Offloading Augmented Reality Tasks with Smart Energy Source-Aware Algorithms at the Edge

Francesco Spinelli
francesco.spinelli@imdea.org
IMDEA Networks Institute &
Universidad Carlos III de Madrid
Madrid, Spain

Antonio Bazco-Nogueras
antonio.bazco@imdea.org
IMDEA Networks Institute
Madrid, Spain

Vincenzo Mancuso
vincenzo.mancuso@imdea.org
IMDEA Networks Institute
Madrid, Spain

ABSTRACT

The development of novel use cases in beyond-5G and 6G networks will rely, among other aspects, on the availability of computing resources at the edge, therefore enabling the realization of applications that are both computationally demanding and latency constrained, such as Mobile Augmented Reality (MAR). Indeed, due to end devices' intrinsic constraints on computation capabilities and battery, newer MAR applications require offloading their most demanding tasks. However, the constrained nature of edge resources implies that these tasks should be carefully allocated at the edge network in order to guarantee satisfactory Quality of Experience to end-users. In this context, we analyze the edge operator's resource allocation to support the energy-aware offloading of MAR tasks at the edge of the cellular network with the goal of not only maximizing service acceptance (i.e., revenue), but also optimizing the operator's business utility, which depends on its carbon footprint and the profit of operating the service. We leverage Deep Reinforcement Learning to propose an efficient model to operate the edge resource allocation that can adapt to different utilities.

CCS CONCEPTS

• **Computing methodologies** → **Planning and scheduling**; • **Networks** → **Cloud computing**; **Network resources allocation**.

KEYWORDS

Mobile Augmented Reality, Deep Reinforcement Learning, Carbon Footprint, Green Energy, Edge Computing

ACM Reference Format:
Francesco Spinelli, Antonio Bazco-Nogueras, and Vincenzo Mancuso. 2023. Offloading Augmented Reality Tasks with Smart Energy Source-Aware Algorithms at the Edge. In *Proceedings of the Int'l ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM '23), October 30-November 3, 2023, Montreal, QC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3616388.3617523

1 INTRODUCTION

One of the key enablers that will allow network operators to realize the envisioned innovative use cases for the next generation of mobile networks is the deployment of computing capabilities at the network edge. Such a need is due to the fact that mobile users are increasingly interested in low-latency applications that are computationally demanding while requiring a great amount of bandwidth resources. This combination prevents cloud computing from providing this service [4], since it is unable to provide round-trip latency of few milliseconds, and the network core would be easily congested. In beyond-5G networks, the Multi-Access Edge Computing (MEC) paradigm [10] places computing nodes at the edge of the cellular network, enabling new disruptive low-latency use cases. Among those use cases, we highlight Extended Reality (XR) applications [1], which cover under their umbrella both Mobile Augmented Reality (MAR) and Virtual Reality (VR) applications. While the network support for the latter will be challenging even for 6G networks [14], MAR applications are becoming widespread among end-users thanks to the development of mobile equipment: a recent Huawei report [8] indicates that by 2026 the MAR market will generate over $30B in revenue, lead by social apps and AR games. However, this will only be possible with the help of edge servers to offload at least partially the computation of MAR tasks [1], since many of these devices will be battery-constrained. Deploying networks that are technically capable of supporting such demanding applications (e.g., with edge computing resources) is an important challenge, but there is an even more crucial aspect to eventually see these systems deployed in real networks: sustaining the required deployment has to be profitable for operators. One manner to provide income to network operators to compensate for the large capital expenditure (CAPEX) required to deploy a MEC system is the business model based on leasing edge resources to service providers or to users on a pay-per-use model [23].

At the same time, there is a growing concern about the energy consumption that these computationally demanding applications can bring forth [1, 24], with a particular interest in reducing the overall carbon footprint of infrastructures, including communication networks.[1] For network providers, the deployment of computing resources at the edge will increase both the overall economic costs and the carbon footprint, which may in turn translate into bigger economic costs. A solution to deal with this problem is to sustain edge nodes with renewable energy sources since there already exists equipment that can be installed in base stations and edge nodes [25]. In this work, we try to answer one of the main questions arising on the topic of how to realize XR applications in next-generation communication networks: "How to distribute

MSWiM '23, October 30-November 3, 2023, Montreal, QC, Canada

ACM ISBN 979-8-4007-0366-9/23/10…$15.00
https://doi.org/10.1145/3616388.3617523

[1] See, for example, the United Nations vision in https://www.un.org/sustainabledevelopment/infrastructure-industrialization, accessed 2023/06/12.

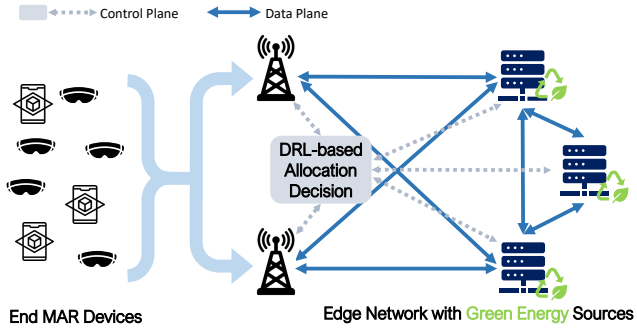


Figure 1: Edge scenario where MAR devices offload their computation to an edge network with migration capabilities.

computation and data between different components in future XR systems?" [15], which is crucial due to the limited available resources at the edge both in computing and energy terms [26].

In particular, we study how to allocate and migrate MAR tasks in an edge network, where edge nodes have a variable amount of renewable energy. Our objective is not only to maximize the operator's profit, but also to find a compromise between profit and carbon footprint, with the ultimate goal of making an edge network sustainable in both costs and energy consumption. We provide a Deep Reinforcement Learning (DRL)-based algorithm to propose a smart model for the allocation and migration of MAR tasks. The main contributions of the paper are:

- This is the first work considering both the tasks' migration and the awareness of variable renewable energy at the edge.
- We optimize both profit and a weighted fair utility to compromise between profit and sustainability.
- We propose a heuristic and a DRL algorithm, which is able to dynamically allocate and migrate jobs according to the presence of renewable energy. We evaluate the algorithms through simulations with different loads, costs, etc., which show that our DRL model outperforms the benchmarks and is able to adapt to different utility expressions.

2 RELATED WORK

The problem of allocating MAR tasks[2] in an edge scenario has been extensively studied in the literature from different perspectives. For example, [3] studies this problem on a multipath edge network, Ren *et al.* [22] propose a three hierarchical MEC-based computation framework for supporting AR, and in [16] the authors design an edge network orchestrator trading off between low latency and accurate object analytics. There also exist works that focus on the interplay of edge offloading for MAR tasks and energy efficiency with the use of ML techniques, which are consequently closer to our scope. For instance, Chen *et al.* [6] minimize the energy consumption of each user when offloading MAR tasks to MEC. In a similar scenario, the authors in [19], leveraging Deep Learning techniques, propose an energy-efficient task offloading algorithm to minimize the battery consumption of devices. Always leveraging on DRL, Chen *et al.* [5] propose an AR tasks offloading scheme that maximizes the computation rate and energy efficiency in Beyond 5G systems, and Wang *et al.* [28] design an energy-aware system that enables MAR clients to dynamically change their parameters to minimize their per-frame energy consumption. Furthermore, Cheng *et al.* [7] study AR task delay and power consumption minimization, while in [30] the authors leverage DRL to reduce the overall (transmission and server computation) energy cost while meeting the latency requirements. Ahn *et al.* [2] propose a theoretical framework to improve both the resolution of offloaded frames and the energy efficiency of multiple MAR devices connected to a single MEC server. In [12], the authors configure the AR tasks offloading problem as a partially observable Markov decision process to minimize the energy consumption of mobile devices while guaranteeing the deadlines of real-time tasks. [29] and [21] provide a detailed implementation on how to integrate ETSI MEC and 5G networks with MAR. Finally, many works consider the broader topic of edge allocation for generic tasks [9, 17, 27].

