Dynamic Time-of-use Pricing for Serverless Edge Computing with Generalized Hidden Parameter Markov Decision Processes

Feridun Tütüncüoğlu[†], Ayoub Ben-Ameur[‡], György Dán[†], Andrea Araldo[‡], Tijani Chahed[‡]
[†]Division of Network and Systems Engineering, KTH Royal Institute of Technology, Sweden
[‡]SAMOVAR, Telecom SudParis, Institut Polytechnique de Paris, France

***Abstract*—The commercial adoption of Edge Computing (EC) will require pricing schemes that cater to the financial interests of the operators and of the users. Pricing in EC is particularly challenging as it has to take into account the limited amount of edge resources as well as the stochasticity of user workloads due to location-specific workload characteristics and differences in user activity. We formulate the problem of maximizing the revenue of a serverless edge operator through dynamically pricing compute and memory resources under time varying workloads as a sequential decision making problem under uncertainty. We provide analytical results for the optimal pricing strategy in a Markovian setting in steady state. For the general case, we propose a novel Generalized Hidden Parameter Markov Decision Process (GHP-MDP) formulation of the revenue maximization problem, and we propose a dual Bayesian neural network approximator as a solution. The key novelty of the proposed solution is that it can be pre-trained on synthetic traces and adapts fast to previously unseen workload characteristics. We use simulations based on synthetic and real traffic traces to show that the proposed solution is sample-efficient thanks to effective transfer learning, and it outperforms state-of-the-art learning approaches in terms of revenue and learning rate by up to 50% on real traces.**

***Index Terms*—Serverless edge computing, dynamic pricing, resource management, queuing theory, transfer learning.**

I. Introduction

Edge computing (EC) is expected to bring computational resources close to the network edge, e.g., co-located with base stations, and will allow Wireless Devices (WDs) to offload latency sensitive computational tasks on-demand. EC is expected to be a key enabler of new services, such as augmented reality [1] and cooperative driving, services that would be too resource intensive to be executed locally or ones that require information from multiple sources.

Despite its potential, EC is rather far from commercial deployment [2]. Existing edge deployments consist of cloud resources deployed at the periphery of the core network (not as far as base stations) and are targeted at corporate users, as a localized version of cloud computing [3]. The slow deployment is partly due to the lack of a versatile programming abstraction and due to the lack of an appropriate business model, including appropriate pricing schemes.

A promising abstraction for edge computing could be serverless computing, which allows the execution of functions, relieving the users from managing virtual machines, compute and memory resources [4]. Serverless computing has found adoption in cloud computing (CC), but the static pricing models that made it popular would not suit edge deployments for several reasons. First, compared to CC, where resources are practically unlimited [5], edge nodes have scarce resources. Hence, some WDs may not get the resources they request immediately, and they may leave the edge service area before they would get served. Moreover, cloud nodes benefit from statistical multiplexing due to requests coming from a broad geographical region [6]. This is not true in EC, hence demand for compute resources is more dynamic.

While practically all the big CC providers offer static, usage-based pricing, serving dynamic demands with constrained resources requires EC pricing to be dynamic, fulfilling two main criteria. First, pricing should be adaptive, i.e., it should learn how to maximize revenue given information about the workload and the available resources. Adaptation should be fast, at the time scale of the workload dynamics. Second, pricing should be transparent to users, so that users can incorporate pricing information in the long term decisions whether to rely on EC for executing their tasks. Finding a pricing scheme that is adaptive, transparent and computationally efficient is, however, extremely challenging. In this paper, we address this challenge and make the following main contributions:

- We formulate the problem of maximizing the revenue of a serverless edge operator as a sequential decision making problem under uncertainty. In our formulation, prices are piecewise constant over time and can be non-linear functions of the requested resources, unlike in existing works [7], [8].
- We propose a steady state approximation of the problem, and provide analytical results for the optimal pricing strategy under mild assumptions. These analytical results provide a lower bound for the revenue achievable by the operator, and at the same time they serve as a good heuristic.
- We provide a novel Generalized Hidden Parameter Markov Decision Process (GHP-MDP) formulation of the problem, and use it to propose a learning scheme based on a dual Bayesian Neural Network (BNN) approximator for fast and accurate transfer learning. The proposed Hidden Parameter Edge (HiPE) pricing algorithm learns latent variables that capture the parameters of the problem instance, and uses these for parameterizing two BNNs used for training a state-of-the-art Reinforcement Learning (RL) algorithm. Our ap-

Notation	Definition
t_i^a	Arrival time of user i to the edge area
D_i	Dwell time of user i
t_i^d	Departure time of user i from the edge area
t_i^o	Time instant at which user i starts offloading
o_i	Offloading decision of user i
$\mathcal{T}_{i,k}^a$	Active time of user i during pricing period T_k
$\mathcal{T}_i^a$	Total active time of user i
α_i	Task generation rate of user i
$\mathcal{F}$, $\mathcal{M}$	Set of CPU capacity and memory allocations
F, M	Edge CPU and memory capacity
$\tau_v(f_v)$, $\overline{\tau}_v$	Execution time and delay bound of function v
$\mathcal{V}_i$	Set of functions of user i
$\Lambda(t)$	User arrival intensity at time t
n_v	Average number of invocations of function v
f_v, m_v	Requested CPU capacity and memory for function v
T_k	Pricing period k, $T_k = [t_k, t_{k+1})$
Δ	Length of pricing periods
$\pi_k^{\boldsymbol{f}}(f)$	Price of requested CPU f during T_k
π_k^{CPU}, π_k^{MEM}	Unit cost of CPU and of memory
$\pi_k^{\boldsymbol{m}}(m)$	Price of requested memory m during T_k
π_k^r	Price paid per function invocation during T_k
$C_i^{\boldsymbol{\pi}_k}$	Expected unit task offloading cost of user i
C_i	Reservation cost of user i
$C_{i,\Sigma}^{\boldsymbol{\pi}_k}$	Total cost of offloading of user i

Table I: Frequently used notations.

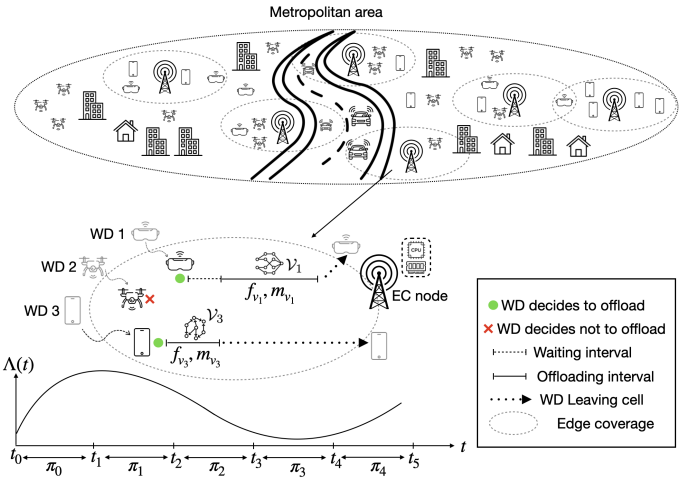

Figure 1: Example of a serverless edge service with time-dependent arrival rate $\Lambda(t)$ and piecewise constant price $\boldsymbol{\pi}_k$. Three WDs arrive in pricing periods T_1 and T_2: WD 3 receives service immediately upon arrival at price $\boldsymbol{\pi}_1$, while WD 2 decides not to offload at price $\boldsymbol{\pi}_1$. WD 1 has to wait for receiving service, at price $\boldsymbol{\pi}_2$.

proach accelerates learning an optimal policy, thus it allows higher average revenue without resorting to assumptions regarding the workload and the edge cell dynamics.

- Extensive simulations on synthetic and real traces show that HiPE outperforms state-of-the-art solutions in terms of revenue by up to 50% as well as consumer surplus.

The rest of the paper is organized as follows. We describe the system model and the problem formulation in Section II. In Section III, we provide analytical results under simplifying assumptions. In Section IV we propose a GHP-MDP formulation of the problem and a dual BNN approximator based solution for the pricing problem. We provide numerical results in Section V. In Section VI, we review related work and we conclude the paper in Section VII.

II. System Model and Problem Formulation

We consider a Multi-access Edge Computing (MEC) system that provides Function as a Service (FaaS) (also known as serverless) computing to a dynamic population of WDs.

A. User Model

WDs arrive to the system following a non-homogeneous process with intensity $\Lambda(t)$ as shown in Fig. 1. WD i remains in the edge service area for D_i amount of time, which we refer to as the dwell time of WD i. We denote by t_i^a and $t_i^d = t_i^a + D_i$ the arrival and departure times of WD i, respectively. WD i generates tasks at rate $\alpha_i > 0$ while it is in the MEC service area. Following the FaaS model, each task of user i involves the execution of a set $\mathcal{V}_i = \{v_1, v_2, \ldots, v_{V_i}\}$ of functions according to a possibly loopy task graph, and we denote by n_v the average number of invocations of function $v \in \mathcal{V}_i$.

User i requests CPU allocation f_v and memory allocation m_v for function $v \in \mathcal{V}_i$, which determine the processing power (in Hz) and memory capacity (in GB) allocated for the function, respectively. The expected execution time $\tau_v(f_v)$ of function $v \in \mathcal{V}_i$ is a convex non-increasing function of the computing power f_v allocated to it [9]. This model generalizes the relation $\tau_v(f_v) = \frac{\mathbb{E}[L_v]}{f_v}$ widely used in edge computing, where $\mathbb{E}[L_v]$ is the average computational complexity of the function measured in CPU cycles. Tasks have finite expected execution time, i.e., $\tau_i(f_{\mathcal{V}_i}) \in (0, \infty)$, where $f_{\mathcal{V}_i} = (f_{v_1}, f_{v_2}, \ldots, f_{v_{V_i}})$ denotes the vector of all CPU allocations given to the tasks in $\mathcal{V}_i$. Moreover, WD i has a constraint $\tau_v(f_v) \leq \overline{\tau}_v$ on the average execution time of its functions $v \in \mathcal{V}_i$, determined based on the delay bound $\overline{\tau}_i \in (0, \infty)$ of its task. We include in $\tau_v(f_v)$ the potential impact of storage access latency [10], [11]. For ease of exposition, we assume that communication delays are measured by the edge operator and WDs determine $\overline{\tau}_v$ considering the communication delays.

