal{K}')$.*

To minimize the nominal cost, we resort to a greedy algorithm for the Simple Allocation Problem [71]: the oracle puts into the cache the objects with the highest $\lambda_{c,p}$, up to filling all K cache-slots. Let us denote with $\mathcal{K}^*$ the set of cached objects in this way. The optimal allocation $\boldsymbol{\theta}^*$ can be obtained by simply counting the number of objects of each SP p that we find in $\mathcal{K}^*$. In particular, θ^*_p is equal to the number of objects of SP p present in $\mathcal{K}^*$.

With no loss of generality, suppose that within the catalog of each SP p the objects are indexed as $c = 1, 2, \ldots, N_p$ and sorted from the most popular to the least, so that $\lambda_{c,p} \geq \lambda_{c+1,p}$. If the discretization step is Δ, objects cannot be selected one by one, but they can only be cached in batches of Δ elements. We thus divide the catalog of each SP in batches of Δ objects. For instance, batch $\mathcal{B}_{i,p}$ is

$$\mathcal{B}_{i,p} = \{\text{object}(c, p)|c = (i-1)\cdot\Delta + 1, \ldots, i\cdot\Delta\},\ i = 1, 2, ..$$

The rate of such a batch is defined as the traffic we can omit downloading from a distant location if we store this batch into the cache, i.e.:

$$\lambda(\mathcal{B}_{i,p}) \triangleq \sum_{(c,p)\in\mathcal{B}_{i,p}} \lambda_{c,p}.$$

In order to minimize the nominal cost with the discretized model, the oracle can add to the cache the batches with the highest rate, up to filling the K cache-slots. We denote by $\hat{\mathcal{K}}^*$ the set of cached objects obtained in this way. The discretely optimal allocation $\hat{\boldsymbol{\theta}}^* = (\hat{\theta}_1^*, \dots, \hat{\theta}_P^*)$ can be obtained by simple counting: $\hat{\theta}_p^*$ is equal to the number of objects of p that are present in $\hat{\mathcal{K}}^*$.

The construction of $\hat{\mathcal{K}}^*$ and $\hat{\boldsymbol{\theta}}^*$ is summarized in Algorithm 2, which extends the greedy algorithm. Note that to construct $\mathcal{K}^*$ and $\boldsymbol{\theta}^*$ one can use the same algorithm, setting $\Delta = 1$.

Algorithm 2: Compute $\hat{\boldsymbol{\theta}}^*$

Data: Δ, $\lambda_{c,p} \forall (c,p)$.
Result: $\hat{\mathcal{K}}^*$, $\hat{\boldsymbol{\theta}}^*$

$\hat{\mathcal{K}}^* \leftarrow \emptyset$;
$\hat{\boldsymbol{\theta}}^* \leftarrow \mathbf{0} = (0, \dots, 0)$;
$i_p = 1$ for $p = 1, \dots, P$; // We use this pointer to save the last added batch of each SP
while $|\hat{\mathcal{K}}^*| + \Delta < K$; // We can still add a batch of Δ objects in the cache
do
  $p^{\text{best}} \in \arg\max_{p=1}^{P} \lambda(\mathcal{B}_{i_p,p})$; // Select the SP with the largest batch rate
  $\hat{\mathcal{K}}^* \leftarrow \hat{\mathcal{K}}^* \cup \mathcal{B}_{i_{p^{\text{best}}},p^{\text{best}}}$; // Add batch of SP p^{best} in the cache
  $\hat{\theta}^*_{p^{\text{best}}} \leftarrow \hat{\theta}^*_{p^{\text{best}}} + \Delta$; // Give it the corresponding cache-slots
end

The optimality gap can be expressed in terms of the sets $\mathcal{K}^*$ and $\hat{\mathcal{K}}^*$:

$$G_\Delta = \mathbb{E}C_{\text{nom}}(\hat{\mathcal{K}}^*) - \mathbb{E}C_{\text{nom}}(\mathcal{K}^*) \tag{50}$$

In order to bound the previous quantity, we construct two sets $\mathcal{K}_-$ and $\mathcal{K}_+$ around $\mathcal{K}^*$ and $\hat{\mathcal{K}}^*$.