Novelty and main contributions: Most of the state-of-the-art works focus on the energy efficiency of the end-user side, while almost none of them focuses on the service/operator side and, more importantly, they do not consider the impact of the availability of *intermittent renewable* energy. Indeed, this aspect can play a significant role for the operator, as nowadays non-renewable energy sources may incur exorbitant prices with high variability. In this paper, we try to fill this gap by analyzing how MAR jobs can be offloaded in an edge system dependent on both renewable and non-renewable energy and in particular focusing on how this could be sustainable for an infrastructure provider in monetary terms.

3 SCENARIO AND PROBLEM FORMULATION

A MAR application is composed of a video source, a tracker of the user's environment position, a model for object recognition in the environment, and a rendering tool that shows the augmented world on the user's display. Except for the video source, the other tasks can be offloaded to the edge network with different latency deadlines [3]. We consider the offloading of jobs associated to the processing of video frames, as done in other works [3, 16].

Our objective is to find an allocation policy that allows the edge operator to maximize its long-term utility, where *allocation* refers to both the initial job assignment and its re-allocation (migration to another node) that might be enforced during task execution.

The utility depends on two main components: *i)* the monetary profit, which has to be maximized, and *ii)* the environmental footprint, which has to be minimized. These two objectives can clearly be contrasting. Thus, besides a mere cost-revenue function, we design a function that describes the inherent trade-off between profit and environmental footprint. The function is inspired on proportional fairness [13] for the normalized versions of the two unaligned objectives identified above, as we will explain in detail.

3.1 System Model

3.1.1 Network. We consider a MEC system as illustrated in Fig. 1, where a Virtualized Network Function (VNF) is responsible for (re-)allocating resources every time slot of duration T_{TS}. Each edge node $n \in N$ is characterized by its maximum power consumption ($P_n^{(\max)}$) and its CPU capability ($\tilde{C}_n$), i.e., the maximum amount of processing cycles per time slot.

[2]From now on, we will refer to offloaded MAR tasks as jobs interchangeably.

Each MEC node is powered by renewable energy sources, which can be located on-site at the same edge node [25]. Consequently, each node has access to a variable amount of green energy that varies through time, and we assume that the amount of renewable power available at the nodes follows a generic distribution Ξ. The available green energy is variable, and the remaining power required to reach the maximum $P_n^{(\max)}$ is provided by the standard electric grid. The grid's energy source is considered to be non-renewable, since a power grid fueled by a mix of renewable and non-renewable sources would only modify the relative goodness of grid energy versus the locally acquired green energy.

Offloading MAR tasks to the edge servers prevents the user from draining its battery, and, moreover, the use of renewable energy sources at the edge implies that offloading tasks is an environmentally beneficial decision. Therefore, we consider that MAR tasks are by default offloaded to the edge network, provided that doing so does not entail a loss of Quality of Experience (QoE) for the user, e.g. by increasing the delay beyond a maximum acceptable threshold. If the user does not enjoy the required wireless channel conditions, its computation is done locally at the end-user device, and such user does not request any offloading. Hence, and due to the fact that we are interested in optimizing the use and allocation of computing and energy resources independently of how offloading requests are generated, we omit the modelling of the wireless network access. The impact of wireless access quality and congestion might be evaluated in future work. Finally, we consider that the edge servers are not located far way from each other, such that the fiber link connecting each other only introduces a few milliseconds of delay [4], which is below the MAR latency budget.

3.1.2 MAR tasks. We consider that each job corresponds to a MAR session requested by a user. We assume that each job has a duration ℓ_j measured in time slots of T_{TS} seconds, and $t_j^\star$ denotes the arrival time slot of job j. Each job has a required processing load that remains constant throughout the session, and the number of processing cycles per time slot required to compute job j (i.e., its size) is given by c_j. The job requests' arrival times follow a generic distribution Λ, and the size of the jobs follows a generic distribution Φ. If a job has been accepted, then it must be served without interruptions for the duration of the session, as it is assumed that this type of applications demand a great quality of service, and interruptions will not be tolerated by users paying a premium service.

3.1.3 Economic model. We consider that job j provides a revenue η_j if accepted, which is lost if it is interrupted or rejected. The revenue is assumed to be proportional to the duration and the requirements of the job. Thus, for a given fixed service fee $\bar{\eta}$ representing revenue per time slot per chunk of processing resources, job j's revenue is $\eta_j = \bar{\eta}\ell_j c_j$. The constant $\bar{\eta}$ already includes all the non-variable costs associated with the operation of the service. In this way, the only remaining operational expenditure (OPEX) to be taken into account is the variable cost of energy consumption. We consider that the locally generated green energy incurs no OPEX, but its availability is not guaranteed, as it fluctuates over time; conversely, the remaining energy obtained from the general power grid is acquired at a cost δ per energy unit and is always available. The power grid can contain a variable amount of green energy, and we model such an aspect by varying the cost δ, although we consider it to be fixed for the duration of each experiment because the price of energy from national grids changes at most every hour.

3.1.4 Decision variables. We denote the placement variable of job j at node n and time t as $x_{jn}^{(t)} \in \{0,1\}$, such that $x_{jn}^{(t)} = 1$ indicates that job j is being managed by node n at time slot t. The processing cycles dedicated at time t for job j are similarly denoted by $c_j^{(t)}$. We further denote by J the total amount of jobs arriving to the system. Next, we formally present our metrics of interest before introducing the optimization problem.

3.1.5 Revenue metric. Let us first define $a_j \in \{0,1\}$ as the parameter that indicates whether job j has been accepted, i.e.,

$$a_j \triangleq 1 - \prod_{n=1}^{N} \prod_{\tau=t_j^\star}^{t_j^\star+\ell_j} (1 - x_{jn}^{(\tau)}), \tag{1}$$

where $a_j = 1$ if job j is accepted and $a_j = 0$ otherwise, $t_j^\star$ is the arrival time of the job and ℓ_j its duration.

Since job j provides a revenue η_j, the total revenue obtained by the operator is $R \triangleq \sum_{j=1}^{J} \eta_j a_j$, while the maximum possible revenue, achieved only if all jobs are accepted, is $R_{\max} \triangleq \sum_{j=1}^{J} \eta_j$. From this notation, we define the *normalized* revenue $\bar{R} \in [0,1]$ as

$$\bar{R} \triangleq \frac{R}{R_{\max}}. \tag{2}$$

3.1.6 Power consumption metric and associated cost. We assume that the power consumption derived from the computation of a job is naturally proportional to the dedicated computation resources. Specifically, job j consumes an amount of power in node n equal to $\alpha c_j^{(t)} + \gamma$ if it is served at time t, where α and γ are constant factors that translate computation capabilities to power consumption.

Furthermore, we assume that serving a job incurs an extra power cost due to the need of reconfiguration, allocation, and initialization of the resources that handle the said job. This cost appears when a job is accepted but also when a job is migrated, since, from the perspective of the node that receives the job, a migrated job is equivalent to *accept* such job in terms of resource reconfiguration. Hence, the power consumed at node n and time t to serve job j is

$$p_{jn}^{(t)} \triangleq (\alpha c_j^{(t)} + \gamma) x_{jn}^{(t)} + \beta y_{jn}^{(t)} \tag{3}$$

where $y_{jn}^{(t)} \in \{0,1\}$ is 1 only if job j arrives to node n in the current time slot, i.e., it is given by

$$y_{jn}^{(t)} = \left(x_{jn}^{(t)} - x_{jn}^{(t-1)}\right) x_{jn}^{(t)}. \tag{4}$$

The cost of migration accounts for the resources' instantiation and management, and β is a constant factor translating such instantiation procedure into power consumption.