B. Edge Resources, Pricing and Offloading

We consider that the operator maintains an edge cloud with CPU capacity F and memory M in the service area [12]. Aligned with common FaaS offerings (e.g. AWS Lambda, Google Cloud Functions), we consider that the operator offers a set of CPU allocations $\mathcal{F} = \{f_{(1)}, f_{(2)}, \ldots, \overline{f}\}$, in ascending order, allowing WDs to select the most appropriate allocation for their tasks' functions [13]. In addition, to ensure that the WDs' tasks can be executed, the operator provides a set of memory allocations $\mathcal{M} = \{m_{(1)}, m_{(2)}, \ldots, \overline{m}\}$, in ascending order. Naturally, $\overline{f} \leq F$ and $\overline{m} \leq M$.

Similar to existing FaaS pricing models, we consider that pricing is based on the execution time, on the amount of resources used, and on the number of function invocations.

The operator sets the prices periodically, at time instants $t_0, t_1 = t_0 + \Delta, \ldots$; the price is constant during pricing period $T_k = [t_k, t_{k+1})$ to make the cost of using the edge service more predictable for the users than under user specific pricing schemes considered in previous works [14], [15].

Contrary to existing FaaS pricing models, we consider a general, non-linear pricing model. The price of compute capacity for CPU allocation f during pricing period T_k is $\pi_k^{\boldsymbol{f}}(f) = f^{\gamma_k}\pi_k^{CPU}$ and the price of memory allocation is $\pi_k^{\boldsymbol{m}}(m) = m^{\gamma_k}\pi_k^{MEM}$, where π_k^{CPU} and π_k^{MEM} are the unit cost of compute capacity (CPU allocation) and memory, respectively, and $\gamma_k \geq 1$ is the price exponent. Observe that for $\gamma_k = 1$ the price is linear in the allocation, as in current FaaS offerings, while for $\gamma_k > 1$ it is nonlinear. We denote by π_k^r the price paid per function invocation during pricing period T_k, and we use the shorthand notation $\boldsymbol{\pi}_k = (\pi_k^{CPU}, \pi_k^{MEM}, \pi_k^r, \gamma_k,)$.

If WD i arrives during pricing period T_k then it will be charged based on the price $\boldsymbol{\pi}_k$ throughout its dwell time D_i, should it decide to offload, even if the dwell time overlaps with subsequent pricing periods. Thus, its expected *unit* cost for offloading a task will be determined by the functions $\mathcal{V}_i$ that constitute its task and by their average number of invocations n_v, $v \in \mathcal{V}_i$, by its choice of $(f_v)_{v\in\mathcal{V}_i}$ and $(m_v)_{v\in\mathcal{V}_i}$, and by the price $\boldsymbol{\pi}_k$, and can be expressed as

$$C_i^{\boldsymbol{\pi}_k}((f_v)_{v\in\mathcal{V}_i}, (m_v)_{v\in\mathcal{V}_i}) = \sum_{v\in\mathcal{V}_i} n_v\Big(\pi_k^r + \tau_v(f_v)(\pi_k^{\boldsymbol{f}}(f_v) + \pi_k^{\boldsymbol{m}}(m_v))\Big). \quad (1)$$

Let us denote by $\overline{C}_i$ the reservation cost of WD i, i.e., its valuation for offloading its task. We make the reasonable assumption that $\overline{C}_i$ is unknown to the operator, and we model it as a random variable with distribution C. WD i decides to offload if its expected unit cost does not exceed its reservation price, i.e.,

$$C_i^{\boldsymbol{\pi}_k}((f_v)_{v\in\mathcal{V}_i}, (m_v)_{v\in\mathcal{V}_i}) \leq \overline{C}_i, \quad (2)$$

otherwise WD i executes the computations locally or discards its tasks. We denote by $o_i \in \{0, 1\}$ the decision of WD i.

Due to edge resource constraints, even if WD i decides to offload, it may not be able to do so immediately, but it may have to wait in a FIFO queue until resources become available. A waiting WD does not offload, hence it does not have to pay for offloading, and if its dwell time D_i expires during the waiting phase, the WD will leave the system without ever offloading. We denote by t_i^o the time instant when WD i can start offloading, hence $t_i^a \leq t_i^o \leq t_i^d$ if $o_i = 1$, and we define $t_i^o = t_i^d$ if $o_i = 0$. We can use these to define the active time of WD i during pricing period T_k as $\mathcal{T}_{i,k}^a = \max\{\min(t_i^d, t_{k+1}) - \max(t_i^o, t_k), 0\}$ and its total active time $\mathcal{T}_i^a = t_i^d - t_i^o$. The active time, the task arrival rate and the unit task offloading cost together determine the expected cost of offloading

$$C_{i,\Sigma}^{\boldsymbol{\pi}_k}((f_v)_{v\in\mathcal{V}_i}, (m_v)_{v\in\mathcal{V}_i}) = \mathcal{T}_i^a \alpha_i C_i^{\boldsymbol{\pi}_k}((f_v)_{v\in\mathcal{V}_i}, (m_v)_{v\in\mathcal{V}_i}). \quad (3)$$

C. Problem Formulation

We consider that the WDs and the operator are profit maximizing entities. The goal of WD i is to minimize its cost of offloading subject to the latency constraints $\big(\overline{\tau}_v(f_v)\big)_{v\in\mathcal{V}_i}$ of the functions $v \in \mathcal{V}_i$ of its task, i.e.,

$$\max_{o_i, (m_v, f_v)_{v\in\mathcal{V}_i}} o_i \cdot (\overline{C}_i - C_i^{\boldsymbol{\pi}_k}((f_v)_{v\in\mathcal{V}_i}, (m_v)_{v\in\mathcal{V}_i})) \quad (4)$$

$$s.t \quad o_i \cdot \tau_v(f_v) \leq \overline{\tau}_v, \forall v \in \mathcal{V}_i. \quad (5)$$

Hence, WD i chooses the CPU allocation $(f_v)_{v\in\mathcal{V}_i}$ and memory allocation $(m_v)_{v\in\mathcal{V}_i}$ that minimize its cost (1).

In pricing period T_k the operator collects revenue from the WDs that offload in the period. Recall, however, that the price $\boldsymbol{\pi}'_k$ that WD i is charged depends on the pricing period $T'_k \ni t_i^a$ when it arrived. We can thus express the revenue of the operator in pricing period T_k for a pricing policy θ as

$$\rho_k^\theta = \sum_{k'=0}^{k} \sum_{i\in\mathcal{N}_{k'}} \mathcal{T}_{i,k}^a \alpha_i C_i^{\boldsymbol{\pi}_{k'}}((f_v)_{v\in\mathcal{V}_i}, (m_v)_{v\in\mathcal{V}_i}), \quad (6)$$

where $\mathcal{N}_k = \{i | t_i^a \in T_k\}$ is the set of users that arrived in pricing period T_k and $\mathcal{T}_{i,k}^a$ is the active time of user i in pricing period T_k. Observe that ρ_k^θ is a random variable, as $C_i^{\boldsymbol{\pi}_k}$ depends on the workload due to tasks of WD i, which is not known by the operator a priori.

The operator's objective is to maximize its expected mean revenue by finding a policy

$$\theta^* = \operatorname*{argmax}_{\theta\in\Theta} \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}[\rho_k^\theta], \quad (7)$$

where Θ is the set of causal pricing policies. Observe that the operator's problem is a sequential decision making problem under uncertainty, where the uncertainty is due to the randomness of the arrivals, departures, reservation costs and resource requirements of the WDs. In what follows we first characterize the WDs' best response and provide analytical results under simplifying assumptions. We then propose a pricing policy based on a semi-parametric approach for the general case.

III. Analytical Results

To obtain insight into the structure of optimal policies, we start with characterizing the best response of the users for a given pricing policy. We then turn to the analysis of pricing policies.

A. WD Best Response Characterization

We first make a simple but important observation about the memory allocations for the functions.

Observation 1. *The execution time $\tau_v(f_v)$ is independent of the memory allocation m_v, while the cost* (1) *is monotonically increasing in m_v. Hence for WD i it is optimal to request the least amount of memory m_v^* that allows function v to be executed.*

Nonetheless, the dependence of the cost on the amount of f_v makes the optimal choice of the CPU allocation non-trivial.

