AI Applications Resource Allocation in Computing Continuum: a Stackelberg Game Approach

Roberto Sala, Hamta Sedghani, Mauro Passacantando, Giacomo Verticale and Danilo Ardagna

Abstract—The growth, development, and commercialization of artificial intelligence-based technologies such as self-driving cars, augmented-reality viewers, chatbots, and virtual assistants are driving the need for increased computing power. Most of these applications rely on Deep Neural Networks (DNNs), which demand substantial computing capacity to meet user demands. However, this capacity cannot be fully provided by users' local devices due to their limited processing power, nor by cloud data centers due to high transmission latency from long distances. Edge cloud computing addresses this issue by processing user requests through 5G, which reduces transmission latency from local devices to computing resources and allows the offloading of some computations to cloud back-ends. This paper introduces a model for a Mobile Edge Cloud system designed for an application based on a DNN. The interaction among multiple mobile users and the edge platform is formulated as a one-leader multi-follower Stackelberg game, resulting in a challenging non-convex mixed integer nonlinear programming (MINLP) problem. To tackle this, we propose a heuristic approach based on Karush-Kuhn-Tucker conditions, which solves the MINLP problem significantly faster than the commercial state-of-the-art solvers (up to 50,000 times). Furthermore, we present an algorithm to estimate optimal platform profit when sensitive user parameters are unknown. Comparing this with the full-knowledge scenario, we observe a profit loss of approximately 1%. Lastly, we analyze the advantages for an edge provider to engage in a Stackelberg game rather than setting a fixed price for its users, showing potential profit increases ranging from 16% to 66%.

Index Terms—Stackelberg game, Mobile Edge Cloud system, DNN partitioning.

✦

1 INTRODUCTION

Artificial Intelligence (AI) and Deep Learning (DL) have experienced significant growth in recent years, largely due to advancements in cloud computing and related technologies. IDC predicts total spending on AI services to reach $52.6 billion by 2025, at a Compound Annual Growth Rate of 21.9% [1]. By combining AI with cloud computing, it is possible to create a vast network capable of handling massive amounts of data, and generating predictions, all while constantly learning and improving performance.

Recently, AI applications have increasingly targeted mobile computing and the Internet of Things (IoT), including virtual reality (VR) [2], augmented reality (AR) [3], autonomous vehicles [4], and natural language processing for real-time translation systems [5]. The significant growth in IoT devices highlights the progress made, with projections suggesting that the number of IoT devices could reach 125 billion by 2030 [6].

The rise of AI-based IoT applications has underscored the challenges of processing data on mobile devices. Cloud computing has emerged as a prevalent solution to support these applications for resource-constrained devices by offloading tasks to the cloud, where specialized AI acceleration hardware can provide powerful task inference capabilities. However, this approach introduces long-distance data transmission, leading to high transmission delays and costs.

Although cloud computing offers large processing capacity via powerful virtual servers, it cannot meet the needs of real-time processing. As the volume of data generated by IoT devices continues to grow rapidly, the need for real-time data processing has become more critical. Most current applications have strict response time requirements, and today's users are sensitive to delay and jitter.

Edge computing represents a new computing paradigm that aims to move AI and machine learning closer to where data generation occurs. According to the Cisco Global Cloud Index, in 2020, 45% of the data produced by things, people, and machines were analyzed, processed, stored, and acted upon by edge computing [7]. In the 5G era, Mobile Edge Computing (MEC) stands out as a compelling strategy that leverages edge cloud technology, with the primary objective of moving computation closer to the end-user to enhance the management and efficiency of geographically dispersed mobile devices and applications [8].

Recent AI applications, especially in the realm of AR, increasingly rely on Deep Neural Networks (DNNs) with multiple layers to process data captured from cameras and to integrate additional inputs from device sensors such as accelerometers, gyroscopes, and GPS [9]. The scarcity of resources in IoT devices can significantly impact the performance of applications with Quality of Service (QoS) response time constraints. One common approach to address this challenge is to partition the DNN and assign the

---

This paper has been published in IEEE Transactions on Cloud Computing*, vol. 13, no. 1, pp. 166–183, Jan.-March 2025, DOI: 10.1109/TCC.2024.3521213, see https://ieeexplore.ieee.org/abstract/document/10811813*

- *R. Sala, H. Sedghani, G.Verticale and D. Ardagna are with Politecnico di Milano. Milan, Italy. Email: {name.lastname}@polimi.it*
- *M. Passacantando is with University of Milano-Bicocca, Milan, Italy. Email: mauro.passacantando@unimib.it*

partitions to the available resources on the edge or cloud. This strategy optimizes resource utilization while meeting QoS constraints [10]. Tasks can be offloaded from IoT devices to the edge and then to the cloud. The two segments of the DNN process the task successively, with different partition points requiring varying amounts of computation and intermediate data transmission between IoT devices and the edge/cloud. Consequently, an efficient resource allocation strategy that maximizes resource utilization in edge-cloud-assisted IoT environments must be closely tied to the DNN partition strategy [11].

The difficulty of allocating resources across the computing continuum arises from the limited capacity and high costs of running applications on mobile devices and in edge/cloud environments. To address this challenge, game theory models are necessary to support the execution of AI applications. These models must consider two main factors: first, the memory and energy available on IoT devices, as well as the user's budget when executing in the field; and second, the costs associated with guaranteeing a specific latency threshold and the competition for resources at the edge provider site (e.g., a set of small data centers accessed through 5G towers) and in the remote cloud.

The focus of this work is a MEC system where a platform, located at the edge, comprises a limited number of edge servers and has access to a cloud-based pool of unlimited virtual machines (VMs) with the goal of serving a large number of mobile users with smart devices. These users want to run a DNN-based AI application that connects to the edge platform over a network with non-negligible latency. Our objective is to optimize the execution cost of the AI application by partitioning the DNN in a way that balances performance and cost. Notice that DNN partitioning favours user-side privacy [12], [13], [14].

Each user has energy and memory limitations to run the AI application, and a response time performance constraint must be met. We develop a model where smart devices require the support of a mobile phone for both running the application and communicating with the edge platform, resulting in three partitions of the DNN application.

To the best of our knowledge, this is the first work aiming at designing a mechanism that encourages users to run an AI application based on their satisfaction level. Users will only be motivated to run the application if the cost is less than their satisfaction value. To achieve this goal, we employ a one-leader-multi-follower Stackelberg game to model the problem.

The goal of the platform provider/application owner, as the leader, is to maximize its profit by incentivizing more users to participate and managing the allocation of resources among users while guaranteeing the application performance constraint. Conversely, the users, as selfish individual followers, aim to minimize the cost of running the AI application by choosing an appropriate deployment based on their budget, energy, and memory constraints.

Furthermore, we examine two distinct scenarios: one where the platform has full knowledge and one where it has partial knowledge about the users' parameters. In the first ideal scenario, the Stackelberg game can be simplified to a mixed-integer nonlinear programming (MINLP) problem because the platform can anticipate users' decisions. However, due to the presence of non-convex terms in the objective function and the large number of constraints, a game-theoretic approach is essential for finding a solution efficiently. Furthermore, the developments in this scenario not only allow us to assess the algorithm effectiveness under partial knowledge but also equip us with the tools to identify solutions within this framework.

In the more realistic scenario of partial knowledge, the platform lacks information about users' sensitive parameters, such as their budgets, energy and data transfer costs, and the energy and memory levels of their devices. This approach enhances user privacy and necessitates a one-leader-multi-follower Stackelberg game structure to solve the associated problem.

In summary, the main contributions of this work are as follows:

1) We formulate the interaction among multiple mobile users and the edge platform as a one-leader (platform) multi-follower (users) Stackelberg game.
2) In the full knowledge scenario, we reformulate the Stackelberg game as a MINLP problem and we develop a heuristic approach to derive a closed form for the optimal number of edge and cloud resources. The resulting algorithm solves the MINLP problem quickly, i.e., in at most 2 seconds in the worst case. Moreover, we evaluate the performance of the proposed approach by comparing it with the BARON state-of-the-art solver [15]. The results show that our approach can find the optimal solution much faster than BARON (up to 50,000x) when considering a small number of users. For a large number of users, the global solver does not achieve convergence within two hours.
3) In the partial knowledge scenario, we develop a heuristic algorithm to estimate the best profit for the edge provider. Experimental results show a profit loss of less than 1% with respect to the full knowledge scenario.

The rest of the paper is structured as follows. Section 2 outlines our approach to modeling the assignment of AI application resources among users' smart devices, edge, and cloud resources. In Section 3, we formulate the problem as a Stackelberg game, and the proposed resolution approach is explained in Section 4. Experimental results are presented in Section 5. Section 6 discusses related works. Finally, Section 7 provides conclusions and possible developments.

2 SYSTEM MODEL

In this section, we model our reference MEC system and formulate the resource allocation problem for an AI application among users' smart devices, edge resources, and the cloud. First, we detail the system model (Section 2.1), followed by the description of the platform problem (Section 2.2) and the user problem (Section 2.3).