Since we are only interested in the consumption of *non-renewable* (-source) energy, we define the non-renewable energy consumption at node n as $p_n^{(t)}$, which is given by

$$p_n^{(t)} \triangleq \max\left(\sum_{j=1}^{J} p_{jn}^{(t)} - g_n^{(t)}, 0 \right) \tag{5}$$

where $g_n^{(t)}$ is the green energy available at node n at time t. Thus, the total non-renewable energy consumption is $P \triangleq \sum_{t=1}^{T} \sum_{n=1}^{N} p_n^{(t)}$, and the associated monetary cost is δP.

3.1.7 Migration of jobs. We consider that jobs can be migrated from one edge server to another. Specifically, we consider that, once a job (i.e., a MAR session) is allocated to one server, such server computes and sends the video frames back to the user, e.g., either 30 or 60 frames per second (fps) at least for the whole duration of one time slot (in the order of seconds, which fits the standard time scale for network function reconfiguration [11]). At the beginning of the next time slot, based on the current network state, the VNF in charge of allocating the jobs may decide to migrate them. Since a job consists of computing video frames, there is no need of heavy data transmission between the servers: it suffices with providing the user metadata. While the new server is instantiating the processes to handle the job, the initial server continues to serve the user. Once the second server is ready, the migration is effectively applied, which provides a seamless experience for the user.

3.2 Optimization problems

We consider two different utilities: the pure economic profit and a proportional fairness-inspired evaluation compromising between profit and consumption of non-renewable energy. We remark that our objective is not finding the optimal allocation for a specific realization of the problem, but finding the *allocation policy* that allows the operator to maximize its long-term utility. This is important because the operator is not aware of the future jobs arrivals nor the future energy availability, and because of that it has to follow a policy that is based on the expected utility from current decisions. Next, we present the two corresponding optimization problems.

3.2.1 Profit maximization. The first problem aims at maximizing the operator's profit. For the sake of readability, we introduce the notations $[X] = \{1, \dots, X\}$, for any positive integer X, and $\mathcal{X} \triangleq \{x_{jn}^{(t)}\}_{j\in[J],n\in[N],t\in[T]}$. We aim at finding the optimal job allocation (and migration), i.e.,

$$\max_{\mathcal{X}} \; E_{\Lambda,\Phi,\Xi}[R - \delta P] \tag{P1}$$

$$s.t. \quad x_{jn}^{(t)} \in \{0,1\} \qquad \forall\, n, j, t \in [N],\ [J],\ [T] \tag{6}$$

$$\sum\nolimits_{n=1}^{N} x_{jn}^{(t)} = a_j \qquad \forall\, j \in [J], t \in [t_j^* : t_j^* + \ell_j] \tag{7}$$

$$\sum\nolimits_{j=1}^{J} c_j^{(t)} x_{jn}^{(t)} \le \tilde{C}_n \qquad \forall\, n, t \in [N],\ [T] \tag{8}$$

where (7) states that, if a job is accepted, it can only be allocated to one node at each time slot, and (8) is the node computation constraint, which ensures that the sum of processing resources allocated at a node n is at most equal to its processing capacity.

We also define the profit margin $\bar{B}$ as the ratio between the profit $R - \delta P$ and the total *potential* revenue $R_{\max}$, such that $\bar{B} \triangleq \frac{R-\delta P}{R_{\max}}$.

3.2.2 Joint optimization of revenue and carbon footprint. For the second optimization problem, the objective follows a proportional fairness structure that allows us to compromise between revenue maximization and minimization non-renewable energy consumption, where the latter is, in our case, equivalent to carbon footprint.

To be able to jointly optimize such disparate metrics as power consumption and revenue, we define a normalized version of the two metrics, such that both are enclosed in the range between 0 and 1. First, for revenue, we consider its normalized expression defined in (2), given by $\bar{R} \triangleq \frac{R}{R_{\max}}$, $\bar{R} \in [0,1]$. For the power metric, we define the *normalized power saving*, which takes the form:

$$\bar{P}_{\rho_p} \triangleq 1 - \rho_p \frac{P}{P_{\max}}, \tag{9}$$

where $P_{\max}$ is defined as the maximum possible non-renewable power consumption, i.e., as $P_{\max} \triangleq \sum_{t=1}^{T}\sum_{n=1}^{N}(P_n^{(\max)} - g_n^{(t)})$, and where $\rho_p \in [0,1)$ is a weight to prevent degenerate cases (since $\bar{P}_{\rho_p} \ge 1 - \rho_b > 0$) and to balance the importance of the power in the objective function. $\bar{P}_{\rho_p}$ can be seen as the percentage of non-renewable energy that we can *save* with respect to the worst case scenario. Note that $\bar{P}_{\rho_p} \in (0,1]$ is maximized when the non-renewable power consumption P is minimized.

Furthermore, we introduce a weight $\rho_r \in [0,1)$ for the revenue term, which aims at tuning the contribution of the revenue to the objective function, such that the final revenue metric is

$$\bar{R}_{\rho_r} = \rho_r \bar{R} + (1 - \rho_r), \tag{10}$$

which is a linear mapping from $[0,1]$ onto $(1-\rho_r, 1]$. The closer the value of ρ_r to 1, the bigger the range of the metric is and thus the more importance it has for the operator.

The two coefficients ρ_r, ρ_p allow us to masquerade or emphasize the contribution of each of the metrics and to avoid that they take value 0, which would cause instability problems due to the logarithmic shape of the function defined next. Their values will depend on the relative importance that each of the two metrics has for the operator. We make use of the proportional-fair rule to jointly optimize both metrics because it ensures that the best solution is such that the sum of relative improvements for each individual term achieved by any other solution is below zero [13], which leads to maximizing $\log(\bar{P}_{\rho_p} \cdot \bar{R}_{\rho_r})$. Hence, our optimization problem is

$$\max_{\mathcal{X}} \; E_{\Lambda,\Phi,\Xi}[\log(\bar{P}_{\rho_p} \cdot \bar{R}_{\rho_r})] \tag{P2}$$

$$s.t. \;\; (6),\ (7),\ (8) \tag{11}$$

3.2.3 Complexity Analysis. We analyze the complexity of both (P1) and (P2), proving that they are NP-hard. For that, we prove that the well-known 0-1 Knapsack Problem (KP) can be reduced to our problem, i.e., that every instance of the KP can be transformed into an instance of our problem. We first recall the definition of the KP.

Definition 3.1 (0-1 Knapsack Problem[18]). Consider a knapsack with capacity $\tilde{C}_n$ and J jobs, where job j has profit p_j and weight w_j. Let $x_j \in \{0,1\}$ denote the variable representing whether job j is introduced in the knapsack ($x_j = 1$). Then, the KP is defined as

$$\underset{\mathcal{X}}{\text{maximize}} \; \sum\nolimits_{j=1}^{J} p_j x_j \tag{12}$$

$$s.t. \quad x_j \in \{0,1\} \qquad \forall j \in [J] \tag{13}$$

$$\sum\nolimits_{j=1}^{J} w_j x_j \le \tilde{C}_n \tag{14}$$

Theorem 3.2. *The problems* (P1) *and* (P2) *are NP-hard.*

Proof. Since (P1) and (P2) only differ in the objective function, we can simultaneously prove both. We start by considering a specific case of our problem, which takes the following assumptions:

(A1) We consider a single time instant ($T = 1$).
(A2) We consider a single node ($N = 1$).