Lemma 1. *Assume a non-linear pricing model with* $\boldsymbol{\pi}_k = (\pi^{CPU}, \pi^{MEM}, \pi^r, \gamma)$*, for some* $\gamma \geq 1$*. Let* $C_i^{\boldsymbol{\pi}_k}(f_v, m_v)$ *be the unit cost of a single function* v*. Then, the CPU allocation that minimizes the unit cost* $C_i^{\boldsymbol{\pi}_k}(f_v, m_v)$ *of WD* i *for function* $v \in \mathcal{V}_i$ *is*

$$f_v^* = \begin{cases} \operatorname{argmin}_{f \in \{\tilde{f}_v^-, \tilde{f}_v^+\}} C_i^{\boldsymbol{\pi}_k}(f, m_v^*) & \text{if } \tilde{f}_v^- \geq \frac{\mathbb{E}[L_v]}{\bar{\tau}_v} \\ \operatorname{argmin}_{f \in \mathcal{F}} \{ f \mid f \geq \frac{\mathbb{E}[L_v]}{\bar{\tau}_v} \} & \text{otherwise,} \end{cases} \quad (8)$$

where $\tilde{f}_v^-, \tilde{f}_v^+$ *are the two adjacent values in* $\mathcal{F}$*, such that*

$$\tilde{f}_v^- \leq \left(f_v^* = \begin{cases} m_v^* \left(\frac{\pi^{MEM}}{(\gamma-1)\pi^{CPU}} \right)^{\frac{1}{\gamma}} & \text{if } \gamma > 1, \\ \overline{f} & \text{if } \gamma = 1, \end{cases} \right) \leq \tilde{f}_v^+. \quad (9)$$

Proof. We prove the statement by linear relaxation of the discrete variable f_v. For simplicity, we use the same notation f_v and let $f_v \in [f_{(1)}, \overline{f}]$. Recall that the WD's objective is to minimize its unit task offloading cost (1) if it chooses to offload (i.e., if $o_i = 1$). Under the considered pricing model, the unit cost (1) becomes

$$C_i^{\boldsymbol{\pi}_k}((f_v)_{v \in \mathcal{V}_i}, (m_v^*)_{v \in \mathcal{V}_i}) = \sum_{v \in \mathcal{V}_i} n_v \Big(\pi_k^r + \frac{\mathbb{E}[L_v]}{f_v} (f_v^\gamma \pi^{CPU} + (m_v^*)^\gamma \pi^{MEM}) \Big). \quad (10)$$

Observe that $\frac{\partial^2 C_i^{\boldsymbol{\pi}_k}((f_v)_{v \in \mathcal{V}_i}, (m_v^*)_{v \in \mathcal{V}_i})}{\partial f_v \partial f_{v'}} = 0$ for any $v \neq v'$ and $v, v' \in \mathcal{V}_i$, i.e., the Hessian matrix of $C_i^{\boldsymbol{\pi}_k}((f_v)_{v \in \mathcal{V}_i}, (m_v^*)_{v \in \mathcal{V}_i})$ is diagonal.

For $\gamma = 1$ the optimal allocation is trivial to compute and $f_v^* = \overline{f}, \forall v \in \mathcal{V}_i$. For $\gamma \geq 2$, the eigenvalues of the Hessian matrix of $C_i^{\boldsymbol{\pi}_k}((f_v)_{v \in \mathcal{V}_i}, (m_v^*)_{v \in \mathcal{V}_i})$ are non-negative, hence $C_i^{\boldsymbol{\pi}_k}((f_v)_{v \in \mathcal{V}_i}, (m_v^*)_{v \in \mathcal{V}_i})$ is jointly convex [16] in $(f_v)_{v \in \mathcal{V}_i}$. The optimal f_v^* can thus be obtained by setting the partial derivatives to zero, leading to $f_v^* = m_v^* \left(\frac{\pi^{MEM}}{(\gamma-1)\pi^{CPU}} \right)^{\frac{1}{\gamma}}$.
For $1 < \gamma < 2$, the cost is not jointly convex but we observe that $f_v^* = m_v^* \left(\frac{\pi^{MEM}}{(\gamma-1)\pi^{CPU}} \right)^{\frac{1}{\gamma}}$ is optimal, as it is the only value at which $\frac{\partial C_i^{\boldsymbol{\pi}_k}((f_v)_{v \in \mathcal{V}_i}, (m_v^*)_{v \in \mathcal{V}_i})}{\partial f_v} = 0$ and $\frac{\partial^2 C_i^{\boldsymbol{\pi}_k}((f_v)_{v \in \mathcal{V}_i}, (m_v^*)_{v \in \mathcal{V}_i})}{\partial f_v^2} > 0$.

Considering the latency constraint $\bar{\tau}_v \geq \frac{\mathbb{E}[L_v]}{f_v}$ for function v, the smallest CPU allocation that can be chosen is $f_v = \frac{\mathbb{E}[L_v]}{\bar{\tau}_v}$. Since $\frac{\partial C_i^{\boldsymbol{\pi}_k}(\cdot)}{\partial f_v}$ is continuous, and the only value at which it is zero is f_v^*, the cost $C_i^{\boldsymbol{\pi}_k}(\cdot)$ is monotone for $f_v > f_v^*$ and for $f_v < f_v^*$. This ensures that the value in set $\mathcal{F}$ that minimizes the cost $C_i^{\boldsymbol{\pi}_k}(\cdot)$ is either $\tilde{f}_v^-$ or $\tilde{f}_v^+$, which leads to (8). □

The above lemma shows that the best response of the WDs can be computed efficiently.

B. Optimal Pricing and Reward in Steady State

We now turn to the analysis of the optimal price, under assumptions that allow analytical tractability. We will propose a solution that does not rely on these assumptions in Section IV.

Assumption 1. *WD arrivals follow a homogeneous Poisson process with intensity* Λ*. Dwell times* D_i *are exponentially distributed with mean* $1/\mu$*.*

Assumption 2. *Each WD has a single function, i.e.,* $|\mathcal{V}_i| = 1$*. Available CPU and memory allocations are* $\mathcal{F} = \{f^*\}$ *and* $\mathcal{M} = \{m^*\}$*, respectively. Thus, each WD requests CPU frequency* f^* *and memory* m^**.*

Assumption 3. *Let* $c = \lfloor F/f^* \rfloor$*. Then* $M \geq c \cdot m^*$*, i.e., the edge system is not memory constrained.*

Under Assumptions 1 to 3 we can define the system state to be $N_a + N_w$, i.e., the total number of WDs offloading or waiting. We can then model the evolution of the system state using a continuous time Markov chain [17], [18]. We consider a system in steady state, and we denote by $P_A(\boldsymbol{\pi})$ the probability that an arriving WD accepts price $\boldsymbol{\pi}$, which depends on the distribution C of reservation costs.

We are interested in computing the expected revenue $\mathbb{E}[\rho_k^\theta]$ of the operator (see (6)), which depends on the probability $P_s(\boldsymbol{\pi})$ that a WD that accepts the offered price $\boldsymbol{\pi}$ actually receives service before departing, to be computed next.

Lemma 2. *Let* $P_A(\boldsymbol{\pi}) > 0$*. A user that arrives to the system receives service with probability*

$$P_s(\boldsymbol{\pi}) = P_A(\boldsymbol{\pi}) \Big(\sum_{l=0}^{c-1} \frac{l+1-c}{l+1} p_l + c\mu \frac{1 - e^{-\frac{\Lambda P_A(\boldsymbol{\pi})}{\mu}}}{\Lambda P_A(\boldsymbol{\pi})} \Big), \quad (11)$$

where

$$p_l = e^{\frac{-\Lambda P_A(\boldsymbol{\pi})}{\mu}} \Big(\frac{\Lambda P_A(\boldsymbol{\pi})}{\mu} \Big)^l \frac{1}{l!}. \quad (12)$$

Proof. We are to compute the probability of acquiring service in an $M/M/c$ queue with reneging and balking, for exponentially distributed reneging time. Our proof is a generalization of the analysis for the $M/M/1$ queue provided in [19]–[21].

For the analysis, observe that total number $N_a + N_w$ of WDs in the system can be modeled by an M/M/∞ queue with arrival rate $\Lambda P_A(\boldsymbol{\pi})$ and service intensity $l\mu$, where l is the position occupied by the WD. The steady state probability p_l that $N_a + N_w = l$ is thus given by (12) [22], [23].

Focusing on the $M/M/c$ queue, let t_i^w denote the waiting time of WD i, i.e., the time between its entrance into the edge service area and the instant in which it gets the requested resources and can start to offload. We can then express the probability of receiving service as $P_s(\boldsymbol{\pi}) = P_A(\boldsymbol{\pi}) \cdot \mathbb{P}[t_i^w \leq D_i]$. To compute this, let P_z denote the probability that a user in the z^{th} position in the system does not leave the system before any user in front of it leaves the system. We focus on the case $z > c$ as users $z \leq c$ already receive service. Let D_z be the dwell time of the z^{th} user, then the probability that no user leaves within time y is $\mathbb{P}[\min\{D_1, \ldots, D_{z-1}\} > y] = e^{-\mu(z-1)y}$,

due to the independence of the dwell times and the memoryless property. Hence, we can write

$$P_z = \mathbb{P}[D_z > \min\{D_1, \dots, D_{z-1}\}]$$
$$= \int_0^\infty \int_0^{d_z} \mu^2 (z-1) e^{-\mu d_z} e^{-\mu(z-1)y} dy\, d(d_z) = \frac{z-1}{z}.$$

Recall that WDs in positions $1 \dots c$ are those who are currently being able to offload. We next express the probability that a WD that arrives at position $l > c$ to the system receives service during its dwell time,

$$P_l^c = \prod_{z=c+1}^{l} P_z = \prod_{z=c+1}^{l} \frac{z-1}{z} = \frac{c}{l}. \tag{13}$$

Thus, if a WD accepts price $\boldsymbol{\pi}$, the probability that it will be able to offload before it leaves is

$$\mathbb{P}[t_i^w \le D_i] = \sum_{l=0}^{c-1} p_l + \sum_{l=c}^{\infty} P_{l+1}^c p_l \tag{14}$$
$$= \sum_{l=0}^{c-1} \frac{l+1-c}{l+1} p_l + c\mu \frac{1 - e^{-\frac{P_A(\boldsymbol{\pi})\Lambda}{\mu}}}{P_A(\boldsymbol{\pi})\Lambda}, \tag{15}$$

since if a WD arrives in state $l < c$ it will get service directly, otherwise it receives service with probability P_{l+1}^c for $l \ge c$. (15) follows using algebraic manipulations and using the convergence of the infinite series.