2.1 Mobile Edge Cloud model

Our MEC scenario is depicted in Figure 1. We consider mobile users with smart devices wishing to run an AI application. For instance, we consider AR smart glasses and an AI application where each frame is processed with a DNN. These devices only have short-range communications

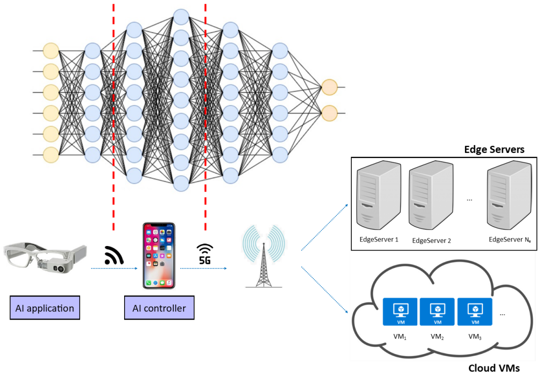


Figure 1: Architecture of the system, showing an example deployment in which the DNN is split into three parts.

capabilities [1] and can either process the DNN locally or they can offload part of the computation to a *far edge* device such as a smartphone. The smartphone, running an AI controller agent, interacts with the edge platform and transmits user-specific information, such as battery energy levels.

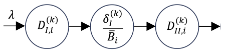
The AI controller can further offload part of the computation to a *near edge* platform or to a cloud platform, both of which can be reached by means of a 5G network. The edge platform comprises N_e identical servers and can be reached with a smaller latency, while the central cloud has an unlimited pool of virtual machines (VMs), but can be reached with a larger latency. We also assume that the wireless link is the bottleneck, while the backhauling network operates at high speed.

We define the concept of *deployment*, which specifies how the DNN operates: either entirely on users' devices or by offloading the final part of the execution to the edge platform. In the latter scenario, the edge platform determines whether to process the final part of the DNN on its servers or to offload and execute it in the cloud. Let $\mathcal{U}$ be the set of users, $\mathcal{D}$ the set of deployments, and $\mathcal{D}' \subset \mathcal{D}$ the subset of deployments where part of the computation is offloaded to the edge platform. From now on, we use the index i to denote users and k to denote deployments. While deployments in $\mathcal{D}'$ involve a tripartition of the DNN, with the third part of the application being offloaded, the local deployments in $\mathcal{D} \setminus \mathcal{D}'$ split the inference of the DNN solely among local devices, differing in how the load is distributed. Figure 1 shows an example of the resource partitioning in the MEC system, for a deployment $k \in \mathcal{D}'$.

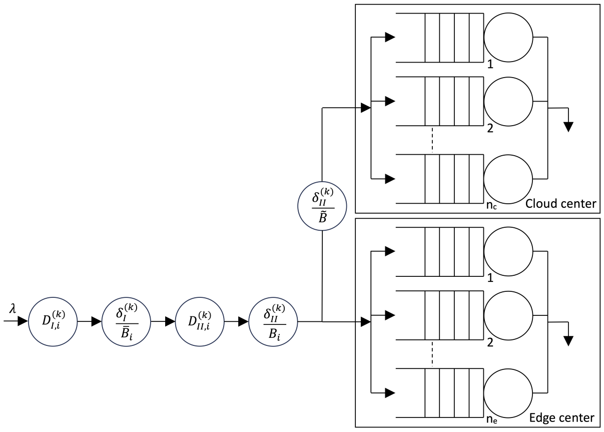
The edge provider aims to maximize its net income, which is the revenue from supporting users' applications minus the costs of running the deployments on edge servers and remote cloud VMs. The platform can serve the deployments by allocating $n_e^{(k)}$ edge servers and $n_c^{(k)}$ cloud VMs for $k \in \mathcal{D}'$. The following constraint must hold:

$$\sum_{k \in \mathcal{D}'} n_e^{(k)} \leq N_e, \tag{1}$$

indicating that the number of edge servers allocated is at most equal to the number of available ones. Both the servers and VMs are modelled as single-server multiple-class queue systems (i.e., individual M/G/1 queues) as in [16], while the local user devices are modelled as delay centers (see Figure 2). Let $D_e^{(k)}$ and $D_c^{(k)}$ denote the *demanding time* (i.e., the time needed to serve a single request without resource contention [17]) for deployments $k \in \mathcal{D}'$ on the edge servers and cloud VMs, respectively. We introduce the following binary variables to define the assignment decisions, specifically to determine which candidate deployment is selected and how the corresponding deployments are assigned to resources:



(a) Deployment that only involves local resources ($k \in \mathcal{D} \setminus \mathcal{D}'$).


λ
$D_{I,i}^{(k)}$
$\frac{\delta_I^{(k)}}{B_i}$
$D_{II,i}^{(k)}$
$\frac{\delta_{II}^{(k)}}{B_i}$
$\frac{\delta_{II}^{(k)}}{\bar{B}}$
1
2
n_c
Cloud center
1
2
n_e
Edge center

(b) Deployment that involves local resources and either edge servers or cloud VMs ($k \in \mathcal{D}'$).

Figure 2: Queuing model of resources.

$$x_i^{(k)} = \begin{cases} 1 & \text{if user } i \text{ selects deployment } k, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

$$y_i^e = \begin{cases} 1 & \text{if user } i \text{ is served by edge servers,} \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

$$y_i^c = \begin{cases} 1 & \text{if user } i \text{ is served by cloud VMs,} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

$x_i^{(k)}$ represent user decisions, while y_i^e and y_i^c are the decision variables of the edge platform. The following constraint ensures that users select at most one deployment:

$$\sum_{k \in \mathcal{D}} x_i^{(k)} \leq 1 \qquad \forall\, i \in \mathcal{U}, \tag{5}$$

It is worth noting that if a user i decides to stop using the application, then all $x_i^{(k)}$ variables are set to zero. Moreover, the following constraint ensures that if user i selects a deployment $k \in \mathcal{D}'$, the platform assigns the computation either to the edge or to the cloud:

$$y_i^e + y_i^c = \sum_{k \in \mathcal{D}'} x_i^{(k)} \qquad \forall\, i \in \mathcal{U}. \tag{6}$$

Let λ be the input load (expressed in requests per second) for each user, such as the frame rate of acquisition for smart glasses. The total load for each deployment k is:

$$\lambda^{(k)} = \sum_{i \in \mathcal{U}} \lambda x_i^{(k)} \qquad \forall\, k \in \mathcal{D}'. \tag{7}$$

Let $L_e^{(k)}$ and $L_c^{(k)}$ be:

$$L_e^{(k)} = \sum_{i \in \mathcal{U}} D_e^{(k)} x_i^{(k)} \lambda y_i^e \qquad \forall\, k \in \mathcal{D}', \tag{8}$$

$$L_c^{(k)} = \sum_{i \in \mathcal{U}} D_c^{(k)} x_i^{(k)} \lambda y_i^c \qquad \forall\, k \in \mathcal{D}'. \tag{9}$$

---

[1] e.g. Wi-Fi 7, which is the state-of-the-art for low-latency mixed-reality applications: https://www.qualcomm.com/products/technology/wi-fi/wi-fi-7.

To avoid resource saturation, the equilibrium conditions for the M/G/1 queue must hold (i.e., utilization less than 1). This is equivalent to prescribing the following constraints:

$$L_e^{(k)} < n_e^{(k)} \qquad \forall\, k \in \mathcal{D}', \tag{10}$$

$$L_c^{(k)} < n_c^{(k)} \qquad \forall\, k \in \mathcal{D}'. \tag{11}$$

From a cost perspective, we assume that the edge platform owns the edge infrastructure. Therefore, it aims to maximize the utilization of edge resources before considering the lease of additional resources on a public cloud. Let c_e and c_c denote the cost per second of edge and cloud VMs, respectively (per-second billing options are recently available in, e.g., AWS [18] and Azure [19]). Let T_i denote the activation time of the application for user i. For instance, T_i could be the user's navigation time, which can be predicted with high accuracy. Additionally, the platform must reconfigure resources for each new user login, each user drop, and periodically at time intervals of T. This computation can run asynchronously in the background based on user logins, ensuring no delays are added to the application [20]. Furthermore, let $r^{(k)}$ represent the cost per second incurred by the users to run the application, which is discussed later. The profit of the edge platform is given by:

$$P_e = \sum_{i\in\mathcal{U}} \sum_{k\in\mathcal{D}} T_i r^{(k)} x_i^{(k)} - T \left[c_e \sum_{k\in\mathcal{D}'} n_e^{(k)} + c_c \sum_{k\in\mathcal{D}'} n_c^{(k)} \right]. \tag{12}$$

The first term represents the revenue from users utilizing the application, while the second term accounts for the costs of edge servers and cloud VMs.

We assume that the cost of running deployments on the edge is always lower than on the cloud, i.e., $c_e D_e^{(k)} < c_c D_c^{(k)}$ holds for any $k \in \mathcal{D}'$. Moreover, we assume that $D_e^{(k)}\lambda < 1$ holds for any $k \in \mathcal{D}'$, indicating that one edge server is sufficiently powerful to handle the load of a single user.

In addition, the users' devices have energy and memory constraints, and there is a latency constraint that must be satisfied for all the users. The objective is to partition the execution of the AI components in such a way that minimizes execution costs while ensuring the latency constraint is met.