Table 1: Notation

Notation	Description
a_j	$a_j = 1 \Leftrightarrow$ job j is accepted, $a_j = 0$ otherwise
$c_j^{(t)}$	Required processing for job j at time t
$\tilde{C}_n$	CPU capabilities at node n
J	Number of jobs
N	Number of nodes
$p_n^{(t)}$	Non-renewable used power at node n, time t
$g_n^{(t)}$	Available green power at node n at time t
P	Total non-renewable power consumption
$P_n^{(\max)}$	Total available power at node n
$\bar{P}_{\rho_p}$	Normalized non-renewable power savings
R	Total revenue
$R_{\max}$	Maximum revenue (if all jobs are accepted)
$\bar{R}$	Normalized revenue $\bar{R} \triangleq \frac{R}{R_{\max}}$
$\bar{R}_{\rho_r}$	Normalized revenue metric
$t_j^{\star}$	Arrival time slot of job j
$y_{jn}^{(t)}$	Indicates if job j arrived to node n at time t
Λ	Distribution of job's arrival time
Φ	Distribution of job's size
Ξ	Distribution of green power availability
α, γ, β	Constant power–computation factors
$x_{jn}^{(t)}$	Placement variable of j at node n at time t

(A3) We consider that $c_j \leq \tilde{C}_1$ for all j.
(A4) We consider that $g_n^{(t)} \geq P_n^{(\max)}$, for any t.
(A5) Λ, Φ, Ξ are deterministic and known constants.
(A6) All jobs last a single time slot ($\ell_j = 1$).

We can remove the expectation over Λ, Φ, Ξ in the objective functions because from (A5) we know the number and size of all jobs. From (A1)-(A2), we can omit the sub-index n and the super-index (t).

Furthermore, (A4) implies that $p_n^{(t)} = 0$, and thus $P = 0$ and $\bar{P}_{\rho_p} = 1$. Hence, the objective function of (P1) becomes $\max_{\mathcal{X}} R$ and that of (P2) becomes $\max_{\mathcal{X}} \log(\bar{R}_{\rho_r})$. Due to the monotonicity of the log function, the values that maximize $\log(\bar{R}_{\rho_r})$ are the same ones that maximize $\bar{R}$; since we are interested in the argument that maximizes the function rather than the maximum value itself, we can consider $\max_{\mathcal{X}} \bar{R}$ as our objective function for (P2) in this specific case. Since $R_{\max}$ does not depend on the decision variables, we can substitute the objective function in (P2) by $\max_{\mathcal{X}} R$, which matches that of (P1). Hence, in this particular setting given by (A1)–(A6), both (P1) and (P2) are equivalent. Since (1) and the assumption $N = T = 1$ imply that $R = \sum_{j=1}^{J} \eta_j x_j$, our problem is equivalent to

$$\max_{\mathcal{X}} \sum_{j=1}^{J} \eta_j x_j \tag{15}$$

$$s.t. \quad x_j \in \{0, 1\} \qquad \forall j \in [J] \tag{16}$$

$$\sum_{j=1}^{J} c_j x_j \leq \tilde{C}_n \tag{17}$$

By assigning $p_j \leftarrow \eta_j$, $w_j \leftarrow c_j$ in (12), the KP can be reduced to this specific case of our problem. Hence, we can argue that (P1) and (P2) are as complex as KP, which is NP-hard. Since this reduction can be built in polynomial time, both problems are NP-hard. □

4 ALGORITHMS

The previous optimization problem is the formal definition of the objective of the operator. In real network deployments, the operator cannot know how many jobs are going to arrive in the incoming time slots. Because of that, it has to rely on probabilistic policies, which determine the best decision to take at the current moment on the basis of the expected behavior of the system. Furthermore, even if the operator had access to future samples, the NP-hardness of the optimization problem would discourage any attempt to directly solve it, as the complexity and required time for iteratively solving such problem would not be acceptable in a real-time system.

Thus, we need to derive practical algorithms to provide a solution for the problem above. As previously explained in Section 2, approaches based on RL are usually considered for these decision-making problems, where we cannot obtain an optimal solution and the objective depends on the previous and future decisions. For the application of RL, the problem is typically modeled as a Markov Decision Process (MDP) and, consequently, we reformulate the problem as an MDP.

4.1 Reformulation of the Problem as a Markov Decision Process

The scenario presented in Section 3 can be stated as a MDP through the 4-tuple (S, A, P_a, R_a) governing any MDP: The *state space* (S), *action space* (A), the probability distribution of the next state given the current state and action a ($P_a(s'|s)$), and the immediate reward from arriving to state s' from state s due to action a ($R_a(s, s')$).

4.1.1 Agent. The agent corresponds to the edge orchestrator controlling the N edge nodes. At every time slot, it must take J_t decisions, where J_t is the number of jobs that are present at the beginning of time slot t, including the jobs already being served and the new requests arrived since the last time slot.

4.1.2 State space. The state space is comprised of all the information obtained by the agent from the environment that influences the action of the agent. In a given time slot, the state S_t (also called observation) indicates the current value of each one of the variables of interest. Our state space is composed of three different parts: the load of each of the edge nodes, the green energy availability at each edge node, and, finally, an indicator stating whether the next job to be managed is a new arrival request or is already being served by one of the edge nodes. Thus, it contains $2N + 1$ dimensions.

To facilitate the learning convergence, and because RL performs better when dealing with discrete variables, we consider a quantized status of both load and green energy availability. Next, we describe the possible state values and how we discretize the variables.

For the green energy availability, we consider a three-step quantization: state 0 means that there is enough renewable energy to fit more jobs, state 1 that in the current state all energy consumed is renewable but there is not enough to serve a new job, and state 2 that the node is already taking energy from non-renewable sources.

Algorithm 1 GreenRL: Admission control and resource (re)allocation at each time slot

Input: state S_t, set of jobs in the system ($J^{(m)}$), set of new arrived jobs ($J^{(+)}$), green energy $g_n^{(t)}$,
Output: Allocation decisions for time slot t
for $j \in J^{(m)}$ **do** { Migration decisions}
  Agent selects action $a = \pi(S_t)$.
  Check constraints violation ((8) or job's interruption (7)).
  If positive, agent receives a penalty, episode is stopped.
  Evaluate reward and update next state.
end for
for $j \in J_{(+)}$ **do** { Acceptance decisions}
  Agent selects action $a = \pi(S_t)$.
  Check constraint violation (8).
  If positive, agent receives a penalty, episode is stopped.
  Evaluate reward and update next state.
end for

Regarding the nodes' load, we quantize the amount of processing cycles required by the node to compute the allocated jobs in a non-linear way. Specifically, we consider that the quantization step follows a logarithmic progression, such that the steps become smaller as the node is more loaded. This follows from the intuitive idea that the exact load is not so important when the node is handling low computation load, but it becomes more important when it is approaching maximum capacity and thus consuming more energy. The quantization is done to enforce that the last step represents the case where the node cannot accept more jobs and the previous step indicates that there is space for at least one job.

Finally, the last state dimension is a discrete variable that can take $N + 1$ values, from 0 to N, where 0 represents that the job is a new arrived job (and thus it can be rejected or allocated), whereas a value V from 1 to N indicates that the job is already being served and it is currently placed at node V (such that it can be migrated to other node but cannot be interrupted).

4.1.3 Action space. The action space is the description of the agent's decision. In our scenario, the agent decides whether to accept, reject, or migrate each job, which translates in our model to an action space composed of a single discrete variable that can take the values $\{0, 1, \ldots, N\}$. A value $a \in \{1, 2, \ldots, N\}$ indicates that the job is allocated on node a for the next time slot. This value can represent either an allocation of a new job or a migration to a different node in the case of already served jobs if $a \neq S_t(2N + 1)$. Finally, $a = 0$ represents that the job is rejected; consequently, $a = 0$ is only allowed for new jobs as active sessions must not be interrupted.

4.2 Deep Reinforcement Learning-Based Solution: GreenRL

We present next the proposed DRL-based solution, denoted as GreenRL, and whose high-level description is presented in Algorithm 1.