□

Using $P_s(\boldsymbol{\pi})$ we can express the active time $\mathcal{T}_i^a$ of offloading users, which we then use for computing the expected revenue of the operator in a pricing period

Lemma 3. *Let $P_A(\boldsymbol{\pi}) > 0$ and assume the queue is in steady state. Then the expected revenue in a pricing period of length Δ is*

$$\mathbb{E}[\rho_k^\theta] = \Delta \overline{\alpha}\, \overline{n} \Big(\sum_{l=0}^{c-1} (l-c) p_l + c \Big) \Big(\frac{\overline{L}}{f^*} \Big(\pi^{\boldsymbol{f}}(f^*) + \pi^{\boldsymbol{m}}(m^*) \Big) + \pi^r \Big), \tag{16}$$

where $\overline{\alpha} = \mathbb{E}[\alpha_i]$, $\overline{n} = \mathbb{E}[n_v]$ and $\overline{L} = E[L_v]$ are the mean task generation intensity, the mean number of function invocations and the mean computational complexity, respectively.

Proof. Under Assumptions 1 and 2, the expected active time conditional on receiving service is

$$\mathbb{E}\Big[\mathcal{T}_i^a | t_i^w \le D_i, \boldsymbol{\pi}\Big] = \frac{\overline{N}_{busy}}{\Lambda P_s(\boldsymbol{\pi})} = \frac{\sum_{l=0}^{c-1}(l-c)p_l + c}{\Lambda P_s(\boldsymbol{\pi})}, \tag{17}$$

where $\overline{N}_{busy}$ is the average number of users in service (able to offload) and (17) follows from Little's Theorem. We can then rewrite the expected revenue (6) as

$$\mathbb{E}[\rho_k^\theta] = \mathbb{E}\Big[\sum_{i \in \mathcal{N}_k} \alpha_i C_i^{\boldsymbol{\pi}}(f^*, m^*) \mathcal{T}_i^a \Big| t_i^w \le D_i, \boldsymbol{\pi} \Big]$$
$$= \Lambda P_s(\boldsymbol{\pi}) \Delta \overline{\alpha}\, \overline{n} \Big(\frac{\overline{L}}{f^*} \Big(\pi^{\boldsymbol{f}}(f^*) + \pi^{\boldsymbol{m}}(m^*) \Big) + \pi^r \Big) \mathbb{E}\Big[\mathcal{T}_i^a \Big| t_i^w \le D_i, \boldsymbol{\pi} \Big]. \tag{18}$$

By substituting (17) to (18), we obtain (16). This concludes the proof. □

We now use the above for characterizing the optimal price and the achievable revenue for linear pricing.

Proposition 1. *Consider linear pricing, i.e., $\boldsymbol{\pi} = (\pi^{CPU}, \pi^{MEM}, \pi^r, 1)$, and let the reservation cost $\overline{C}_i$ be uniformly distributed on $(0, b)$, for $b > 0$. Then the expected revenue $\mathbb{E}[\rho_k^\theta]$ is jointly concave in (π^{CPU},π^{MEM}, π^r).*

Proof. We provide a proof sketch for brevity. For uniformly distributed reservation cost on $(0, b)$ we obtain $P_A(\boldsymbol{\pi}) = 1 - \overline{n} \cdot \frac{\pi^r + \frac{\overline{L}}{f^*}(f^* \pi^{CPU} + m^* \pi^{MEM})}{b}$. We substitute this into (16) and use Sylvester's criterion to show concavity [16], as follows. The first and second order principal minors of the resulting Hessian matrix are non-positive. In addition, the third order principal of the Hessian matrix is the determinant, which is always non-positive, and hence all principal minors are non-positive. Thus, the Hessian matrix is negative semi-definite, and $\mathbb{E}[\rho_k^\theta]$ is jointly concave. □

The joint concavity of the expected revenue would enable the operator to use a gradient-based iterative algorithm for finding an optimal price, assuming all parameters are known. We next provide a good initial guess for the optimal price and use it for obtaining a lower bound on the expected revenue.

Proposition 2. *Consider linear pricing $\boldsymbol{\pi} = (\pi, \pi, \pi, 1)$, and uniform reservation cost distribution. Let*

$$\pi^* = b \frac{\mathcal{L}(e^{\rho+1}) - 1}{B\rho}, \tag{19}$$

where $B = \overline{n}\Big(\frac{\overline{L}}{f^*}\Big(f^* + m^*\Big) + 1\Big)$, $\mathcal{L}(.)$ is the Lambert function, and $\rho = \frac{\Lambda}{c\mu}$. Then the expected revenue satisfies

$$\max_{\boldsymbol{\pi}} \mathbb{E}[\rho_k^\theta] \ge c \cdot \mathbb{E}[\rho_k^{\theta^*} | F = f^*, \Lambda = \Lambda/c], \tag{20}$$

where $\mathbb{E}[\rho_k^{\theta^*} | F = f^*, \Lambda = \Lambda/c]$ is the expected revenue in a system with price π^*, $F = f^*$ and arrival rate Λ/c.

Proof. We show the result by showing that (19) is the optimal price for $c = \frac{F}{f^*} = 1$. For price π and $c = 1$ the expected revenue (16), using (12), becomes

$$\mathbb{E}[\rho_k^\theta] = \Delta \overline{\alpha}\, \overline{n} \Big(1 - e^{\frac{-\Lambda P_A(\boldsymbol{\pi})}{\mu}}\Big) \Big(\frac{\overline{L}}{f^*} \Big(f^* + m^* \Big) + 1 \Big) \pi.$$

For uniformly distributed reservation cost the probability that a user accepts the price offer is $P_A(\pi) = 1 - \frac{B\pi}{b}$ where $B =$

$\overline{n}\left(\frac{\overline{L}}{f^*}\left(f^*+m^*\right)+1\right)$. To express the price that maximizes the expected revenue we use the first order optimality condition

$$\frac{\partial \mathbb{E}[\rho_k^\theta]}{\partial \pi} = \Delta\overline{\alpha}\, B\left[1 - e^{-\rho(1-\frac{B\pi}{b})}\frac{\rho B\pi}{b} - e^{-\rho(1-\frac{B\pi}{b})}\right] = 0,$$

whose solution is (19).

Consider now that the operator creates c virtual queues, one per f^* amount of resource, announces price (19), and assigns arriving users with probability f^*/F to one of the queues. This policy is suboptimal, but it provides the expected revenue (20), which is a lower bound to the optimal expected revenue. □

The above results provide insight into optimal pricing if parameters are known or can be estimated and the system is in or near steady state. Furthermore, they serve as a good heuristic even if Assumptions 1, 2 and 3 do not hold (see Section V). In the next section, we abandon Assumptions 1, 2 and 3 and we introduce a data-driven approach that learns to adapt pricing based on the workload dynamics.

IV. Data-driven Optimization

Recall that problem (7) faced by the operator is a sequential decision making problem under uncertainty. A straightforward approach would be to formulate the problem as a Markov Decision Process (MDP) and use model-free RL for solving it. This approach may work for a single edge deployment with a stationary workload, but a new policy would have to be learned for each edge deployment and learning an optimal policy requires a long exploration phase, which is detrimental for the achievable revenue.

A. Generalized Hidden Parameter MDP Formulation

To overcome this problem, we propose to follow a semi-parametric approach, formulating the operator's problem as a GHP-MDP. A GHP-MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, W, T, R, \tilde{\gamma}, P_W \rangle$, where $\mathcal{S} \subseteq \mathbb{R}^{N^s}$ and $\mathcal{A} \subseteq \mathbb{R}^{N^a}$ where $N^s, N^a \in \mathbb{N}^+$, and $\tilde{\gamma} \in (0,1)$ are the state space, the action space and the discount factor, respectively, as in a MDP. The transition function $s_{k+1} \sim T(s_{k+1}|s_k, a_k, w_g^t)$ and the reward $r_k \sim R(s_k, a_k, w_g^r)$ are, however, parameterized by w_g^t and w_g^r, which are drawn from prior P_W and is not observable. A GHP-MDP defines a class of problems; a particular problem instance (a MDP) is obtained once the parameters w_g^t, w_g^r are drawn. A GHP-MDP is different from a Partially Observable MDP (POMDP) as the state is observable, but the environment dynamics and reward are parameterized and the learning agent has to estimate the parameter based on interaction with the environment, while maximizing its reward.

We argue that this semi-parametric approach is a powerful abstraction for the considered problem, combining adaptiveness with transferability, leveraging the intuition that the state transition and the reward in the underlying queuing system can be approximated by a family of functions, parameterized by the latent variables w_g^t, w_g^r. In fact, the considered problem can be modeled as a GHP-MDP, as we show next.

Algorithm 1 Hidden Parameter Edge (HiPE) pricing algorithm

Compute weights of the BNNs $\mathcal{W}^t, \mathcal{W}^r$ using $\mathcal{G}_{tra}$
Draw $w_{g'}^t, w_{g'}^r \sim P_W$ for the new environment
Randomly initialize policy $\hat{\pi}_{g'}$
Initialize model, replay and fictional buffers $\mathcal{D}_{g'}, \mathcal{D}_{g'}^f$
for n **from** 1 **to** N_T **do**
  while t^{upd} episodes are not terminated **do**
    Take action $a \leftarrow \hat{\pi}_{g'}(s)$
    $\mathcal{D}_{g'} \leftarrow (s, a, r, s', w_{g'}^t, w_{g'}^r)$
  end while
  if $\hat{T}_{g'}$ and $\hat{R}_{g'}$ is inaccurate **then**
    TRAIN-BNN$(\mathcal{D}_{g'}, \mathcal{W}^t, \mathcal{W}^r, w_{g'}^t, w_{g'}^r)$
  end if
  if $n \,\%\, t^{upd} == 0$ **then**
    FICTIONAL-TRAIN$(\mathcal{D}_{g'}^f, \mathcal{W}^t, \mathcal{W}^r, w_{g'}^t, w_{g'}^r)$
  end if
end for

---

procedure TRAIN-BNN$(\mathcal{D}_{g'}, \mathcal{W}^t, \mathcal{W}^r, w_{g'}^t, w_{g'}^r)$
  for k **from** 0 to N_u **do**
    Update $w_{g'}^t$ using $\mathcal{D}_{g'}$
    Update $w_{g'}^r$ using $\mathcal{D}_{g'}$
    Update $\mathcal{W}^t, \mathcal{W}^r$ using $\mathcal{D}_{g'}$
  end for
end procedure