From now on, we denote the smart device capturing frames as $j = I$ and the smartphones as $j = II$. Define $p_{j,i}^{(k)}$ as the power consumption per request to run deployment k, and $\bar{E}_{j,i}$ as the battery level of the smart device and phone of user i. The energy consumption constraint is as follows:

$$\lambda T_i^2 \sum_{k\in\mathcal{D}} p_{j,i}^{(k)} x_i^{(k)} \leq \bar{E}_{j,i}, \qquad \forall\, j \in \{I, II\}, \tag{13}$$

In particular, λT_i represents the total number of requests to be satisfied within the time T_i, while $T_i p_{j,i}^{(k)}$ is the energy required per request for user i. Multiplying these two quantities gives the total energy needed by the user to run deployment k on each device ($j = I, II$).

Similarly, let $m_j^{(k)}$ denote the memory requirements of deployment k, and $\bar{M}_{j,i}$ the available memory on the devices. The memory constraint is:

$$\sum_{k\in\mathcal{D}} m_j^{(k)} x_i^{(k)} \leq \bar{M}_{j,i}, \qquad \forall\, j \in \{I, II\} \tag{14}$$

Let $D_{I,i}^{(k)}$ and $D_{II,i}^{(k)}$ be the demanding times required to run deployment $k \in \mathcal{D}$ on the smart device and smartphone of user i, respectively. Let $\delta_I^{(k)}$ and $\delta_{II}^{(k)}$ denote, respectively, the data transfer size from the user's smart device to the smartphone and from the smartphone to the edge servers or to the cloud VMs for deployment k.

The smart device of user i sends the data $\delta_I^{(k)}$ over a fast Wi-Fi connection with a bandwidth $\bar{B}_i$. In case the deployment offloads part of the computation to the edge servers, the smartphone sends the data $\delta_{II}^{(k)}$ over a 5G wireless link with bandwidth B_i. Alternatively, if the deployment offloads part of the computation to the cloud VMs, the smartphone sends the data $\delta_{II}^{(k)}$ over the 5G access network with bandwidth B_i and then to the cloud over a high-speed network with bandwidth $\tilde{B}$. We assume that the bandwidth of the 5G wireless link B_i is smaller than both $\bar{B}_i$ and $\tilde{B}$. It is also worth noting that each user can be served by either the edge or the cloud, but not both.

The latency constraint for the user is related to the response time of the application (see Figure 2 for the full queuing model):

$$\begin{aligned}
&\sum_{k\in\mathcal{D}} D_{I,i}^{(k)} x_i^{(k)} + \sum_{k\in\mathcal{D}} \frac{\delta_I^{(k)} x_i^{(k)}}{\bar{B}_i} + \sum_{k\in\mathcal{D}} D_{II,i}^{(k)} x_i^{(k)} + \\
&\quad \sum_{k\in\mathcal{D}'} \frac{\delta_{II}^{(k)} x_i^{(k)}}{B_i} + \sum_{k\in\mathcal{D}'} \frac{D_e^{(k)} x_i^{(k)} y_i^e}{1 - \frac{L_e^{(k)}}{n_e^{(k)}}} + \sum_{k\in\mathcal{D}'} \frac{\delta_{II}^{(k)} x_i^{(k)} y_i^c}{\tilde{B}} + \\
&\qquad \sum_{k\in\mathcal{D}'} \frac{D_c^{(k)} x_i^{(k)} y_i^c}{1 - \frac{L_c^{(k)}}{n_c^{(k)}}} \leq \bar{R}, \qquad \forall\, i \in \mathcal{U}.
\end{aligned} \tag{15}$$

The first and third terms represent the execution time on the smart device and smartphone, respectively. The second and fourth terms denote the data transmission latencies, while the fifth and seventh terms indicate the execution time on the edge and cloud, respectively. The sixth term represents the data transmission time between the edge and the cloud.

We assume that $D_c^{(k)} + \frac{\delta_{II}^{(k)}}{\tilde{B}} < D_e^{(k)}$ holds for any $k \in \mathcal{D}'$, which implies that cloud VMs are more powerful and faster, considering the transfer delay, compared to edge servers. From a cost perspective, user i is incentivized to join the system if the value U_i (inspired by incentive mechanisms like [21]), representing the satisfaction derived from running the application, exceeds the total cost incurred. This cost includes the expenses related to using the application itself, the energy consumption of the device, and the cost associated with data transmission. Regarding the AI application, users pay a per-second fee $r^{(k)}$ for running it. This fee is constant for deployments that run entirely on local devices, i.e., $r^{(k)} = r_0,\ \forall k \in \mathcal{D} \setminus \mathcal{D}'$. For deployments involving resource offloading, users incur an additional cost. In this case, the fee is $r^{(k)} = r_0 + \gamma^{(k)} r$, where $\gamma^{(k)} \geq 1$ for all $k \in \mathcal{D}'$. The value of r is set within the range $[r_{\min}, r_{\max}]$, while $\gamma^{(k)}$ is proportional to the execution time of the offloaded DNN partition. Let α_i be a parameter to balance the trade-off between cost and energy, β_i a coefficient to convert the energy consumption of the user's device to monetary cost, and ζ_i a coefficient to convert the data transfer size to monetary cost. The cost for user i to run the application is as follows:

$$\begin{aligned}
C_i = T_i \Bigg\{ &\sum_{k\in\mathcal{D}} x_i^{(k)} \left[\alpha_i r^{(k)} + (1-\alpha_i)\beta_i \left(p_{I,i}^{(k)} + p_{II,i}^{(k)} \right) \lambda T_i \right] \\
&+ \sum_{k\in\mathcal{D}'} x_i^{(k)} \zeta_i\, \delta_{II}^{(k)} \lambda \Bigg\}.
\end{aligned} \tag{16}$$

An example of an AI application with four deployments is shown in Figure 3. The first two deployments ($k = 1, 2$, see Figure 3a and Figure 3b) are deployed only on local devices, while the other two ($k = 3, 4$, see Figure 3c and Figure 3d) have three partitions, with the first two running on user devices and the third on edge servers or cloud VMs. Note that this model mitigates data privacy concerns by performing partial local processing of image frames and transmitting final tensors, thereby enhancing security and privacy for end-users with minimal additional overhead [12], [13], [14]. All the parameters and variables adopted in this paper are summarized in Tables 1 and 2.

Table 1: Problem Parameters.

$\mathcal{U}$	Set of users.
$\mathcal{D}$	Set of deployments.
$\mathcal{D}'$	Set of deployments running partially on edge/cloud.
N_e	Max. number of available nodes in the edge.
$D_{I,i}^{(k)}$	Demanding time to run dep. k on user i smart device.
$D_{II,i}^{(k)}$	Demanding time to run dep. k on user i smartphone.
$D_e^{(k)}$	Demanding time to run dep. k on edge server.
$D_c^{(k)}$	Demanding time to run dep. k on cloud VM.
λ	Incoming workload of user.
$\bar{B}_i$	Bandwidth of connection from user i smart device to phone (Wi-Fi 7 network).
B_i	Bandwidth of connection from user i smartphone to the edge platform (5G connection).
$\tilde{B}$	Bandwidth of connection from the edge platform to the cloud (dedicated links).
R_i	User's i response time.
$\bar{R}$	Upper bound threshold for the average response time of the application.
c_e	Cost per second of edge server.
c_c	Cost per second of cloud VMs.
β_i	Coefficient to convert the energy consumption of user i's device to monetary costs.
α_i	Parameter to trade off the cost of the application and energy consumption of users.
T_i	Total application activation time for user i.
T	Time horizon.
$\gamma^{(k)}$	Multiplication coefficient of extra cost for dep. k.
r_0	Constant service fee per second for local deployment.
ζ_i	Coefficient to convert the data size transfer from user's i device to monetary costs.
U_i	Value for user i to run the application.
$p_{I,i}^{(k)}$	Power consumption per request of user i smart device to run dep. k.
$p_{II,i}^{(k)}$	Power consumption per request of user i smartphone to run dep. k.
$m_I^{(k)}$	Memory requirement on smart device to run dep. k.
$m_{II}^{(k)}$	Memory requirement on smartphone to run dep. k.
$\bar{E}_{I,i}$	Energy level of user i smart device.
$\bar{E}_{II,i}$	Energy level of user i's smartphone.
$\bar{M}_{I,i}$	Memory level of user i's smart device.
$\bar{M}_{II,i}$	Memory level of user i's smartphone.
$\delta_I^{(k)}$	Data transfer size of the first partition of dep. k from user's smart device to its smartphone.
$\delta_{II}^{(k)}$	Data transfer size of the second partition of dep. k from user's smartphone to the edge platform.

Table 2: Decision Variables.

$x_i^{(k)}$	1 if user i selected dep. k; 0 otherwise
r	Extra cost per second for running dep. $k \in \mathcal{D}'$
$n_e^{(k)}$	Number of edge servers to serve dep. k
$n_c^{(k)}$	Number of cloud VMs to serve dep. k
y_i^e	1 if user i is served by the edge; 0 otherwise
y_i^c	1 if user i is served by the cloud; 0 otherwise

To summarise, the edge provider has to decide on the application price r and the allocation of computational resources for users opting for resource offloading ($n_e^{(k)}$, $n_c^{(k)}$, y_i^e, y_i^c), based on users' participation, to maximize profit. On the other hand, users, knowing the price set by the edge provider, have to choose their deployment $x_i^{(k)}$, according to the energy and memory levels of their devices, to minimize costs. Since each user's decision follows the provider's decision on the application price, we model the problem as a one-leader-multi-follower Stackelberg game, where the edge platform acts as the leader and the users as followers. Accordingly, we introduce the platform and users' problem in the following subsections (2.2 and 2.3, respectively) and formulate the Stackelberg game in Section 3.