As described above, at the beginning of each time slot t the operator makes a decision for each job present in the network. The operator first handles the jobs that are currently being served in the system. Those jobs must be served until they finish, but they can be migrated from one node to another, which would incur a migration cost as described in Section 3. The agent evaluates job-by-job the possibility to migrate, and once they have been managed, it starts deciding whether the new arrived jobs are accepted or not, and where to allocate them. If accepted, the job provides revenue to the operator (unless it is later interrupted).

We note that it is not trivial to correctly define an adequate reward function, inasmuch as the state depends on decisions taken several time slots in advance due to the shorter time scale of the resource re-allocation time slot (few seconds) with respect to the duration of the jobs (many minutes). Because of that, we consider that the reward is a normalized sliding-window version of the objective functions defined in Section 3, as is detailed in the following.

- *Profit*: To compute the reward, we first calculate the profit obtained in the last T_r time slots, computed as the difference between the revenue provided by the jobs accepted in those T_r time slots and the energy cost generated by *all* the jobs in the system during such interval. Then, we normalize this profit by the total revenue of all the jobs that arrived during the T_r time slots, i.e., the reward corresponds to the value of $\bar{B}$ for the last T_r time slots and lies in the range $[0, 1]$.
- *Fairness*: Similarly, the reward depends on the revenue and cost during the last T_r time slots. Since reward normalization is known to help to achieve better performance for DRL algorithms, instead of computing $\log(\bar{P}_{\rho_p} \cdot \bar{R}_{\rho_r})$ as indicated in (P2), with lies in the range $[\log((1-\rho_r)(1-\rho_p)), 0]$, the reward is given by $\log(\bar{P}_{\rho_p} \cdot \bar{R}_{\rho_r} + 1)/\log(2)$, such that it only takes values in the $[0, 1]$ interval.

Training is split into episodes, each one including up to a maximum number of time slots, and up until the agent takes a decision that violates any of the physical constraints (i.e., it interrupts a job that has been accepted or it allocates to a node more computing resources than its maximum capacity). When an episode is terminated due to a constraint violation, the last reward is set to -1 to prevent the agent to repeat the mistake. Once trained, the agent is modeled through a probabilistic policy $\pi(S_t)$.

In our work, we leverage an algorithm called Asynchronous Advantage Actor Critic (A2C) [20] and, in particular, we use the implementation provided by Stable Baselines3 library[3]. A2C is based on Actor-Critic policy gradient methods, and its main idea is the use of an asynchronous updating scheme that operates on fixed-length segments of experience, executing asynchronously multiple agents in parallel. We refer to the original paper for further details [20].

GreenRL may take actions that lead to QoE disruptions: It could (i) reject jobs when there was green energy available for them, (ii) allocate jobs to a node that is already full, (iii) interrupt an ongoing job. The training process must learn to avoid all these cases.

4.3 Heuristic Algorithm: GreenH

We also propose a heuristic algorithm to compare it with the performance of the DRL-based approach. The goal is to understand the benefits that DRL can bring over designed algorithms that do not suffer from the low performance that the system can face during (possibly long) training periods. This heuristic algorithm, to which

[3] https://stable-baselines3.readthedocs.io/en/master/modules/a2c.html

we refer as GreenH, also handles in-system job migrations and is aware of the green energy distribution across nodes.

The algorithm acts similarly for both in-system jobs and new jobs: It computes the unused green energy at each node, i.e., the available green energy at the node minus the current node consumption due to the jobs processing. Then, it allocates the job to the node with more unused green energy. When all nodes are consuming polluting energy, the decision is random. A job can be forcibly rejected or interrupted if the selected node runs out of computing resources.

4.4 Baselines

We also evaluate the performance of two simple baselines. These algorithms do not implement migration of in-system jobs across nodes, and they do not take into account the distribution of green energy, i.e., they focus on maximizing only the operator's revenue. Consequently, the two algorithms accept all the incoming jobs, and if the node has not enough computing power, the job is forcibly interrupted with the subsequent loss of user QoE. These two baselines algorithms are defined as follows.

- Random: This algorithm selects randomly the node to which each job is sent. The decision is drawn from a uniform discrete distribution of range $\{1, 2, \ldots, N\}$.
- Emptier sends the job to the node that has the lowest load among the N nodes. If there are several nodes with the lowest load, the choice is uniformly random among such nodes.

5 NUMERICAL EVALUATION

We evaluate numerically the four previously described algorithms on a set of network scenarios and varying parameters. Besides these four solutions, we also provide the optimal solution obtained by solving the optimization problem ((P1) or (P2)). We remark that this last result, to which we refer as Solver, is an *ideal* solution that is not feasible since, in order to solve the optimization, we must consider that we know in advance the state of the system in the future time slots (number of arrivals, energy availability, etc.). Solver is also impractical due to the complexity of the problem, since its computation takes a time that is several orders of magnitude bigger than the actual operation time. We built a Python simulator, where the DRL framework is built on Stable-Baselines library and the optimal solution for Solver is built using Python's SciPy library. We performed our experiments in a Dell T640 server with 128 GB of RAM and 40 logical cores.

5.1 Simulation scenario

We evaluate a MEC system as the one presented in Fig. 1, where all edge nodes are interconnected in a full-mesh topology and have the same computing capacity and maximum power consumption. We consider that each edge node offers a computing capacity of $\tilde{C}_n = 2$ TeraFLOPS, which is in line with first MEC deployments in metropolitan areas [25]. Time slots last 5 seconds, which is thus the longest a user would wait to start a session. The value of the factors to transform computation to energy consumption are $\alpha = 0.9$, $\beta = 0.1$ and $\gamma = 0.1$, such that the additional cost of migrating a job is approximately a 10% of the cost of computation per time slot.

We consider MAR jobs as a sequence of video frames. The edge nodes process video frames with resolution 800×800, which is the same order of magnitude usually considered in the literature [16, 21, 28]. According to [16], this video frame size requires 20% of the total computing resources of a 2-TeraFLOPS edge server.

Table 2: Default simulation parameters

Job length (time slots)	7
Arrival Rate (Per time slot)	3
Job computation resources	20% of server capacity $\tilde{C}_n$
$\bar{\eta}$ **(Revenue/time slot/flop)**	10
δ **(Cost/time slot/flop)**	15
Renewable Energy	Random Uniform $\mu = 0.5P_n^{(\max)}$
Fairness weights	$\rho_r = 0.4$, $\rho_p = 0.95$

We assume that a job requires a constant computation per frame. Considering dynamic video frame sizes, which would vary the computation requirements, could be investigated in future work. We assume a static session length of several minutes,[4] much longer than the slot length, and that jobs' arrivals follow a Poisson process.

We assume a set of default values for all the parameters that are valid for all the experiments unless stated otherwise, and which are provided in Table 2. In some cases, we consider that the revenue unit $\bar{\eta}$ is smaller than the cost unit δ (of energy coming from the power grid) due to several reasons: (i) operators' profit margin per unit of service is known to be very small, (ii) the final cost of the service is smaller due to the (relatively) free use of local green energy, and (iii) naturally, if the revenue of a job is always bigger than its cost, the decision will be simpler because all jobs will be accepted, and the only aspect that will matter is *where* to allocate them. The experiments are evaluated by averaging at least 20 different realizations with different renewable energy realizations. Each energy realization is independent of each other to obtain a comprehensive analysis covering all the possible energy distributions.