---

procedure FICTIONAL-TRAIN$(\mathcal{D}_{g'}^f, \mathcal{W}^t, \mathcal{W}^r, w_{g'}^t, w_{g'}^r)$
  for t **from** 0 to N_f episodes **do**
    Take action $a \leftarrow \hat{\pi}_{g'}(s)$
    Estimate next state $\hat{s}' \leftarrow \hat{T}(s, a, w_{g'}^t)$
    Estimate reward $\hat{r} \leftarrow \hat{R}(s, a, w_{g'}^r)$
    Store $\mathcal{D}_{g'}^f \leftarrow (s, a, \hat{r}, \hat{s}')$
    if $n \,\%\, t^u == 0$ **then**
      Update $\hat{\pi}_{g'}$ using $\mathcal{D}_{g'}^f$
    end if
  end for
end procedure

Proposition 3. *Under Assumptions 1, 2 and 3 problem* (7) *is a GHP-MDP, with state* $(N_a + N_w)$*, action* $a_k = (\pi_k^{CPU}, \pi_k^{MEM}, \pi_k^r, \gamma_k)$*, reward* $R_k^{\boldsymbol{\pi}_k} = \sum_{i \in \mathcal{N}_k} C_{i,\Sigma}^{\boldsymbol{\pi}_k}((f_v)_{v \in \mathcal{V}_i}, (m_v)_{v \in \mathcal{V}_i})$ *and latent parameter* $w = f(D_i, \Lambda(t), \alpha_i, \overline{C}_i, c)$.

Proof. The system with the considered state representation is Markovian due to Assumptions 1, 2 and 3. The transition function depends on the state, the action, and the system parameters $(D_i, \Lambda(t), \alpha_i, \overline{C}_i, c)$ [22]. Nonetheless, the reward ρ_k^θ defined in (6) during pricing period T_k does not only depend on the state and the action, but also on past actions, and is hence not Markovian.

To provide a Markovian formulation, we can define the

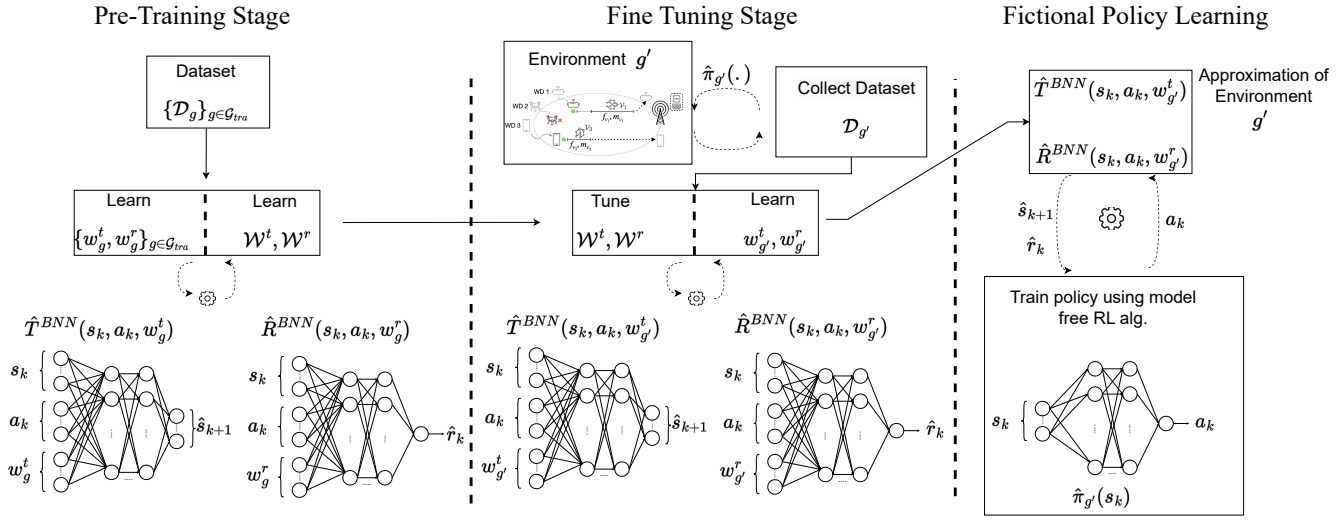

Figure 2: The workflow of HiPE showing *Pre-Training*, *Fine Tuning* and *Fictional Policy Learning* stages from left to right.

revenue from the WDs accepted during pricing period T_k as

$$R_k^{\boldsymbol{\pi}_k} = \sum_{i\in\mathcal{N}_k} C_{i,\Sigma}^{\boldsymbol{\pi}_k}((f_v)_{v\in\mathcal{V}_i}, (m_v)_{v\in\mathcal{V}_i}). \tag{21}$$

Observe that $R_k^{\boldsymbol{\pi}_k}$ only depends on the state, the action and the system parameters. Furthermore, maximizing $\lim_{K\to\infty}\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[R_k^{\boldsymbol{\pi}_k}]$ is equivalent to solving (7). Hence problem (7) is a GHP-MDP. □

B. Hidden Parameter Edge (HiPE) Pricing Algorithm

In what follows we propose a data-driven solution that leverages the above GHP-MDP formulation, called Hidden Parameter Edge Pricing Algorithm (HiPE). ***HiPE*** relies on two BNN approximators: one to approximate the state transition function and one to approximate the reward,

$$\hat{s}_{k+1} \sim \hat{T}^{BNN}(s_k, a_k, w_g^t) + \epsilon_t \tag{22}$$

$$\hat{r}_k \sim \hat{R}^{BNN}(s_k, a_k, w_g^r) + \epsilon_r \tag{23}$$

$$\epsilon_t, \epsilon_r \sim \mathcal{N}(0, \sigma_n^2), \tag{24}$$

where $\hat{T}^{BNN}(s, a, w_g^t)$ and $\hat{R}^{BNN}(s, a, w_g^r)$ are the function approximators (c.f. Fig. 2) and ϵ_t, ϵ_r are additive Gaussian noise that improve the ability of the BNNs to approximate the stochasticity of the transition and reward functions, respectively [24]. Note that the functions that approximate the state transition distribution (22) and the reward (23) are parameterized by environment instance specific latent variables w_g^t and w_g^r, respectively. The latent variables w_g^t and w_g^r serve as low dimensional representations of the dynamics of environment g. Importantly, we allow the latent variables w_g^t and w_g^r used for the two approximators to be different, which allows us to learn potentially different latent embeddings for the transition and for the reward.

Our approach is different from HiP-MDP formulations [25]–[27], where the reward function is assumed to be known (and hence need not be approximated), and from the solution in [28], where the policy is learnt using trajectory sampling combined with cross-entropy minimization, introducing extra computational complexity. Instead, we propose a novel approach for computing a policy based on the GHP-MDP formulation via model-free deep RL based on the two BNN approximators.

Alg. 1 shows the procedure for learning a policy using the proposed approach. Learning makes use of a set $\mathcal{G}_{tra}$ of pre-training problem instances, which can be simulated problem instances. For every problem instance $g \in \mathcal{G}_{tra}$, we collect a replay buffer $\mathcal{D}_g$ with the observed transitions and rewards (s, a, s', r). Using the global replay buffer $\mathcal{D} = \bigcup_{g\in\mathcal{G}_{tra}} \mathcal{D}_g$, the BNN parameters $\mathcal{W}^t, \mathcal{W}^r$ and the environment specific latent embeddings $w_g^t, w_g^r,\ \forall g \in \mathcal{G}_{tra}$ are learned (Line 1 in Alg. 1). Observe that the BNN weights $\mathcal{W}^t$ and $\mathcal{W}^r$ are learnt using $\mathcal{D}$, while the latent embeddings w_g^t, w_g^r are learned based on $\mathcal{D}_g$. This ensures that the BNN weights capture the general dynamics of the environment, and the latent embeddings w_g^t and w_g^r capture the environment specific parameters. We refer this stage as the *Pre-Training* stage (see Fig. 2).

Upon deployment in a new environment instance g', the algorithm aims at determining the latent embeddings $w_{g'}^t$ and $w_{g'}^r$, based on a few observations $(s, a, s', r) \in \mathcal{D}_{g'}$ taken during t^{upd} episodes using a random policy $\hat{\pi}_{g'}(.)$ (lines 6-9). For learning $w_{g'}^t$ and $w_{g'}^r$, the algorithm initializes the latent embeddings randomly, then it minimizes the α-divergence [29] of the observed transitions and rewards $(s, a, s', r) \in \mathcal{D}_{g'}$ and the ones predicted by $\hat{T}^{BNN}(s, a, w_{g'}^t)$ and $\hat{R}^{BNN}(s, a, w_{g'}^r)$, respectively, which were trained in the *Pre-Training* phase. We refer to this stage as the *Fine Tuning* stage (see Fig. 2) (lines 10-12).

Finally, HiPE uses $\hat{T}^{BNN}(s, a, w_{g'}^t)$ and $\hat{R}^{BNN}(s, a, w_{g'}^r)$ parameterized by $w_{g'}^t$ and $w_{g'}^r$ for generating *fictional* transitions and rewards, which are collected in a replay buffer $\mathcal{D}_{g'}^f$. The *fictional* transitions and rewards serve as the environment on which the RL agent $\hat{\pi}_{g'}$ is trained using a model free RL algorithm (lines 13-15). We refer to this stage as the *Fictional*

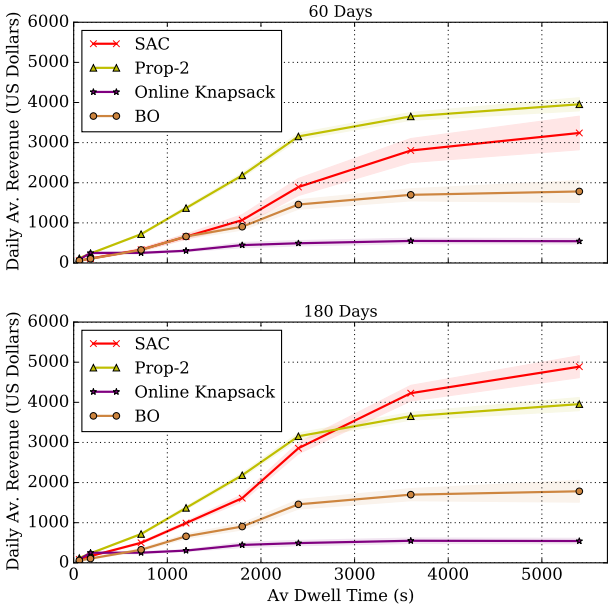

Figure 3: Average daily revenue (US Dollars) vs. average dwell time under homogeneous resource allocation.