Lemma D-.2. *There exists two sets $\mathcal{K}_-$ and $\mathcal{K}_+$ such that*

$$\mathcal{K}_- \subseteq \mathcal{K}^* \subseteq \mathcal{K}_+ \tag{51}$$
$$\mathcal{K}_- \subseteq \hat{\mathcal{K}}^* \subseteq \mathcal{K}_+ \tag{52}$$

and

$$\mathbb{E}C_{nom}(\mathcal{K}_-) - \mathbb{E}C_{nom}(\mathcal{K}_+) \leq \sum_{p=1}^{P} \sum_{c=1}^{\Delta} \lambda_{c,p}$$

Proof. We know that the θ_p^* most popular objects of SP p are stored in $\mathcal{K}^*$. These include the $\left\lfloor \frac{\theta_p^*}{\Delta} \right\rfloor$ batches of SP p with the highest rate. Let us construct a set composed of these batches:

$$\mathcal{K}_- = \bigcup_{p=1}^{P} \left\{ \bigcup_{i=1}^{\left\lfloor \frac{\theta_p^*}{\Delta} \right\rfloor} \mathcal{B}_{i,p} \right\} \tag{53}$$

By construction, $\mathcal{K}_- \subseteq \mathcal{K}^*$. By construction of $\hat{\mathcal{K}}^*$, the aforementioned batches are also contained into $\hat{\mathcal{K}}^*$. Therefore, $\mathcal{K}_- \subseteq \hat{\mathcal{K}}^*$.

We now construct set $\mathcal{K}_+$ adding to $\mathcal{K}_-$ one additional batch per each SP:

$$\mathcal{K}_+ = \bigcup_{p=1}^{P} \left\{ \bigcup_{i=1}^{\left\lceil \frac{\theta_p^*}{\Delta} \right\rceil} \mathcal{B}_{i,p} \right\}$$

By construction $\mathcal{K}_+ \supseteq \mathcal{K}^*$ and $\mathcal{K}_+ \supseteq \hat{\mathcal{K}}^*$. Summarizing what we have obtained so far:

$$\mathcal{K}_- \subseteq \mathcal{K}^* \subseteq \mathcal{K}_+$$
$$\mathcal{K}_- \subseteq \hat{\mathcal{K}}^* \subseteq \mathcal{K}_+$$

By construction, we have

$$\mathbb{E}C_{\text{nom}}(\mathcal{K}_+) - \mathbb{E}C_{\text{nom}}(\mathcal{K}_-) = \sum_{p=1}^{P} \left(\sum_{i=\left\lfloor \frac{\theta_p^*}{\Delta} \right\rfloor}^{\left\lceil \frac{\theta_p^*}{\Delta} \right\rceil} \lambda(\mathcal{B}_{i,p}) \right) \leq \sum_{p=1}^{P} \lambda(\mathcal{B}_{i',p})$$

where $i' = \left\lfloor \frac{\theta_p^*}{\Delta} \right\rfloor$. Since objects are sorted, within each SP p, from the highest to the lowest rate, the batches have decreasing rate, and thus $\lambda(\mathcal{B}_{i',p}) \leq \lambda(\mathcal{B}_{1,p}) = \sum_{c=1}^{\Delta} \lambda_{c,p}$. □

Thanks to Lemma D-.1, equations (51)-(52) imply that

$$\mathbb{E}C_{\text{nom}}(\mathcal{K}_-) \leq \mathbb{E}C_{\text{nom}}(\mathcal{K}^*) \leq \mathbb{E}C_{\text{nom}}(\mathcal{K}_+)$$
$$\mathbb{E}C_{\text{nom}}(\mathcal{K}_-) \leq \mathbb{E}C_{\text{nom}}(\hat{\mathcal{K}}^*) \leq \mathbb{E}C_{\text{nom}}(\mathcal{K}_+)$$

which, in turn, imply that $\mathbb{E}C_{\text{nom}}(\hat{\mathcal{K}}^*) - \mathbb{E}C_{\text{nom}}(\mathcal{K}^*) \leq \mathbb{E}C_{\text{nom}}(\mathcal{K}_-) - \mathbb{E}C_{\text{nom}}(\mathcal{K}_+)$. Using (50) and (53), we obtain the proposition.