2.2 Platform side problem

The platform aims to maximize its profit by deciding on the extra cost r, the number of edge servers $n_e^{(k)}$, and cloud VMs $n_c^{(k)}$ allocated for each deployment $k \in \mathcal{D}'$ to meet user demand while satisfying all constraints. Additionally, it considers allocation variables y_i^e and y_i^c for users selecting partitioned deployments on edge or cloud. The edge platform solves the following optimization problem:

$$\max_{r, n_e^{(k)}, n_c^{(k)}, y_i^e, y_i^c} \quad P_e \tag{17a}$$

$$\text{s.t. constraints (1), (6), (10), (11), (15),} \tag{17b}$$

$$r^{(k)} = r_0 + \gamma^{(k)} r \quad \forall k \in \mathcal{D}', \tag{17c}$$

$$r \in [r_{min}, r_{max}], \quad y_i^e, y_i^c \in \{0,1\}, \quad n_e^{(k)}, n_c^{(k)} \in \mathbb{Z}_+, \tag{17d}$$

where P_e is computed as in (12). The platform net profit is a piecewise linear function: as the extra price r increases, some users change deployments, and others stop running the application altogether. An example of this scenario with $N = 10$ users is shown in Figure 5. This piecewise linear function is dependent on r, and given the objective function of the platform (12), the slope is equal to $\sum_{i \in \mathcal{U}} \sum_{k \in \mathcal{D}'} T_i \gamma^{(k)} x_i^{(k)}(r)$, which is piecewise constant. The dependence of $x_i^{(k)}$ on r, as highlighted in the previous expression, indicates that for different values of r, the slope of the profit function varies. Moreover, for "high" values of r, the platform profit is constant because users either stop running the application or choose one deployment in $\mathcal{D} \setminus \mathcal{D}'$, whose cost is independent of that variable.

2.3 User side problem

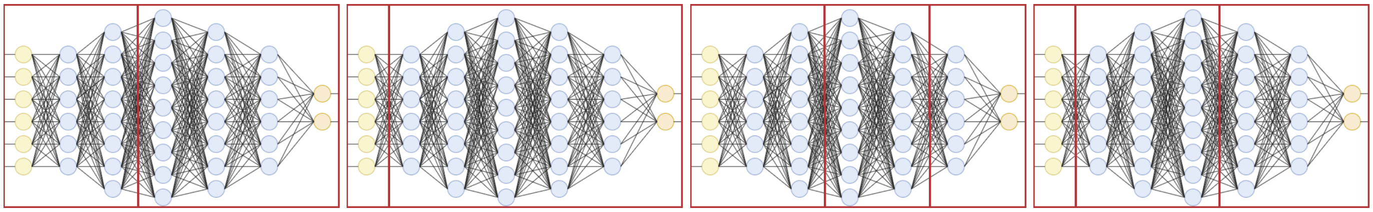
Each user i aims to run the application only if the cost C_i associated with it is lower than its satisfaction value U_i. Given constraints on energy and memory levels of its devices, user i has to select the deployment $x_i^{(k)}$ for running the application. The optimization problem for each user i is formulated as follows:

$$\max_{x_i^{(k)}} \sum_{k \in \mathcal{D}} x_i^{(k)} U_i - C_i \tag{18a}$$

$$\text{s.t. constraints (5), (13), (14),} \tag{18b}$$

$$x_i^{(k)} \in \{0,1\} \qquad \forall\, k \in \mathcal{D}, \tag{18c}$$

where C_i is computed as in (16). In this way, user i is motivated to join the system if the price proposed by the platform is lower than the value U_i associated with the application. Note that with U_i defined, a specific budget constraint for the user is not necessary because if the price of the deployments exceeds U_i, it is not profitable for the



(a) Candidate dep. 1. (b) Candidate dep. 2. (c) Candidate dep. 3. (d) Candidate dep. 4.

Figure 3: Example of AI application component with four deployments.

user to run the application.

To ensure the feasibility of the problem, and assuming $U_i \geq C_i$, the following assumptions (related to the satisfaction of the performance constraint) must hold:

- if energy and memory levels of the devices are enough to run a full local deployment: $D_{I,i}^{(k)} + \frac{\delta_I^{(k)}}{B_i} + D_{II,i}^{(k)} \leq \bar{R}$ for any $k \in \mathcal{D} \setminus \mathcal{D}'$.
- if energy and memory levels of the devices are enough to run the k-th deployment, $k \in \mathcal{D}'$: $D_{I,i}^{(k)} + \frac{\delta_I^{(k)}}{B_i} + D_{II,i}^{(k)} + \frac{\delta_{II}^{(k)}}{B_i} + \min(D_e^{(k)}, D_c^{(k)} + \frac{\delta_{II}^{(k)}}{\tilde{B}}) \leq \bar{R}$ for any $k \in \mathcal{D}'$.

3 STACKELBERG GAME FORMULATION

In the system under study, we model the MEC problem as a one-leader-multi-follower Stackelberg game, where the edge platform is the leader and the users are the followers:

$$\max_{r, n_e^{(k)}, n_c^{(k)}, y_i^e, y_i^c} \quad P_e \tag{19a}$$

$$\text{s.t.} \quad \mathbf{x}_i(r) = \begin{cases} \arg\max\limits_{\mathbf{x}_i} \sum\limits_{k \in \mathcal{D}} x_i^{(k)} U_i - C_i \\ \text{s.t. constraints (5), (13), (14)} \\ \qquad x_i^{(k)} \in \{0,1\} \quad \forall k \in \mathcal{D}, \end{cases} \quad \forall\, i \in \mathcal{U}, \tag{19b}$$

$$\text{constraints (1), (6), (10), (11), (15),} \tag{19c}$$

$$r^{(k)} = r_0 + \gamma^{(k)} r \qquad \forall\, k \in \mathcal{D}', \tag{19d}$$

$$r \in [r_{min}, r_{max}], \quad y_i^e, y_i^c \in \{0,1\}, \quad n_e^{(k)}, n_c^{(k)} \in \mathbb{Z}_+. \tag{19e}$$

In the scenario where the platform has full knowledge of users' parameters, the optimal decision of each user can be anticipated by the platform. From a mathematical point of view, the optimality constraint (19b) can be equivalently reformulated as a system of linear inequalities with additional binary variables (see (47)–(53) in the Appendix which is available as additional material and here [22]). Hence, the Stackelberg game (19a)–(19e) is equivalent to an NP-hard MINLP problem, which poses significant challenges for its solution. This problem can potentially be solved using a global optimization solver. However, due to the presence of non-convex terms like $r^{(k)} x_i^{(k)}$ in the objective function, there is no guarantee of finding an optimal solution. Moreover, given the large number of constraints in (15), limited instances size can be considered, as we show in Section 5.2.

On the other hand, in the partial knowledge scenario, the platform cannot anticipate the optimal decision of users and must interact directly with them to evaluate its objective function P_e. In other words, the platform proposes a price r, users respond with their best reply by selecting a deployment $x_i^{(k)}$, then the platform determines the optimal allocation of computational resources and evaluates the objective function P_e. Therefore, in this case, the Stackelberg game cannot be reformulated as a single MINLP problem, and an interaction between the platform and users is required to find an equilibrium of the game.

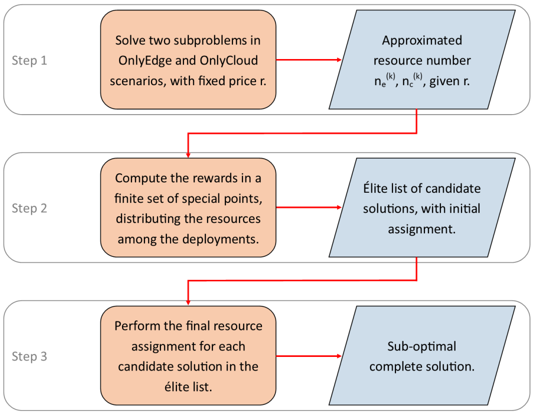

Step 1
Solve two subproblems in OnlyEdge and OnlyCloud scenarios, with fixed price r.
Approximated resource number $n_e^{(k)}$, $n_c^{(k)}$, given r.
Step 2
Compute the rewards in a finite set of special points, distributing the resources among the deployments.
Élite list of candidate solutions, with initial assignment.
Step 3
Perform the final resource assignment for each candidate solution in the élite list.
Sub-optimal complete solution.

Figure 4: Proposed approach in the full knowledge scenario.

4 SOLUTION

To address problem instances of practical interest, we develop a heuristic approach. We provide the solution to the Stackelberg game incrementally: first, we assume the platform has full knowledge of user parameters through the controller agent, enabling it to solve the Stackelberg game in a centralized manner (see Sections 4.1–4.4), then we consider the more realistic scenario where the platform does not know the sensitive parameters of users (Section 4.5).