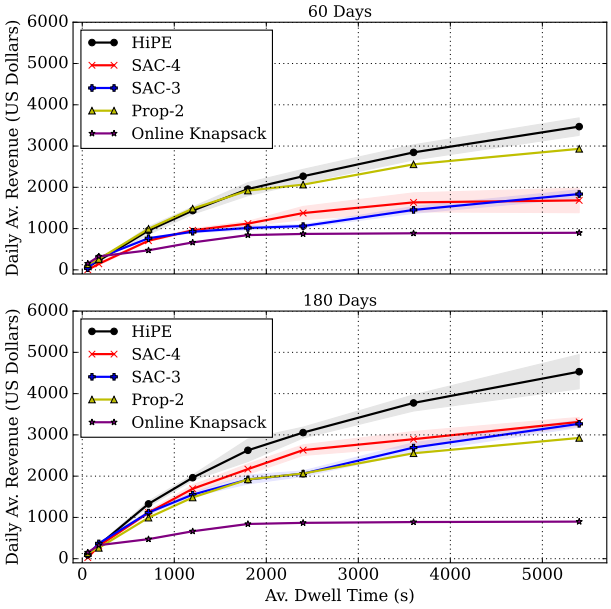

Figure 4: Average daily revenue (US Dollars) vs. average dwell time under heterogeneous resource allocation.

Policy Learning stage. The learnt policy $\hat{\pi}_{g'}$ is then directly used in the environment g'.

V. Numerical Results

We use simulations on synthetic and on measured traces for evaluating the performance of the steady state approximation and of the proposed *HiPE* algorithm.

A. Evaluation methodology

For the evaluation, we consider an EC system with compute capacity $F = 120$ GHz and memory capacity $M = 300$ GB as in [30], [31]. This is equivalent to having a small cluster of about 10 compact edge servers [32]. The edge operator offers CPU allocations $\mathcal{F} = \{1, 1.3, 1.6, ..., 4\}$ GHz, and memory allocations $\mathcal{M} = \{1, 1.2, 1.4, ..., 3\}$ GB. We consider that the memory requirements of task $m^*_{v_i}, \forall v_i \in \mathcal{V}_i$ for all i are chosen uniformly at random from $\mathcal{M}$.

The number of functions $|V_i|$ of each user i is drawn from a discrete uniform distribution on $\{1, \ldots, 4\}$, and the average number of invocations n_v is uniformly distributed on $[1, 3]$. The average computation complexity L_v is exponentially distributed with mean 0.01 GCycles, and we use typical computation density values of 30 KCycles/bit [33, Table II], for which offloading may be more convenient than local computing [34, Fig. 4]. The maximum number of users active in a cell is 120. Delay bounds $\overline{\tau}_v$ are distributed uniformly at random in [5, 10] ms (corresponding to, e.g., augmented reality applications [1]), and the reservation cost $\overline{C}_i$ is drawn either from a uniform distribution on $[0, 0.001]$\$ or from a truncated Gaussian distribution on $[0, 0.001]$\$, which are typical costs for Amazon serverless offerings.

We use synthetic and measured traces of user arrivals for the evaluation. For synthetic traces, we use two dwell time distributions: deterministic and exponential. For measured traces, we choose 3 cells from the Greater Shanghai metropolitan area traces [35]–[37]: i) Cell #1 is in the city center with high load, ii) Cell #2 is in a suburban area with medium load and iii) Cell #3 is in a rural area with low load.

We consider three variants of the pricing model: (i) univariate pricing ($\pi_k^{CPU} = \pi_k^{MEM} = \pi_k^r = \pi_k, \gamma_k = 1$), (ii) multivariate linear pricing ($\pi_k^{CPU}, \pi_k^{MEM}, \pi_k^r, \gamma_k = 1$) and (iii) non-linear pricing ($\pi_k^{CPU}, \pi_k^{MEM}, \pi_k^r, \gamma_k \geq 1$). We set the length of the pricing periods to $\Delta = 1$ hour.

For the GHP-MDP we use $s_k = (\rho_{CPU}, \rho_{MEM}, k) \in \mathcal{S} \subseteq [0, 1]^2 \times \mathbb{N}^+$ as the state, where ρ_{CPU} and ρ_{MEM} are the CPU and memory utilization, respectively, and k is the index of pricing period T_k in a day (as the length of pricing period T_k is 1 hour, we have $k \in \{0, 1, 2, \ldots, 23\}$). The choice of the state is motivated by that ρ_{CPU} and ρ_{MEM} capture the congestion on computation and memory resources, which in turn determine the reward in a pricing period T_k. The use of the time index of the period in a day is motivated by the periodicity of user arrivals observed in real data. We found this to be a concise state representation that allows fast convergence. We define the action to be $a_k = (\pi_k^{CPU}, \pi_k^{MEM}, \pi_k^r, \gamma_k) \in \mathcal{A} \subseteq \mathbb{R}_+^4$, and the reward $R(s_k, a_k, w_g^r)$ as the revenue collected during the pricing period.

To approximate the transition and reward dynamics for an environment instance g, we used a 2-layer Bayesian neural network architecture; each layer contains 25 neurons with Gaussian priors on the weights. We used the BNN hyperparameters reported in [25], except the learning rate, which

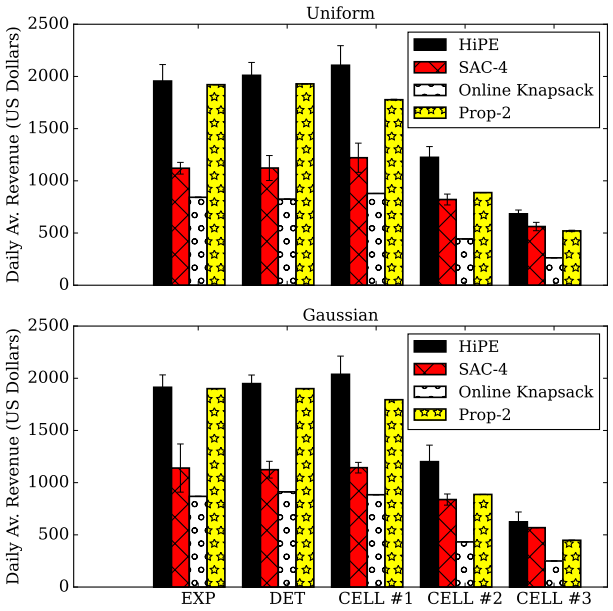

Figure 5: Average daily revenue for two synthetic dwell time distributions ($\mathbb{E}[D_i] = 1800$ sec.) and trace-based distributions, and for two reservation cost distributions (Uniform and truncated Gaussian) for 60 days of training.

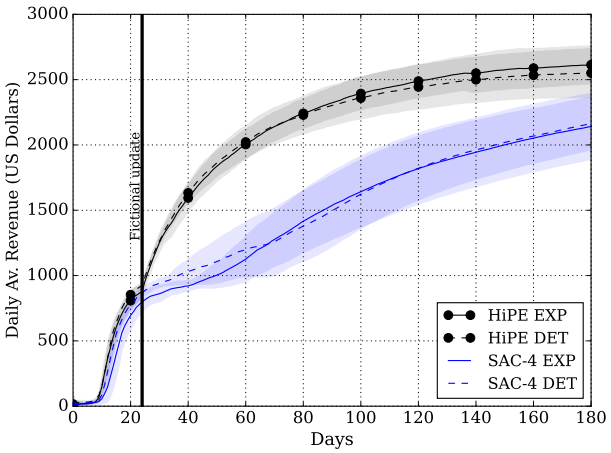

Figure 6: Learning curves of SAC and HiPE for synthetic traces with exponential and deterministic dwell time distributions.

we set to 0.0005 for both BNNs. For pre-training the BNNs, we collected $\sim$ 400.000 transition samples from synthetic traces with exponentially distributed dwell times with mean $\{180, 720, 1200, 1800, 2400, 3600, 5400\}$. We used $\mathbb{R}^5$ for the latent parameter space. Increasing the dimension of the latent variables increases the computational complexity, whereas choosing low dimensionality results in limited representation of the environment instances g, which negatively affects transferability. We learn the latent parameters w_g^t and w_g^r and the network weights by minimizing the α-divergence using ADAM with $\alpha = 1$ [24].

During the *Fictional Policy Learning* stage, we used Soft Actor-Critic (SAC) to learn the policy (Line $26-36$) and we used $t^{upd} = 25$ days to tune the BNNs (weigths and latent variables) in the unseen environment, where we assume one episode is 1 day, and applied $N_f = 60$ days of fictional updates. We used the default hyper-parameters in the stable baselines library for SAC [38]. Tuning the BNN weights in the *Fine Tuning Stage* took on the order of minutes on an i9-10900K Intel processor, which is negligible compared to the training time of the BNNs in the *Pre-Training Stage*, which was in the order of hours.

We use four baselines for comparison. The first baseline is the pricing scheme for the online Knapsack problem proposed in [15], which is a user-specific pricing scheme based on the instantaneous system load. The second baseline is using Bayesian Optimization (BO) for each time index of a day, as proposed in [39], where we assign an agent to each time index of a day, and each agent maximizes the expected revenue using a Gaussian process approximation. The third baseline is based on the bound in Proposition 2 where we use (19) as a starting price and implement a gradient ascent algorithm (labeled as Prop-2 in the figures). The fourth baseline is a model-free RL agent using the SAC algorithm [40].