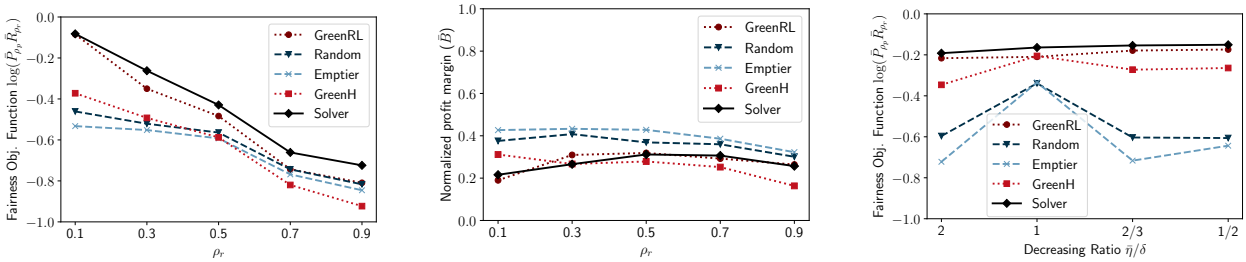
Training of DRL solution GreenRL. The DRL solution is built upon the well-known A2C algorithm. We evaluated also other state-of-the-art DRL approaches, such as Proximal Policy Optimization (PPO) or Deep Q-Learning (DQN), but they were underperforming for all the experiments, and hence we do not include them in the results. As training parameters, we consider a total training duration of 10 million decisions (although it is enough to train for 1 million steps for simple cases, e.g., when $N = 3$), a maximum length per episode of 1024, a learning rate of 0.0007, and a batch size of 8. Learning rate and batch size are selected after a careful evaluation, and they lie within their typical range. Both policy and value neural networks have the same architecture: each one is defined as a MLP network composed of two hidden layers of 64 neurons each.

5.2 Results

We provide the results of the described experiments. In the figures, all the vertical bars represent the 95% confidence interval. We also omit the transient phase from all the experiments.

5.2.1 Trade-off of fairness function. As previously mentioned, the operators may be interested in optimizing different Key Performance Indicators (KPIs) besides the pure economic benefit, KPIs

[4]For longer sessions, we can assume that, once the session has reached a certain duration, a new offloading request is made to renew the service.



(a) (b) (c)

Figure 2: Evaluation of the performance of the algorithms as function of several system parameters for problem (P2)**. We present in (a) the value of the objective function of** (P2) **for different values of the weight of the revenue metric ρ_r, and in (b) the corresponding value of normalized profit margin obtained in this case (when we do not directly optimize the profit). In (c), we show the impact of varying the ratio between revenue ($\bar{\eta}$) and cost (δ) again when solving** (P2)**.**

that are more aligned with high-level goals of the company such as satisfying certain environmental objectives, e.g., the flexible utility introduced in (P2). Yet, the optimal decisions for (P2) will strongly differ depending on the weights ρ_r, ρ_p that are best suited for the operator's objective. To understand the impact of these parameters, we evaluate the performance of the algorithms for different values of $\rho_r \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ for a fixed value of $\rho_p = 0.95$. We evaluate a scenario with 3 edge nodes, serving a set of users whose requests amount to an average load of 50% of the total capacity of the nodes. In this experiment, we consider that the revenue per job is 20% *higher* than the cost of computing such job without local green energy, i.e., $\bar{\eta} = 1.2\delta$. Yet, due to the varying weight of the revenue term in (P2), it is not always better to accept all the jobs.

The results are shown in Figure 2. Figure 2a represents the main objective function (that of (P2)) for which both `Solver` and `GreenRL` are optimized. We observe how `GreenRL` performs close to the (ideal) `Solver`, outperforming the other algorithms by more than 20% except for $\rho_r \geq 0.7$. For $\rho_r \geq 0.7$, `GreenRL` performs as well as the best of the other algorithms because, with the considered level of green energy, accepting all the jobs is almost optimal. Figure 2b represents the resulting (normalized) profit margin. We recall that the algorithms are optimized to maximize the fairness-like expression (P2), and not the profit (P1). `GreenRL` and `Solver` perform worse than the baselines for this metric, but it is expected since they aim to optimize the other metric. Indeed, the relative result w.r.t. the baselines worsens as the weight of the profit (ρ_r) decreases.

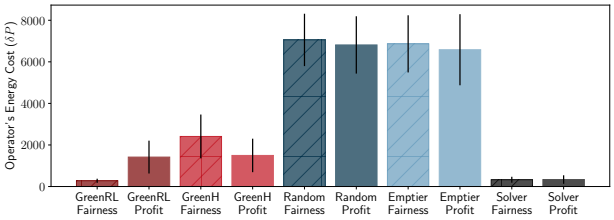
5.2.2 Impact of revenue/cost ratio. The cost of non-renewable energy, which may vary greatly, also impacts the performance of the edge network. Figure 2c shows how changing the relation between revenue and cost could affect the performance of all algorithms, for the case where `Solver` and `GreenRL` are optimized for (P2) ($\log(\bar{P}_{\rho_p}\bar{R}_{\rho_r})$). `GreenRL` is able to perform really close to the solver's performance for any cost, and with an important gap w.r.t. to the baselines. The good result of all the algorithms when $\frac{\bar{\eta}}{\delta} = 1$ is due to the fact that, with that value, accepting a job that consumes only non-renewable energy has the same profit as rejecting the job, and thus different decisions can lead to similar performances.

5.2.3 Baseline network topology with 7 nodes. Figure 3 shows the results for a network topology of 7 edge nodes that abides by the parameters of Table 2, except for the jobs arrival rate, which is set to $\lambda = 4$, such that the average system load is 65%. We provide a detailed analysis, whose results are summarized in Fig. 3, by presenting four metrics of interest: (i) the cost of energy (Fig. 3a), (ii) the normalized profit margin $\bar{B}$ (Fig. 3b), (ii) the portion of users accepted, rejected and interrupted (Fig. 3c), and (iv) the use (and excess) of *green* and *polluting* energy (Fig. 3d), and for the two considered problems: The bars labeled as "Fairness" correspond to the case where the algorithms are designed to optimize (P2), whereas the cases labeled as "Profit" when we optimize for (P1).

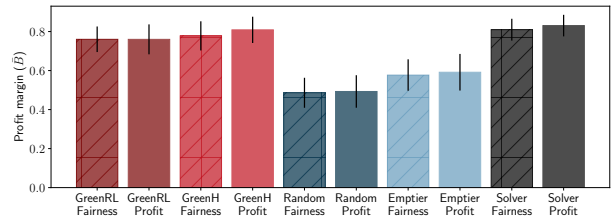
Fig. 3a-3b show how, in the "Fairness" case, `GreenRL` attains the same profit as when it is optimized to maximize the profit, while also greatly reducing the power cost, which endorses the consideration of (P2). In fact, `GreenRL` for (P2) reduces the energy cost by more than 80% w.r.t. `GreenH` and the same `GreenRL` optimized for profit, 95% w.r.t. the baselines, and matches that of `Solver`.

Fig. 3c indicates that `GreenRL` is the most conservative algorithm, since it accepts the smallest number of jobs, but it does so to ensure that no job is interrupted. We remark that, in our model, not accepting a job is not critical, as it is then computed at the user device (with the only drawback of draining its battery), whereas interrupting a job that was offloaded has a huge impact in the end-user QoE, since due to the nature of the MAR sessions it is highly probable that the user is unable to continue the session. The explanation of this conservative behavior is also complemented with Fig. 3d: `Random` is shown in Fig. 3c to be the most aggressive, and Fig. 3d shows that such approach is detrimental because it incurs high consumption of non-renewable energy. Similar rationale can be applied, to a lesser extent, to `Emptier`, and while both `Random` and `Emptier` accept more jobs, they also incur more jobs interruptions and power consumption. Moreover, `GreenRL` performs quite close to `Solver`, which is also quite conservative, although it is able to accept more jobs. In terms of underuse of green energy, the algorithms perform similarly, except for `Solver`; yet, `GreenRL` is the only one that does not use non-renewable energy in place of green energy. The patterns in energy sources usage are also maintained for the case with $N = 10$ edge nodes, represented in Fig. 4.