In the full knowledge scenario, we note that the user problem is an integer linear optimization problem that can be solved by checking a few conditions (see Section 4.1). The first step of our heuristic approach determines the approximate number of edge servers and cloud VMs needed to meet user demands for a fixed price r (see Section 4.2). Then, we demonstrate that the optimal price can be identified by inspecting $O(N)$ special prices, where N is the number of users (see Section 4.3). Finally, we solve the user assignment problem for the $PopSize$ most promising prices and identify the price and resource assignment that maximizes platform profit (see Section 4.4). Figure 4 summarizes the main steps of the proposed heuristic in the full knowledge scenario.

In the partial knowledge scenario, we propose an efficient algorithm to estimate the best platform profit (see Section 4.5). This scenario assumes continuous interaction between users and the edge platform, and the proposed algorithm builds partially on the methods used in the full knowledge scenario.

4.1 Users' algorithm

The user side problem (18a)–(18c) is an integer linear optimization problem that can be solved by checking a few conditions, as depicted in Algorithm 1. The algorithm takes as input the user's parameters including cost-related factors (α_i, β_i, ζ_i), satisfaction U_i, power consumption $p_{j,i}^{(k)}$, energy and memory levels $\bar{E}_{j,i}$ and $\bar{M}_{j,i}$, activation time T_i, and bandwidths $\bar{B}_i$ and B_i, along with the cost of running the application, $r^{(k)}$. The user computes the cost of running each deployment (lines 3-4). The conditions required to execute deployment k are that the user has sufficient energy and memory and that the value U_i of the running application is greater than the cost of executing deployment k (line 5). Since U_i is constant, regardless of the deployment chosen, the user selects the deployment with the lowest cost among those that meet the necessary conditions (lines 6-10).

Algorithm 1 User's algorithm

```
1: Input: $i, \alpha_i, \beta_i, \zeta_i, r^{(k)}, \delta_{II}^{(k)}, p_{j,i}^{(k)}, m_j^{(k)}, \bar{E}_{j,i}, \bar{M}_{j,i}, T_i, \lambda$
2: Initialization: $BestSolution \leftarrow \emptyset$, $BestCost \leftarrow \infty$
3: for all $k \in \mathcal{D}$ do
4:     $C_i^{(k)} = T_i \left(\alpha_i r^{(k)} + (1-\alpha_i)\beta_i(p_{I,i}^{(k)} + p_{II,i}^{(k)})\lambda T_i + \zeta_i \delta_{II}^{(k)} \lambda\right)$
5:     if $(C_i^{(k)} < U_i)$ and $(\lambda T_i^2 p_{j,i}^{(k)} \leq \bar{E}_{j,i})$ and $(m_j^{(k)} \leq \bar{M}_{j,i})$ and $(C_i^{(k)} < BestCost)$ then
6:         $BestCost \leftarrow cost$
7:         $BestSolution \leftarrow k$
8:     end if
9: end for
10: return $BestSolution$
```

In the full knowledge scenario, the AI application control agent supplies the platform with detailed user parameter information. This allows the platform to predict users' decisions centrally, eliminating the need for direct interaction. On the other hand, in the partial knowledge scenario, users directly solve problem (18a)–(18c) and send the solution to the edge platform in response to any proposed price r.

4.2 Relaxed problem with fixed price

The edge platform resource allocation problem, which involves finding the optimal values for $n_e^{(k)}$ and $n_c^{(k)}$, can be approximately reformulated as a convex optimization problem by assuming fixed prices. Given a price, users select the best deployment through Algorithm 1, and the total incoming load $\lambda^{(k)}$, $k \in \mathcal{D}'$, computed as in (7), is partitioned between edge ($\lambda_e^{(k)}$) and cloud ($\lambda_c^{(k)}$), such that

$$\lambda_e^{(k)} + \lambda_c^{(k)} = \lambda^{(k)} \qquad \forall\, k \in \mathcal{D}'. \tag{20}$$

Next, the platform computes the amount of edge and cloud resources ($n_e^{(k)}$, $n_c^{(k)}$) and decides how to split $\lambda^{(k)}$ into the rate of users that will be served by the edge servers and by the cloud VMs. This is achieved through a continuous relaxation of the subproblem obtained by fixing the price r. Notice that, the amount of edge and cloud resources obtained by solving the relaxed problem are rounded up to the next integer to ensure compliance with saturation and time constraints. This approach is widely used in the literature for large systems (e.g., in our setting with tens of VMs in use) [23], [24]. Rounding to the next integer has little impact on the costs.

In Section 4.2.1, we solve the relaxed problem for particular settings that, nonetheless, offer insights into the general case. The application involves a single deployment capable of offloading the AI inference task to the edge or cloud ($|\mathcal{D}'| = 1$). This case is the only one with a closed-form solution to the problem. We will use it to allocate resources to a deployment utilizing both the edge and cloud. In the case of multiple offloading deployments ($|\mathcal{D}'| > 1$, see Section 4.2.2), we first estimate the resources needed in scenarios where only edge servers (*Only Edge* scenario) or only cloud VMs (*Only Cloud* scenario) are available. Then, we decide which deployments to assign completely to the edge, which to the cloud, and which to divide between the two. Finally, we propose an algorithm that, for a fixed price r, estimates the optimal edge and cloud resources to meet user demands (see Section 4.2.3).

4.2.1 Single deployment offloading scenario

The case of single deployment offloading, $|\mathcal{D}'| = 1$, is the only one for which we have a closed-form solution to the problem with a fixed price r. Let $\bar{k}$ be the deployment involving the offloading of resources, and $N^{(\bar{k})}$ be the number of users who selected that deployment. The response time of the platform is redefined as:

$$\bar{\bar{R}} = \bar{R} - \frac{1}{N^{(\bar{k})}} \sum_{i \in \mathcal{U}} \left[D_{I,i}^{(\bar{k})} x_i^{(\bar{k})} + \frac{\delta_I^{(\bar{k})} x_i^{(\bar{k})}}{\bar{B}_i} + D_{II,i}^{(\bar{k})} x_i^{(\bar{k})} + \frac{\delta_{II}^{(\bar{k})} x_i^{(\bar{k})}}{B_i} \right]. \tag{21}$$

The term $\frac{1}{N^{(\bar{k})}} \sum_{i \in \mathcal{U}} \left[D_{I,i}^{(\bar{k})} x_i^{(\bar{k})} + D_{II,i}^{(\bar{k})} x_i^{(\bar{k})} \right]$ denotes the expected demand time of running the first two parts of the DNN on the local devices of users who selected deployment $\bar{k}$. The term $\frac{1}{N^{(\bar{k})}} \sum_{i \in \mathcal{U}} \left[\frac{\delta_I^{(\bar{k})} x_i^{(\bar{k})}}{\bar{B}_i} + \frac{\delta_{II}^{(\bar{k})} x_i^{(\bar{k})}}{B_i} \right]$ represents the expected transmission delay between smart devices and the phone through the Wi-Fi network, plus the delay between the phone and the edge platform through the mobile network. The rationale behind (21) is that to guarantee the server-side response time, it is necessary to provide a margin to account for the user-side processing time and data transmission delay. Hence, the response time constraint (15) is redefined on the edge platform side as:

$$\frac{\lambda_e^{(\bar{k})}}{\lambda^{(\bar{k})}} \cdot \frac{D_e^{(\bar{k})} n_e^{(\bar{k})}}{n_e^{(\bar{k})} - D_e^{(\bar{k})} \lambda_e^{(\bar{k})}} + \frac{\lambda_c^{(\bar{k})}}{\lambda^{(\bar{k})}} \cdot \left[\frac{\delta_{II}^{(\bar{k})}}{\tilde{B}} + \frac{D_c^{(\bar{k})} n_c^{(\bar{k})}}{n_c^{(\bar{k})} - D_c^{(\bar{k})} \lambda_c^{(\bar{k})}} \right] \leq \bar{\bar{R}}. \tag{22}$$

We can prove the following results.

Theorem 4.1. *The response time constraint* (22) *is convex.*

Proof. See Appendix. ☐

To estimate $n_e^{(\bar{k})}$ and $n_c^{(\bar{k})}$, we can formulate a simplified edge platform problem. This is done by neglecting y_i^e and y_i^c, assuming a fixed price for the deployments, and relaxing the integrality constraint on $n_e^{(\bar{k})}$ and $n_c^{(\bar{k})}$, which are now treated as continuous variables:

$$\min_{n_e^{(\bar{k})}, n_c^{(\bar{k})}, \lambda_e^{(\bar{k})}, \lambda_c^{(\bar{k})}} \quad c_e n_e^{(\bar{k})} + c_c n_c^{(\bar{k})} \tag{23a}$$

$$\text{s.t.} \quad \text{constraint (22)},$$

$$n_e^{(\bar{k})} \leq N_e, \tag{23b}$$

$$D_e^{(\bar{k})} \lambda_e^{(\bar{k})} < n_e^{(\bar{k})}, \quad D_c^{(\bar{k})} \lambda_c^{(\bar{k})} < n_c^{(\bar{k})}, \tag{23c}$$

$$\lambda_e^{(\bar{k})} + \lambda_c^{(\bar{k})} = \lambda^{(\bar{k})}, \tag{23d}$$

$$n_e^{(\bar{k})}, n_c^{(\bar{k})} \geq 0, \quad \lambda_e^{(\bar{k})}, \lambda_c^{(\bar{k})} \geq 0. \tag{23e}$$

Since the latter problem has a linear objective function, convex inequality constraints, linear equality constraints, and the Slater constraint qualification holds, the Karush-Kuhn-Tucker (KKT) optimality conditions can be used to find the optimal values of the variables $n_e^{(\bar{k})}$ and $n_c^{(\bar{k})}$. By applying Theorem 4.2, we derive the required resources to run the final part of the DNN on both the edge and cloud, and determine the optimal split of $\lambda^{(\bar{k})}$ into $\lambda_e^{(\bar{k})}$ and $\lambda_c^{(\bar{k})}$. The solutions are then rounded to obtain integer values for $n_e^{(\bar{k})}$ and $n_c^{(\bar{k})}$, and the loads are adjusted proportionally to λ so that an integer number of users are served by either the edge or the cloud. In the following theorem, we assume that the transmission delay between the edge and cloud is negligible ($\delta_{II}^{(\bar{k})}/\tilde{B} \approx 0$).