B. Operator Revenue

Fig. 3 shows the daily average revenue of the edge operator as a function of average dwell time over a period of 60 and 180 days based on synthetic traces that satisfy Assumptions 1, 2, and 3 (homogeneous resource allocation): $\overline{f} = 2$ GHz, $\overline{m} = 1$ GB, $c = 10$. We use the univariate pricing model and uniform reservation cost distribution to evaluate the accuracy of the analytical approximation (Proposition 2). We observe that over 60 days the analytical approximation outperforms all schemes, as it does not have to learn the system parameters. Over 180 days it performs best for low average dwell times, when the mixing times are short and the steady state approximation is accurate, but it performs slightly worse than *SAC* for long dwell times. The BO scheme, which learns a price for each pricing period, fails to find an effective pricing policy. We can also observe that the online knapsack algorithm does not work well either; it consistently offers high prices with increasing system load, resulting in the rejection of too many WDs (see Sec. V-E). These results show that pricing based on the steady state approximation works rather well when the modeling assumptions are satisfied, but its advantage is mainly due to that it does not have to learn the system parameters. We now turn to more complex scenarios.

Fig. 4 shows the daily average revenue of the operator as a function of the average dwell time for 60 days and for 180 days, for heterogeneous resource allocation and for traces that do not satisfy Assumptions 1, 2, 3. The figure shows results for multivariate (SAC-3) and non-linear (SAC-4) pricing using *SAC* (the numbers denote the dimension of the action of the *SAC*), to assess the advantage of non-linear

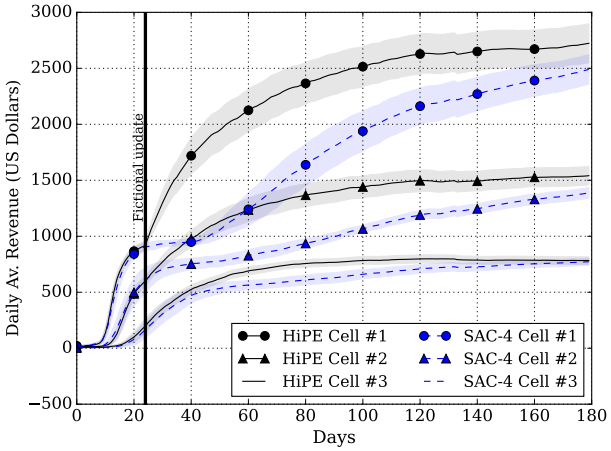

Figure 7: Learning curves of SAC and HiPE for Cells #1, #2, and #3.

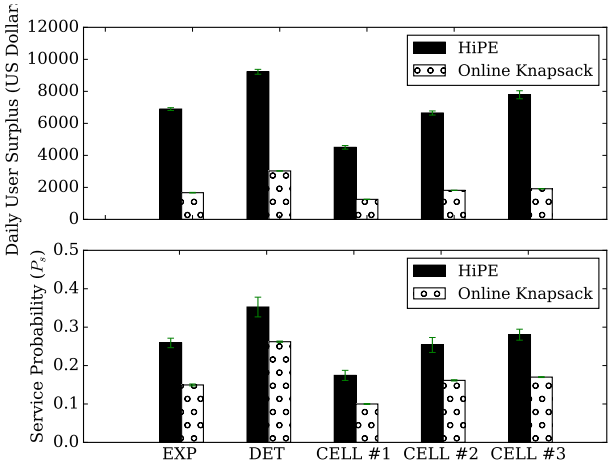

Figure 8: Consumer surplus and P_s for two synthetic dwell time distributions ($\mathbb{E}[D_i] = 1800$ sec.) and trace-based distributions, for uniform reservation cost distribution.

pricing. The figure shows that the proposed *HiPE* pricing scheme outperforms all baselines, with an increasing margin as the average dwell time increases, providing up to 80% higher revenue than *SAC*-4. Interestingly, over 60 days even the analytical approximation outperforms *SAC*, even though it uses univariate pricing and the average resource requirement of the WDs, indicating that *SAC* suffers from slow learning, which is detrimental to the average revenue, unlike the proposed *HiPE* pricing algorithm. Comparing linear (*SAC*-3) and non-linear (*SAC*-4) pricing we can observe that non-linear pricing is most beneficial for moderate average dwell times, and allows up to 50% higher revenue than linear pricing, as WDs tend to request less resources and resource intensive functions can be charged more aggressively. We can also observe that the online knapsack algorithm exhibits consistently low revenue similar to the case of homogeneous resource allocation.

C. Sensitivity Analysis

Fig. 5 shows the daily average revenue for synthetic traces using exponentially and deterministically distributed dwell times with mean $\mathbb{E}[D_i] = 1800$ sec, and for real traces based on Cells $\#1$, $\#2$ and $\#3$; reservation costs follow uniform and truncated Gaussian distributions. The figure shows that even though the *HiPE* algorithm was pre-trained on synthetic traces, the data collected within the initial 25 days of the evaluation is sufficient to fine-tune the BNNs for a previously unseen environment, achieving superior performance compared to all baselines. The figure shows subtle revenue differences among the results obtained using different dwell time and reservation cost distributions, indicating that the revenue is predominantly influenced by the average dwell time. Comparing the performance achieved on the real traces from Cell $\#1$, $\#2$ and $\#3$, we observe that the highest gain is achieved in Cell $\#1$, which has the highest load. This emphasises the difficulty of learning an effective policy using a model-free approach in such scenarios and highlights the advantages of the proposed *HiPE* algorithm. Overall, Fig. 5 shows the robustness and adaptability of the proposed *HiPE* algorithm, which make it well-suited for real-world environments.

D. Learning Curves

Fig. 6 and Fig. 7 show the daily average revenue achieved using the *HiPE* and *SAC*-4 pricing schemes, for synthetic traces with exponentially and deterministically distributed dwell times and for real traces, respectively.

The figures show that *HiPE* learns significantly faster than *SAC*; taking approximately 60 days to achieve the average revenue that *SAC* achieves after 180 days of learning. The learning curves also confirm that *HiPE* is most advantageous under high traffic load (Cell #1), which has the highest potential revenue. The ability of *HiPE* to learn fast makes it particularly appealing for real deployments, and emphasises the necessity to use transfer learning for pricing in EC.

E. Consumer Surplus and Service Probability

Finally, we consider the consumer surplus, which correspond to the added value of the edge service as perceived by the WDs (the average of the difference of the reservation cost and the actual cost of using the edge service) and the probability P_s of a WD receiving service, i.e., the fraction of WDs served. Fig. 8 shows the results obtained for synthetic traces with exponentially and deterministically distributed dwell times with mean $\mathbb{E}[D_i] = 1800$ seconds and for real traces. The figure shows that the *HiPE* pricing algorithm yields up to 5 times higher consumer surplus and up to 2 times higher probability of receiving service compared to the baseline, online knapsack. This shows that the proposed *HiPE* pricing algorithm is not only superior in terms of operator revenue but it is also preferable from the perspective of WDs in terms of the added value received. In technical terms, our proposed approach effectively addresses admission control through pricing and at the same time offers a favorable edge service for users, combining adaptivity with transparency.

VI. Related Work

Several approaches have been proposed for the network operator to allocate its limited resources at the edge to different tenants. Auctions [41]–[43] lack transparency, as the price at which resources are allocated is not announced in advance, but it is determined by the set of bids. Similar to our setting, in [44] users queue if all EC resources utilized, and the operator needs to find a trade off between accepting more users, thus increasing revenue, and avoiding long waiting times, affecting user Quality of Service (QoS). However, [44] assumes that the operator uses admission control by suggesting users to join or balk, for given prices. We instead explore this trade-off by means of dynamic pricing, which can clearly provide higher revenue than just admission control.

Pricing is often studied via game theory (Stackelberg [45]–[48] or coalitional games [49]) where the network operator sets prices and WDs take offloading decisions. In [45], [46], [49] and [48] calculating the optimal price requires prior information about EC traffic characteristics, which is unrealistic. More realistic is the setting of [50], where the operator has incomplete information, i.e., it only knows the distribution of WDs' traffic characteristics. However, in reality, not even such distributions are known, and they would vary over time. In [47] pricing decisions are taken under the assumption that if too many users use an edge server, it will fail and no one can use it anymore. Linear pricing is learned via RL in [51] for stationary traffic, which our proposed solution outperforms.

Recently proposed dynamic pricing schemes and data driven approaches are based on learning and require no a-priori knowledge about user traffic [7], [17], [52], but suffer from three main limitations. First, the training phase is very long, which makes these approaches infeasible in practice, since training requires exploration of prices, which can lead to revenue loss for a long period of time. The "Spot" pricing proposed in [52] requires training on 10^4 servers, while the number of training epochs (i.e., days) is 400 in [17, Fig.1]. The number of training epochs are not reported for the RL agent of [7] and the Multi-agent RL of [53] (which also performs price discrimination). Second, pricing is linear, which we show to result in reduced revenue in EC. Third, the RL algorithm in [17] does not include the current occupancy of edge resources into the state. Therefore, the algorithm essentially learns a repetitive workload, whose dynamic is the same from one day to another and cannot adapt to unseen workloads, which our proposed approach is capable of.

Different from previous works, we solve the task offloading and pricing problem under a dynamic workload considering a non-linear pricing scheme. We propose two pricing policies, one based on a steady state approximation, and one based on a transfer learning approach that makes use of a novel GHP-MDP formulation of the pricing problem. To the best of our knowledge, ours is the first work to propose near-optimal non-linear dynamic pricing schemes for edge offloading.