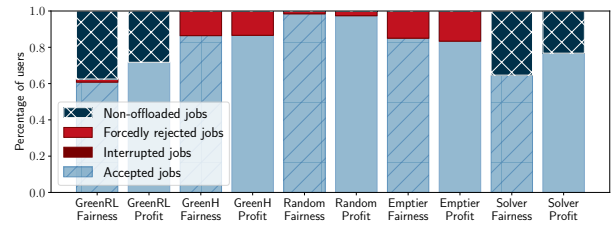
`GreenRL` and `GreenH` enjoy a great performance because they allow migration to nodes with more available green energy. Specifically, throughout all the experiments here presented, `GreenRL`



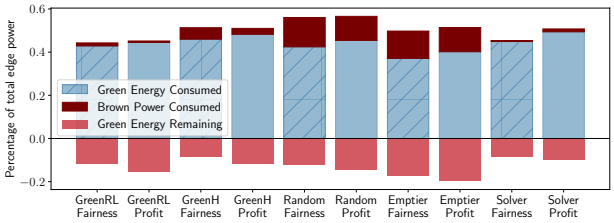
(a) Mean energy cost and standard deviation for the different algorithms



(b) Mean profit margin $\bar{B}$ and standard deviation for the different algorithms



(c) Type of service provided to the users



(d) Energy use by type of energy source and wasted (unused) green energy

Figure 3: Performance for the scenario with $N = 7$ nodes. We represent the results obtained when solving both the problem (P1) (labeled "Profit") and (P2) (labeled "Fairness").

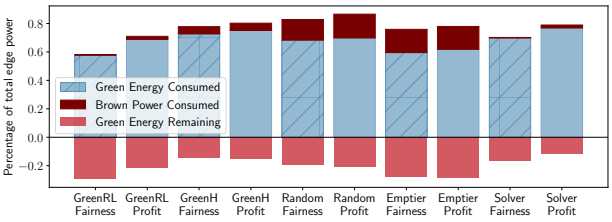


Figure 4: Energy use by type of energy source and wasted (unused) green energy for the scenario with $N = 10$ nodes.

migrates an average of 10% of the jobs in the system, while GreenH migrates 25% of the jobs. We omit a more detailed discussion on migration due to space constraints.

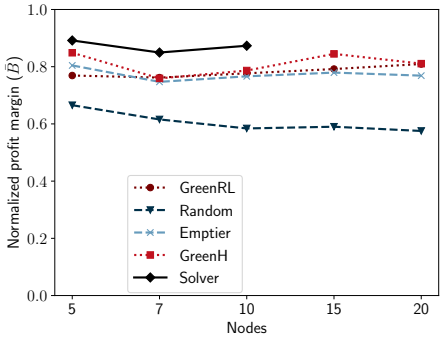


Figure 5: Performance as function of the number of nodes for (P1). Computing load is 65% of the total capacity.

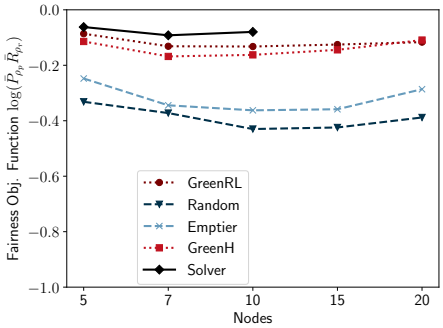


Figure 6: Performance as function of the number of nodes for (P2). Computing load is 65% of the total capacity.

5.2.4 Performance as function of the number of edge nodes. We evaluate the performance obtained by the algorithms when optimizing (P1) (Fig. 5) and (P2) (Fig.6) as function of the number of edge nodes. Due to its intractable complexity, `Solver` is evaluated only up to 10 nodes. For (P1), all algorithms have similar performance except for `Random` since, to maximize profit under the considered parameters, especially the low-to-medium load, the optimal choice is almost always accepting the job. Instead, in Fig. 6, differences are more pronounced as `GreenRL` and `GreenH` outplay the baselines, with `GreenRL` being the best algorithm, tied with `GreenH` for $N = 20$. This is another evidence of how having a DRL-approach adapting its decisions based on different factors could lead to more robust and scalable solutions in MEC settings.

5.2.5 Impact of higher loads with low green energy. Finally, we also evaluate the performance of `GreenRL` in the case where the edge network processes a higher load while sustaining low availability of green energy. For all experiments, we consider the 5-edge-node scenario with an average load of 96% of the maximum capacity of the system and a green energy distribution being uniformly random between 10% and 40% of the maximum required energy. For this scenario, we considered two values of δ for a fixed revenue $\bar{\eta} = 10$.

We report the results in Table 3. Again, `GreenRL` greatly outperforms the other algorithms for problem (P2), achieving a performance which is within the confidence interval of the `Solver`'s one. Instead, for (P1) with cost $\delta = \bar{\eta}$, the simplest baseline performs as good as `Solver`. As previously mentioned, this is due to the fact that the operator obtains the same profit by accepting jobs that

Table 3: Result with high load and low green energy level

Objective →	Fairness (P2)	Fairness (P2)	Profit (P1)	Profit (P1)
Cost (δ)→	10 ($\frac{\eta}{\delta}=1$)	15 ($\frac{\eta}{\delta}=\frac{2}{3}$)	10 ($\frac{\eta}{\delta}=1$)	15 ($\frac{\eta}{\delta}=\frac{2}{3}$)
GreenRL	**-0.410 ± 0.128**	**-0.396 ± 0.019**	**0.332 ± 0.032**	**0.180 ± 0.035**
GreenH	-0.803 ± 0.044	-0.689 ± 0.035	0.194 ± 0.041	0.095 ± 0.041
Random	-0.934 ± 0.096	-0.777 ± 0.041	0.313 ± 0.026	0.126 ± 0.036
Emptier	-1.285 ± 0.144	-1.012 ± 0.080	**0.341 ± 0.035**	0.103 ± 0.043
Solver	-0.378 ± 0.102	-0.331 ± 0.018	0.348 ± 0.102	0.280 ± 0.034

only consume non-renewable energy as it does by rejecting them. However, when costs increase, GreenRL stands out again.

6 CONCLUSIONS

We have analyzed the offloading of MAR tasks in an edge scenario where the edge nodes have variable availability of renewable energy sources, and we have proposed a DRL-based algorithm that is able to adapt the decisions to the current energy availability and energy costs, as well as to different business utilities. We have proposed a flexible utility that offers a trade-off between pure net economic profit and the minimization of non-renewable energy consumption (and, consequently, carbon footprint). The proposed approach is able to adapt the admission control, resource allocation and migration depending on the state of the network, and we have proven through simulations that the model achieves performances close to an ideal optimal solution. We have also shown how job migrations between edge nodes can help to sustain the MAR business model at the edge, which motivates further analysis to understand if migrations also benefit when considering, e.g., the latency of the wireless link or a comprehensive energy model that includes the end devices and the energy consumption due to the data transport.

ACKNOWLEDGMENTS

This work is supported by Project AEON-CPS (TSI-063000-2021-38), funded by the Ministry of Economic Affairs and Digital Transformation and the European Union NextGeneration-EU in the framework of the Spanish Recovery, Transformation and Resilience Plan. A. Bazco-Nogueras is supported by the Regional Government of Madrid through the grant 2020-T2/TIC-20710 for Talent Attraction.