Theorem 4.2. *If the total load satisfies the following condition*

$$\lambda^{(\bar{k})} \leq \frac{N_e(\bar{\bar{R}} - D_e^{(\bar{k})})}{\bar{\bar{R}} D_e^{(\bar{k})}},$$

then the edge platform does not use cloud resources and the optimal number of edge resources is

$$n_e^{(\bar{k})} = \frac{\bar{\bar{R}} D_e^{(\bar{k})} \lambda^{(\bar{k})}}{\bar{\bar{R}} - D_e^{(\bar{k})}}.$$

Otherwise, if

$$\lambda^{(\bar{k})} > \frac{N_e(\bar{\bar{R}} - D_e^{(\bar{k})})}{\bar{\bar{R}} D_e^{(\bar{k})}},$$

then edge resources are saturated ($n_e^{(\bar{k})} = N_e$), the optimal edge load is

$$\lambda_e^{(\bar{k})} = \frac{N_e \lambda^{(\bar{k})} (\bar{\bar{R}} - \sqrt{D_c^{(\bar{k})} D_e^{(\bar{k})}})}{N_e D_e^{(\bar{k})} + \bar{\bar{R}} \lambda^{(\bar{k})} D_e^{(\bar{k})} - N_e \sqrt{D_c^{(\bar{k})} D_e^{(\bar{k})}}},$$

and the optimal number of cloud resources is

$$n_c^{(\bar{k})} = \frac{D_c^{(\bar{k})} \lambda^{(\bar{k})} [\bar{\bar{R}} D_e^{(\bar{k})} \lambda^{(\bar{k})} - N_e(\bar{\bar{R}} - D_e^{(\bar{k})})]}{N_e(\sqrt{D_e^{(\bar{k})}} - \sqrt{D_c^{(\bar{k})}})^2 + D_e^{(\bar{k})} \lambda^{(\bar{k})} (\bar{\bar{R}} - D_c^{(\bar{k})})}.$$

Proof. See Appendix. ☐

In the most general case, we also consider the transmission delay term between the edge and cloud. Here, there is no closed-form solution to the associated KKT system (see Appendix). Given the high bandwidth for data transfer from edge to cloud, the transmission delay is minimal, making the solution to the system of equations nearly identical to the one given by Theorem 4.2. Therefore, we numerically solve the KKT system, providing the solver with the solution proposed by Theorem 4.2 as the initial estimate.

4.2.2 Multiple deployments offloading scenario

In the following, we seek a solution to the edge platform problem (17a)-(17d) for scenarios involving multiple offloading deployments ($|\mathcal{D}'| > 1$). Following a similar approach as in Section 4.2.1, we aim to simplify the problem by fixing the price r and subsequently solving it through continuous relaxation of $n_e^{(k)}$ and $n_c^{(k)}$. First, we redefine the response time threshold:

$$\bar{\bar{R}} = \bar{R} - \frac{1}{\bar{N}} \sum_{i \in \mathcal{U}} \left[\sum_{k \in \mathcal{D}'} D_{I,i}^{(k)} x_i^{(k)} + \sum_{k \in \mathcal{D}'} \frac{\delta_I^{(k)} x_i^{(k)}}{\bar{B}_i} + \right.$$
$$\left. \sum_{k \in \mathcal{D}'} D_{II,i}^{(k)} x_i^{(k)} + \sum_{k \in \mathcal{D}'} \frac{\delta_{II}^{(k)} x_i^{(k)}}{B_i} \right]. \tag{24}$$

The first and third terms in square brackets represent the demanding times on users' devices, while the second and fourth terms denote the data transmission times, analogous to (21). Specifically, $\bar{N}$ denotes the total number of users selecting an offloading deployment. Recall that user behaviour is deterministic at a fixed price r. Therefore, in the full knowledge scenario, the values of the variables $x_i^{(k)}$ are known for a fixed price r. Once $\bar{\bar{R}}$ is defined, the response time constraint (15) on the edge platform side is reformulated as follows:

$$\sum_{k \in \mathcal{D}'} \frac{\lambda_e^{(k)}}{\lambda^{(k)}} \cdot \frac{D_e^{(k)} n_e^{(k)}}{n_e^{(k)} - D_e^{(k)} \lambda_e^{(k)}} +$$
$$\sum_{k \in \mathcal{D}'} \frac{\lambda_c^{(k)}}{\lambda^{(k)}} \cdot \left[\frac{\delta_{II}^{(k)}}{\tilde{B}} + \frac{D_c^{(k)} n_c^{(k)}}{n_c^{(k)} - D_c^{(k)} \lambda_c^{(k)}} \right] \leq \bar{\bar{R}}. \tag{25}$$

Note that once $x_i^{(k)}$ is known, $L_e^{(k)}$, $L_c^{(k)}$, $n_e^{(k)}$, and $n_c^{(k)}$ depend solely on $\lambda^{(k)}$, which is determined by $x_i^{(k)}$. Furthermore, since the platform can anticipate the users' choice of deployment for a fixed r, the expectations of y_i^e and y_i^c are conditioned by this knowledge. Similarly to Theorem 4.1, the following result can be proven.

Theorem 4.3. *The response time constraint* (25) *is convex.*

Proof. See Appendix. ☐

The simplified platform problem is formulated assuming a fixed price for the deployments, neglecting y_i^e and y_i^c, and relaxing the integrality constraint on $n_e^{(k)}$ and $n_c^{(k)}$:

$$\min_{n_e^{(k)}, n_c^{(k)}, \lambda_e^{(k)}, \lambda_c^{(k)}} \sum_{k \in \mathcal{D}'} c_e n_e^{(k)} + c_c n_c^{(k)} \tag{26a}$$

$$\text{s.t.} \quad \text{constraint (25)},$$

$$\sum_{k \in \mathcal{D}'} n_e^{(k)} \leq N_e, \tag{26b}$$

$$D_e^{(k)} \lambda_e^{(k)} < n_e^{(k)}, \quad D_c^{(k)} \lambda_c^{(k)} < n_c^{(k)} \qquad \forall\, k \in \mathcal{D}', \tag{26c}$$

$$\lambda_e^{(k)} + \lambda_c^{(k)} = \lambda^{(k)} \qquad \forall\, k \in \mathcal{D}', \tag{26d}$$

$$n_e^{(k)}, n_c^{(k)} \geq 0, \quad \lambda_e^{(k)}, \lambda_c^{(k)} \geq 0, \qquad \forall\, k \in \mathcal{D}'. \tag{26e}$$

Note that for an arbitrary number of deployments, the KKT system associated with the latter problem does not have a closed-form solution. However, closed-form solutions can be derived in the *Only Edge* (Theorem 4.4) and *Only Cloud* (Theorem 4.5) scenarios. These solutions provide insights that help in formulating a heuristic approach for scenarios with $|\mathcal{D}'| > 1$ offloading deployments.

Only Edge scenario: Assume $\lambda^{(k)} = \lambda_e^{(k)}$ $\forall k \in \mathcal{D}'$. The edge platform problem (26a)-(26e) becomes:

$$\min_{n_e^{(k)}} c_e \sum_{k \in \mathcal{D}'} n_e^{(k)}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{D}'} n_e^{(k)} \leq N_e,$$

$$\sum_{k \in \mathcal{D}'} \frac{D_e^{(k)} n_e^{(k)}}{n_e^{(k)} - D_e^{(k)} \lambda^{(k)}} \leq \bar{\bar{R}},$$

$$n_e^{(k)} - D_e^{(k)} \lambda^{(k)} > 0 \qquad \forall\, k \in \mathcal{D}'.$$

In the following theorem, we assume that there are more edge servers than needed to serve all deployments, so $\sum_{k \in \mathcal{D}'} n_e^{(k)} < N_e$. This allows us to estimate the number of servers required to satisfy the response time constraint for all users.