VII. Conclusion

In this work, we have considered the problem of pricing in serverless EC under dynamic workloads. We formulated the problem of maximizing the revenue of the operator as a sequential decision making problem under uncertainty. We showed that, under Markovian assumptions, a pricing policy can be obtained analytically that serves as a lower bound for the operator revenue. We then showed that the problem can be cast as a GHP-MDP and proposed a dual BNN approximator as a solution. The proposed solution is a form of transfer learning; after pre-training on synthetic traces, it adapts fast to previously unseen workloads. Our results show that the proposed solution accelerates learning and achieves superior performance compared to the state of the art, on par with the steady state approximation, but without the need for prior information about the parameters of the system. Furthermore, our results show that non-linear pricing models could benefit edge deployment, as they encourage users to request computational resources sparingly, and thereby effectively increasing the number of concurrent users that can be served.

VIII. Acknowledgment

This work was partly funded by the Vinnova Center for Trustworthy Edge Computing Systems and Applications (TECoSA), the Swedish Research Council (project 2020-03860), the European Action Scheme for the Mobility of University Students (ERASMUS) and the French Embassy in Sweden | French Institute of Sweden (SFVE-A program).

References

[1] A. Ben-Ameur, A. Araldo, and F. Bronzino, "On the deployability of augmented reality using embedded edge devices," in *Proc. of IEEE CCNC*, 2021.

[2] N. Mohan, L. Corneo, A. Zavodovski, S. Bayhan, W. Wong, and J. Kangasharju, "Pruning edge research with latency shears," in *Proc. of ACM HotNets*, 2020.

[3] "Edge solutions," in *AT&T*. [Online]. Available: https://www.business.att.com/categories/att-edge-solutions.html

[4] R. Xie, Q. Tang, S. Qiao, H. Zhu, F. R. Yu, and T. Huang, "When serverless computing meets edge computing: Architecture, challenges, and open issues," *IEEE Wireless Communications*, vol. 28, no. 5, pp. 126–133, 2021.

[5] W.-T. Tsai and G. Qi, "DICB: Dynamic intelligent customizable benign pricing strategy for cloud computing," in *Proc. of IEEE International Conference on Cloud Computing*, 2012.

[6] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *Proc. of International Conference on Autonomic Computing*, 2010.

[7] F. Lyu, X. Cai, F. Wu, H. Lu, S. Duan, and J. Ren, "Dynamic pricing scheme for edge computing services: A two-layer reinforcement learning approach," in *Proc. of IEEE/ACM IWQoS*, 2022.

[8] X. Wang, J. Ye, and J. C. S. Lui, "Decentralized scheduling and dynamic pricing for edge computing: A mean field game approach," *IEEE/ACM ToN*, vol. 31, no. 3, pp. 965–978, 2023.

[9] K. Choi, R. Soma, and M. Pedram, "Off-chip latency-driven dynamic voltage and frequency scaling for an mpeg decoding," in *Proc. of Annual Design Automation Conference*, 2004.

[10] F. Tütüncüoğlu and G. Dán, "Optimal service caching and pricing in edge computing: a Bayesian Gaussian process bandit approach," *IEEE Trans. on Mobile Computing*, vol. 23, pp. 705–718, Jan. 2024.

[11] ——, "Optimal pricing for service caching and task offloading in edge computing," in *Proc. of IFIP/IEEE WONS*, 2022.

[12] F. Guillemin and V. Q. Rodriguez, "Evaluating the impact of tower companies on the telecommunications market," in *Proc. of IEEE Conference on Innovation in Clouds, Internet and Networks and Workshops*, 2021.

[13] M. Bilal, M. Canini, R. Fonseca, and R. Rodrigues, "With great freedom comes great opportunity: Rethinking resource allocation for serverless functions," in *Proc. of ACM European Conference on Computer Systems*, 2023.

[14] H. Qiu and T. Li, "Auction method to prevent bid-rigging strategies in mobile blockchain edge computing resource allocation," *Future Generation Computer Systems*, vol. 128, pp. 1–15, 2022.

[15] Z. Zhang, Z. Li, and C. Wu, "Optimal posted prices for online cloud resource allocation," in *Proc. of ACM POMACS*, 2017.

[16] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1990.

[17] S. Chen, L. Li, Z. Chen, and S. Li, "Dynamic pricing for smart mobile edge computing: a reinforcement learning approach," *IEEE Wireless Communications Letters*, vol. 10, no. 4, pp. 700–704, 2021.

[18] B. Baek, J. Lee, Y. Peng, and S. Park, "Three dynamic pricing schemes for resource allocation of edge computing for iot environment," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4292–4303, 2020.

[19] C. Ancker and A. Gafarian, "Queueing with impatient customers who leave at random," *Journal of Industrial Engineering*, 1962.

[20] B. Han *et al.*, "A utility-driven multi-queue admission control solution for network slicing," in *Proc. of IEEE INFOCOM*, 2019.

[21] M. Dai, L. Luo, J. Ren, H. Yu, and G. Sun, "PSACCF: Prioritized online slice admission control considering fairness in 5G/B5G networks," *IEEE Trans. on Network Science and Eng.*, vol. 9, no. 6, 2022.

[22] V. G. Kulkarni, *Modeling and analysis of stochastic systems, 3rd ed.* CRC Press, 2016.

[23] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, *Fundamentals of queueing theory*. John Wiley & Sons, 2018.

[24] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft, "Learning and policy search in stochastic dynamical systems with Bayesian Neural Networks," in *Proc. of ICLR*, 2017.

[25] T. W. Killian *et al.*, "Robust and efficient transfer learning with hidden parameter Markov decision processes," *Proc. of NeurIPS*, 2017.

[26] F. Doshi-Velez and G. Konidaris, "Hidden parameter Markov decision processes: A semiparametric regression approach for discovering latent task parametrizations," in *Proc. of IJCAI*, 2016.

[27] F. Tütüncüoğlu and G. Dán, "Sample-efficient learning for edge resource allocation and pricing with BNN approximators," in *in Proc. of IEEE INFOCOM Workshops (ICCN)*, 2024, pp. 37–42.

[28] C. Perez, F. Petroski Such, and T. Karaletsos, "Generalized hidden parameter MDPs: Transferable model-based RL in a handful of trials," in *Proc. of AAAI*, 2020, pp. 5403–5411.

[29] J. Hernandez-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernandez-Lobato, and R. Turner, "Black-box alpha divergence minimization," in *Proc. of ICML*, 2016.

[30] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE Transactions on Mobile Computing*, 2019.

[31] ——, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. of IEEE INFOCOM*, 2017.

[32] "Compact edge server specifications," 2023. [Online]. Available: https://www.electronics-lab.com/seeed-studio-reserver-a-mini-edge-server-for-high-performance-computing-applications/

[33] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, 2015.

[34] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Transactions on Communications*, 2019.

[35] Y. Li, A. Zhou, X. Ma, and S. Wang, "Profit-aware edge server placement," *IEEE Intenet of Things Journal*, 2022.

[36] M. K. Kasi, S. Abu Ghazalah, R. N. Akram, and D. Sauveron, "Secure mobile edge server placement using multi-agent reinforcement learning," *Electronics*, 2021.

[37] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE TMC*, 2019.

[38] A. Hill et al., "Stable baselines," 2018. [Online]. Available: https://github.com/hill-a/stable-baselines

[39] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proc. of ICML*, 2010.

[40] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. of ICML*, 2018.

[41] Y.-H. Hung, C.-Y. Wang, and R.-H. Hwang, "Combinatorial clock auction for live video streaming in mobile edge computing," in *Proc. of IEEE INFOCOM WKSHPS*, 2018.

[42] L. Yang, H. Zhang, X. Li, H. Ji, and V. C. Leung, "A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing," *IEEE/ACM ToN*, vol. 26, no. 6, pp. 2762–2773, 2018.

[43] T. Bahreini, H. Badri, and D. Grosu, "Mechanisms for resource allocation and pricing in mobile edge computing systems," *IEEE TPDS*, vol. 33, no. 3, pp. 667–682, 2021.

[44] S. Chen, L. Wang, and F. Liu, "Optimal admission control mechanism design for time-sensitive services in edge computing," in *Proc. of IEEE INFOCOM*, 2022.

[45] Z. Xiong, S. Feng, D. Niyato, P. Wang, and Z. Han, "Optimal pricing-based edge computing resource management in mobile blockchain," in *Proc. of IEEE ICC*, 2018.

[46] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A Stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Generation Computer Systems*, vol. 108, pp. 273–287, 2020.

[47] G. Mitsis, E. E. Tsiropoulou, and S. Papavassiliou, "Price and risk awareness for data offloading decision-making in edge computing systems," *IEEE Systems Journal*, 2022.

[48] R. Roostaei, Z. Dabiri, and Z. Movahedi, "A game-theoretic joint optimal pricing and resource allocation for mobile edge computing in noma-based 5g networks and beyond," *Computer Networks*, vol. 198, p. 108352, 2021.

[49] T. Zhang, "Data offloading in mobile edge computing: A coalition and pricing based approach," *IEEE Access*, vol. 6, pp. 2760–2767, 2017.

[50] J. Yan, S. Bi, L. Duan, and Y.-J. A. Zhang, "Pricing-driven service caching and task offloading in mobile edge computing," *IEEE Trans. on Wireless Comm.*, vol. 20, no. 7, pp. 4495–4512, 2021.

[51] L. Li, M. Siew, and T. Q. Quek, "Learning-based pricing for privacy-preserving job offloading in mobile edge computing," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2019.

[52] Z. Tang, F. Zhang, X. Zhou, W. Jia, and W. Zhao, "Pricing model for dynamic resource overbooking in edge computing," *IEEE Transactions on Cloud Computing*, 2022.

[53] P. Wang, B. Di, L. Song, and N. R. Jennings, "Multi-layer computation offloading in distributed heterogeneous mobile edge computing networks," *IEEE Trans. on Cognitive Comm. and Netw.*, vol. 8, no. 2, pp. 1301–1315, 2022.