REFERENCES

[1] 2020. *3GPP TR 26.928 Extended Reality (XR) in 5G.* Tech. Rep.
[2] Jaewon Ahn, Joohyung Lee, Dusit Niyato, and Hong-Shik Park. 2020. Novel QoS-Guaranteed Orchestration Scheme for Energy-Efficient Mobile Augmented Reality Applications in Multi-Access Edge Computing. *IEEE Trans. on Vehicular Technology* 69, 11 (2020), 13631–13645. https://doi.org/10.1109/TVT.2020.3020982
[3] Tristan Braud, Pengyuan Zhou, Jussi Kangasharju, and Pan Hui. 2020. Multipath Computation Offloading for Mobile Augmented Reality. In *IEEE Int. Conf. Pervasive Computing and Communications (PerCom).* 1–10. https://doi.org/10.1109/PerCom45495.2020.9127360
[4] R. Buyya and S.N. Srirama. 2019. *Fog and Edge Computing: Principles and Paradigms.* Wiley.
[5] Miaojiang Chen, Wei Liu, Tian Wang, Anfeng Liu, and Zhiwen Zeng. 2021. Edge intelligence computing for mobile augmented reality with deep reinforcement learning approach. *Computer Networks* 195 (2021), 108186.
[6] Xing Chen and Guizhong Liu. 2021. Energy-Efficient Task Offloading and Resource Allocation via Deep Reinforcement Learning for Augmented Reality in Mobile Edge Networks. *IEEE Internet of Things Journal* 8, 13 (2021), 10843–10856. https://doi.org/10.1109/JIOT.2021.3050804
[7] Yuan Cheng. 2020. Edge caching and computing in 5G for mobile augmented reality and haptic internet. *Computer Commun.* 158 (2020), 24–31.
[8] Huawei Technologies Co. 2021. *AR Insight and Application Practice White Paper.* https://carrier.huawei.com/~/media/CNBGV2/download/bws2021/ar-insight-and-application-practice-white-paper-en.pdf.
[9] Xiaoheng Deng, Jingjing Zhang, Honggang Zhang, and Ping Jiang. 2023. Deep-Reinforcement-Learning-Based Resource Allocation for Cloud Gaming via Edge Computing. *IEEE Internet of Things Journal* 10, 6 (2023), 5364–5377. https://doi.org/10.1109/JIOT.2022.3222210
[10] ETSI. 2019. *Multi-Access Edge Computing (MEC); Framework and Reference Architecture.* Technical Report. ETSI MEC ISG.
[11] Juliver Gil Herrera and Juan Felipe Botero. 2016. Resource Allocation in NFV: A Comprehensive Survey. *IEEE Transactions on Network and Service Management* 13, 3 (Sep. 2016), 518–532. https://doi.org/10.1109/TNSM.2016.2598420
[12] Hui Huang, Qiang Ye, and Yitong Zhou. 2022. Deadline-Aware Task Offloading With Partially-Observable Deep Reinforcement Learning for Multi-Access Edge Computing. *IEEE Trans. Netw. Science and Eng.* 9, 6 (2022), 3870–3885. https://doi.org/10.1109/TNSE.2021.3115054
[13] Frank P Kelly, Aman K Maulloo, and David Kim Hong Tan. 1998. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society* 49, 3 (1998), 237–252.
[14] Zeqi Lai, Y. Charlie Hu, Yong Cui, Linhui Sun, Ningwei Dai, and HungSheng Lee. 2020. Furion: Engineering High-Quality Immersive Virtual Reality on Today's Mobile Devices. *IEEE Trans. Mobile Comput.* 19, 7 (2020), 1586–1602. https://doi.org/10.1109/TMC.2019.2913364
[15] Matti Latvaho, Kari Leppänen, Federico Clazzer, and Andrea Munari. 2019. *Key drivers and research challenges for 6G ubiquitous wireless intelligence.* University of Oulu, Oulu, Finland.
[16] Qiang Liu, Siqi Huang, Johnson Opadere, and Tao Han. 2018. An Edge Network Orchestrator for Mobile Augmented Reality. In *IEEE Conf. on Computer Communications (INFOCOM).* 756–764. https://doi.org/10.1109/INFOCOM.2018.8486241
[17] Tong Liu, Shenggang Ni, Xiaoqiang Li, Yanmin Zhu, Linghe Kong, and Yuanyuan Yang. 2023. Deep Reinforcement Learning Based Approach for Online Service Placement and Computation Resource Allocation in Edge Computing. *IEEE Transactions on Mobile Computing* 22, 7 (2023), 3870–3881. https://doi.org/10.1109/TMC.2022.3148254
[18] S. Martello. 1990. Knapsack Problems: Algorithms and Computer Implementations. *Wiley-Interscience series in discrete mathematics and optimiza tion* (1990). https://ci.nii.ac.jp/naid/20000416220/en/
[19] Mahshid Mehrabi, Shiwei Shen, Yilun Hai, Vincent Latzko, George P. Koudouridis, Xavier Gelabert, Martin Reisslein, and Frank H. P. Fitzek. 2021. Mobility- and Energy-Aware Cooperative Edge Offloading for Dependent Computation Tasks. *Network* 1, 2 (2021), 191–214.
[20] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. *CoRR* abs/1602.01783 (2016). arXiv:1602.01783 http://arxiv.org/abs/1602.01783
[21] Diego González Morín, Pablo Pérez, and Ana García Armada. 2022. Toward the Distributed Implementation of Immersive Augmented Reality Architectures on 5G Networks. *IEEE Commun. Magazine* 60, 2 (2022), 46–52. https://doi.org/10.1109/MCOM.001.2100225
[22] Jinke Ren, Yinghui He, Guan Huang, Guanding Yu, Yunlong Cai, and Zhaoyang Zhang. 2019. An Edge-Computing Based Architecture for Mobile Augmented Reality. *IEEE Network* 33, 4 (2019), 162–169. https://doi.org/10.1109/MNET.2018.1800132
[23] Sandra Rodriguez. [n. d.]. *Crafting a Market for for independent XR.* https://xnquebec.co/pdf/Etude_Distribution_XR.pdf.
[24] Yushan Siriwardhana, Pawani Porambage, Madhusanka Liyanage, and Mika Ylianttila. 2021. A Survey on Mobile Augmented Reality With 5G Mobile Edge Computing: Architectures, Applications, and Technical Aspects. *IEEE Communications Surveys Tutorials* 23, 2 (2021), 1160–1192. https://doi.org/10.1109/COMST.2021.3061981
[25] Francesco Spinelli, Antonio Bazco-Nogueras, and Vincenzo Mancuso. 2022. Edge Gaming: A Greening Perspective. *Computer Commun.* 192 (2022), 89–105. https://doi.org/10.1016/j.comcom.2022.05.022
[26] Francesco Spinelli and Vincenzo Mancuso. 2021. Toward Enabled Industrial Verticals in 5G: A Survey on MEC-Based Approaches to Provisioning and Flexibility. *IEEE Commun. Surveys Tutorials* 23, 1 (2021), 596–630. https://doi.org/10.1109/COMST.2020.3037674
[27] Ming Tang and Vincent W.S. Wong. 2022. Deep Reinforcement Learning for Task Offloading in Mobile Edge Computing Systems. *IEEE Transactions on Mobile Computing* 21, 6 (2022), 1985–1997. https://doi.org/10.1109/TMC.2020.3036871
[28] Haoxin Wang and Jiang Xie. 2020. User Preference Based Energy-Aware Mobile AR System with Edge Computing. In *IEEE Conf. on Computer Communications (INFOCOM).* 1379–1388. https://doi.org/10.1109/INFOCOM41043.2020.9155517
[29] Yue Wang, Tao Yu, and Kei Sakaguchi. 2021. Context-Based MEC Platform for Augmented-Reality Services in 5G Networks. In *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall).* 1–5. https://doi.org/10.1109/VTC2021-Fall52928.2021.9625304
[30] Han Xiao, Changqiao Xu, Yunxiao Ma, Shujie Yang, Lujie Zhong, and Gabriel-Miro Muntean. 2021. Edge Computing-Assisted Multimedia Service Energy Optimization based on Deep Reinforcement Learning. In *IEEE Global Communications Conf. (GLOBECOM).* https://doi.org/10.1109/GLOBECOM46510.2021.9685687