Theorem 4.4. *If*

$$\sum_{k\in\mathcal{D}'} D_e^{(k)}\sqrt{\lambda^{(k)}}\left(\sqrt{\lambda^{(k)}} + \frac{\sum_{m\in\mathcal{D}'} D_e^{(m)}\sqrt{\lambda^{(m)}}}{\bar{\bar{R}} - \sum_{m\in\mathcal{D}'} D_e^{(m)}}\right) < N_e,$$

then the optimal edge resources are given by:

$$n_e^{(k)} = D_e^{(k)}\lambda^{(k)} + D_e^{(k)}\sqrt{\lambda^{(k)}} \cdot \frac{\sum_{m\in\mathcal{D}'} D_e^{(m)}\sqrt{\lambda^{(m)}}}{\bar{\bar{R}} - \sum_{m\in\mathcal{D}'} D_e^{(m)}} \qquad \forall\, k \in \mathcal{D}'.$$

Proof. See Appendix. ☐

Only Cloud scenario: Assume $\lambda^{(k)} = \lambda_c^{(k)}\ \forall k \in \mathcal{D}'$. The edge platform problem (26a)-(26e) becomes:

$$\min_{n_c^{(k)}} c_c \sum_{k\in\mathcal{D}'} n_c^{(k)}$$

$$\text{s.t.} \sum_{k\in\mathcal{D}'} \left(\frac{\delta_{II}^{(k)}}{\bar{B}} + \frac{D_c^{(k)} n_c^{(k)}}{n_c^{(k)} - D_c^{(k)}\lambda_c^{(k)}}\right) \leq \bar{\bar{R}},$$

$$n_c^{(k)} - D_c^{(k)}\lambda^{(k)} > 0 \qquad \forall\, k \in \mathcal{D}'.$$

The following theorem computes the number of cloud VMs required to serve all deployments.

Theorem 4.5. *The optimal cloud resources are given by:*

$$n_c^{(k)} = D_c^{(k)}\sqrt{\lambda^{(k)}}\left[\sqrt{\lambda^{(k)}} + \frac{\sum_{m\in\mathcal{D}'} D_c^{(m)}\sqrt{\lambda^{(m)}}}{\bar{\bar{R}} - \sum_{m\in\mathcal{D}'}\left(D_c^{(m)} + \frac{\delta_{II}^{(m)}}{\bar{B}}\right)}\right] \ \forall\, k \in \mathcal{D}'.$$

Proof. See Appendix. ☐

Note that in both scenarios, the values obtained are rounded up to the next integer. In this way, we ensure that the number of servers/VMs meets saturation and time constraints. These results allow us to develop a heuristic algorithm for assigning edge/cloud resources to different deployments. Notice that, the final solution depends on the order with which we assign deployments to edge ($n_e^{(k)}$) or cloud ($n_c^{(k)}$). Each order is one of the possible $|\mathcal{D}'|!$ permutations of deployments that offload part of the computation. Since edge servers are cheaper than cloud VMs, we prioritize assigning deployments to the edge first until it is saturated by relying on the result of Theorem 4.4, then assign to the cloud according to the established order. Note that no specific order guarantees an optimal solution a priori. As a baseline, we consider a "combinatorial" approach, which searches for the optimal solution among all possible permutations in the order of assigning deployments to the edge and cloud. To ensure the most time-efficient algorithm is used, we focus on a specific ordering in the experimental section and compare these results with those of the combinatorial approach.

4.2.3 Optimal resource numbers algorithm

In this section, we present the algorithm that estimates the optimal number of edge servers and cloud VMs given a fixed price r. Specifically, Algorithm 2 relies on the theorems stated in Sections 4.2.1 and 4.2.2.

The algorithm receives as input *order*, an ordered list of deployment indices, the loads $\lambda^{(k)}$, and the number of available edge servers N_e. The optimal number of edge servers is computed considering the *Only Edge* scenario (line 3). If the edge servers are sufficient, all the loads are assigned to the edge (lines 4-5). Otherwise, the optimal number of cloud VMs is computed considering the *Only Cloud* scenario (line 7). The allocation of deployments is initially done by assigning all deployments to the edge and calculating the number of edge servers required to meet demand (line 8). If this number exceeds the number of available servers N_e, the last deployment in *order* assigned to the edge is moved to the cloud (lines 9-13). In this way, the last deployment moved to the cloud is the one that is split between the remaining resources of the edge, $\bar{n}_e$, and the cloud VMs, according to the single deployment offloading scenario (line 15). Finally, the load of the remaining deployments, i.e. the first in the order, is assigned to the edge (lines 16-19), setting to zero their cloud VMs (initially computed at line 7).

Algorithm 2 Optimal Resource Numbers

```
1: Input: $order$, $\lambda^{(k)}$, $N_e$
2: Compute $\bar{\bar{R}}$ as in (24)
3: $n_e^{(k)} \leftarrow$ Solution through Theorem 4.4 $\forall k \in \mathcal{D}'$
4: if $\sum_{k\in\mathcal{D}'} n_e^{(k)} \leq N_e$ then
5:     $\lambda_e^{(k)} \leftarrow \lambda^{(k)}$,   $\lambda_c^{(k)} \leftarrow 0$,   $n_c^{(k)} \leftarrow 0 \quad \forall k \in \mathcal{D}'$
6: else
7:     $n_c^{(k)} \leftarrow$ Solution through Theorem 4.5 $\forall k \in \mathcal{D}'$
8:     $UsedEdgeVMs \leftarrow \sum_{k\in\mathcal{D}'} n_e^{(k)}$, $idx \leftarrow |\mathcal{D}'|$
9:     while $UsedEdgeVMs > N_e$ do
10:        $idx \leftarrow idx - 1$, $dep \leftarrow order[idx]$
11:        $UsedEdgeVMs \leftarrow UsedEdgeVMs - n_e^{(dep)}$
12:        $\lambda_c^{(dep)} \leftarrow \lambda^{(dep)}$, $n_e^{(dep)} \leftarrow 0$
13:    end while
14:    $\bar{n}_e \leftarrow N_e - \sum_{k\in\mathcal{D}'} n_e^{(k)}$
15:    $n_e^{(dep)}, \lambda_e^{(dep)}, n_c^{(dep)}, \lambda_c^{(dep)} \leftarrow$ Solution through Theorem 4.2,
       with $\bar{n}_e$
16:    for $k \leftarrow 1$ to $dep - 1$ do
17:        $idx \leftarrow order[k]$
18:        $\lambda_e^{(idx)} \leftarrow \lambda^{(idx)}$, $n_c^{(idx)} \leftarrow 0$
19:    end for
20: end if
21: return $(n_e^{(k)}, n_c^{(k)}, \lambda_e^{(k)}, \lambda_c^{(k)})$
```

4.3 Optimal prices

In the previous section, we assumed a fixed price for the deployments and determined a nearly optimal number of resources using KKT conditions. To solve the Stackelberg game, we need to find the optimal price, which we prove is located in a finite set of special points. Therefore, the optimal value can be identified by inspection. For instance, the platform profit function with 10 users and 4 deployments is shown in Figure 5. By varying the price, the platform profit exhibits discontinuities. These discontinuities occur when a small change in price forces users to either drop from the system or change their deployment. The cost for user i to run deployment k is computed as in Algorithm 1 (line 4). Accordingly, we define two types of discontinuity points:

- **Dropping points:** these are the price r values at which the utility of user participation drops to zero, indicating that the user has no motivation to run the application. If the price r that the user is willing to pay for running the application is less than the cost of the cheapest available deployment, the user chooses to drop from the system

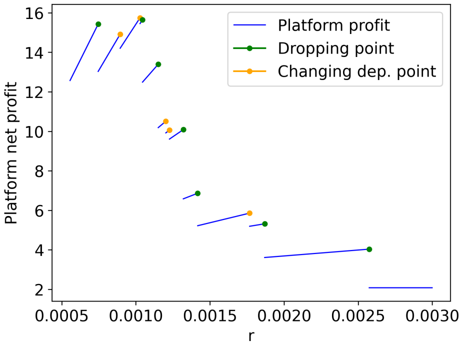


Figure 5: Platform profit function with 10 users and 4 deployments.

and stop using the application. Therefore, the dropping points for user i are computed by setting $U_i = C_i$ for the offloading deployments. For any $k \in \mathcal{D}'$, the dropping points are computed as follows:

$$drop_k = \frac{1}{T_i \alpha_i \gamma^{(k)}} \{U_i - T_i [\alpha_i r_0 \\ +(1-\alpha_i)\beta_i \left(p_{g,i}^{(k)} + p_{p,i}^{(k)}\right) \lambda T_i + \zeta_i \delta_{II}^{(k)} \lambda\Big]\Big\}. \quad (27)$$

- **Changing deployment points:** these points specify the prices at which the utility of running two different deployments becomes equal, making a user indifferent between the choices. Let $C_i^{(k)}$ denote the cost of running the k-th deployment. The user's utility of running the j-th and l-th deployments are $U_i - C_i^{(j)}$ and $U_i - C_i^{(l)}$, respectively. Therefore, the values of r that make the utilities of running two deployments equal for user i are determined by setting $C_i^{(j)} = C_i^{(l)}$. The changing deployment points are calculated as follows:

$$change_{jl} = \frac{(1-\alpha_i)\beta_i \lambda T_i \left[\left(p_{g,i}^{(l)} + p_{p,i}^{(l)}\right) - \left(p_{g,i}^{(j)} + p_{p,i}^{(j)}\right)\right]}{\alpha_i(\gamma^{(j)} - \gamma^{(l)})} + \\ \frac{\zeta_i \lambda (\delta_{II}^{(l)} - \delta_{II}^{(j)})}{\alpha_i(\gamma^{(j)} - \gamma^{(l)})} \quad \forall j, l \in \mathcal{D}, j > l, \{j, l\} \nsubseteq \mathcal{D} \setminus \mathcal{D}'. \quad (28)$$

Note that changing deployment points are computed for $j > l$ to avoid repetitions and are not computed for local deployments whose price is independent of r.

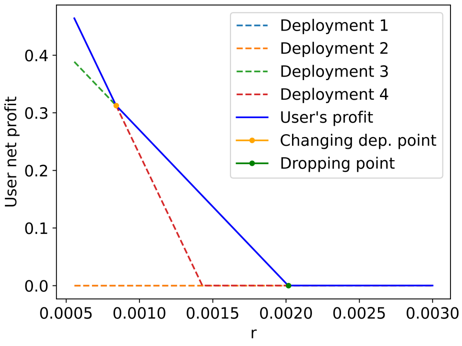


Figure 6: User profit profile with 4 deployments.

An example of a user's profit function is shown in Figure 6. In this case, there is a changing deployment point, $change_{43}$, where the user is indifferent between deployments 3 and 4, and a dropping point, $drop_3$, where the user's profit is zero. The profit related to the first and second deployments is zero for any r because the user devices do not have sufficient power or memory to run the full DNN locally. Now, we state the fundamental theorem of our solution approach.

Algorithm 3 Optimal Prices Algorithm

```
1: Input: $N$, users' parameters, $r_{\min}$, $r_{\max}$, $r_0$, $\gamma^{(k)}$, $\lambda$, $PopSize$, $ordering$
2: Initialization: $Solutions \leftarrow []$
3: for $i \leftarrow 1$ to $N$ do
4:     Compute dropping and changing deployment points for user $i$
5:     for $r$ in $\{drop_i, drop_i - \epsilon, change_{jl}, change_{jl} - \epsilon\}$ do
6:         if $r_{min} \leq r \leq r_{max}$ then
7:             $r^{(k)} \leftarrow r_0 + \gamma^{(k)} r \ \forall k \in \mathcal{D}'$
8:             $x \leftarrow$ Solve user model given $r$
9:             $\lambda^{(k)} \leftarrow \sum_{i \in \mathcal{U}} \lambda x_i^{(k)} \ \forall k \in \mathcal{D}'$
10:            $orders \leftarrow$ Compute orders given ordering
11:            for $order$ in $orders$ do
12:                $n_e^{(k)}, n_c^{(k)}, \lambda_e^{(k)}, \lambda_c^{(k)} \leftarrow$ OPTIMALRESOURCENUMBERS($order, \lambda^{(k)}, N_e$)
13:                $P \leftarrow$ Compute platform profit given $r, x, n_e^{(k)}, n_c^{(k)}$
14:                $Solutions$.append $(P, r, x, n_e^{(k)}, n_c^{(k)}, \lambda_e^{(k)}, \lambda_c^{(k)}, order)$
15:            end for
16:        end if
17:    end for
18: end for
19: sort $Solutions$ by $P$ decreasingly
20: return first $PopSize$ elements of $Solutions$
```

Theorem 4.6. *The optimal price solution lays at one of the points of discontinuity or in a left neighborhood of a point of discontinuity.*

Proof. See Appendix. □

Theorem 4.6 states that we only need to inspect the $O(N)$ points of discontinuity to find the optimal price (see Algorithm 3). Note, however, that the values for the number of resources n_e and n_c obtained in Section 4.2 are not guaranteed to be optimal, as they were computed by considering an average response time (24). For this reason, since the cost of running the infrastructure is a first approximation in the net profit evaluation, not only the best price among the points of discontinuity but also an elite set of optimal solutions is considered.

Algorithm 3 estimates the profit for all possible discontinuity points r and returns the most promising ones before allocating resources to users. Specifically, it takes as input the number of users N, the users' parameters, minimum and maximum r, the fixed price r_0, $\gamma^{(k)}$, λ, the number $PopSize$ of elite solutions, and *ordering*. The *ordering* specifies whether resource allocation is done in a predetermined order or by considering all possible $|\mathcal{D}'|!$ permutations of deployments (see Section 4.2.2). For each user i, the points of discontinuity, $drop_k$ and $change_{jl}$, are computed (line 4). For each discontinuity point r in the range $[r_{\min}, r_{\max}]$, Algorithm 1 solves the users' problem to obtain their deployment choice $x_i^{(k)}$ and computes the total loads for each deployment (lines 8-9). For each considered order, Algorithm 2 computes the approximate $n_e^{(k)}$, $n_c^{(k)}$, $\lambda_e^{(k)}$, $\lambda_c^{(k)}$, and a preliminary estimate of the platform profit is obtained (lines 10-15). Each result is stored in the $Solutions$ list, which is finally sorted by profit P. The best $PopSize$ elements are returned (lines 19-20).

Algorithm 4 Users' resource assignment

```
1: **Input:** $EliteSols$, $\bar{R}$, $D^{(k)}_{j,i}$, $\delta^{(k)}_j$, $\bar{B}_i$, $B_i$, $\tilde{B}$, $D^{(k)}_e$, $D^{(k)}_c$, $\lambda$, $N_e$
2: **Initialization:** $BestProfit \leftarrow 0$, $BestSol \leftarrow \emptyset$, $y^e_i, y^c_i \leftarrow 0\ \forall i \in \mathcal{U}$, $\bar{n}^{(k)}_e, \bar{n}^{(k)}_c \leftarrow 0\ \forall k \in \mathcal{D}'$
3: **for all** $Sol$ **in** $EliteSols$ **do**
4:     $r, x, n^{(k)}_e, n^{(k)}_c, \lambda^{(k)}_e, \lambda^{(k)}_c, order \leftarrow Sol$
5:     $N^{(k)} \leftarrow \{i \in \mathcal{U} : x^{(k)}_i = 1\}\ \forall k \in \mathcal{D}'$
6:     $V^{(k)} \leftarrow \{\}\ \forall k \in \mathcal{D}'$
7:     **for** $k \in \mathcal{D}'$ **do**
8:         COMPUTEEDGESERVERS($V^{(k)}$, $N^{(k)}$, $\bar{R}$, $D^{(k)}_{j,i}$, $\delta^{(k)}_j$, $\bar{B}_i$, $B_i$, $D^{(k)}_e, n^{(k)}_e, n^{(k)}_c, \lambda^{(k)}_e, \lambda^{(k)}_c, y^e_i, y^c_i, \bar{n}^{(k)}_e, \bar{n}^{(k)}_c$)
9:     **end for**
10:    **if** $\sum_{k \in \mathcal{D}'} \bar{n}^{(k)}_e > N_e$ **then**
11:        $idx \leftarrow 0, UsedVMs \leftarrow 0$
12:        **while** $UsedVMs \leq N_e$ **do**
13:            $idx \leftarrow idx + 1, dep \leftarrow order[idx]$
14:            $UsedVMs \leftarrow UsedVMs + \bar{n}^{(dep)}_e$
15:        **end while**
16:        $\bar{n}_e \leftarrow N_e - (UsedVMs - \bar{n}^{(dep)}_e)$
17:        **while** $\bar{n}^{(dep)}_e > \bar{n}_e$ **do**
18:            MOVEUSER($V^{(dep)}, N^{(dep)}, \bar{R}, D^{(dep)}_e, \lambda^{(dep)}_e, \lambda^{(dep)}_c, \lambda, y^e_i, y^c_i, \bar{n}^{(dep)}_e$)
19:        **end while**
20:        **for** $k \leftarrow idx + 1$ **to** $|\mathcal{D}'|$ **do**
21:            $dep \leftarrow order[k]$
22:            **if** $\bar{n}^{(dep)}_e > 0$ **then**
23:                MOVEDEPLOYMENT($N^{(dep)}$, $\lambda^{(dep)}_e$, $\lambda^{(dep)}_c$, $y^e_i$, $y^c_i$, $\bar{n}^{(dep)}_e$)
24:            **end if**
25:        **end for**
26:    **end if**
27:    **for** $k \in \mathcal{D}'$ **do**
28:        COMPUTECLOUDVMS($V^{(k)}$, $N^{(k)}$, $\bar{R}$, $\delta^{(k)}_{II}$, $\tilde{B}$, $D^{(k)}_c$, $\lambda^{(k)}_c$, $y^c_i, \bar{n}^{(k)}_c$)
29:    **end for**
30:    $P \leftarrow$ Compute platform profit given $r, x, \bar{n}^{(k)}_e, \bar{n}^{(k)}_c$
31:    **if** $P > BestProfit$ **then**
32:        $BestProfit \leftarrow P$
33:        $BestSol \leftarrow (P, r, \bar{n}^{(k)}_e, \bar{n}^{(k)}_c, y^e_i, y^c_i)$
34:    **end if**
35: **end for**
36: **return** $BestSol$
```

4.4 Assigning the resources to the users

Hitherto, we have computed some candidate optimal prices based on Theorem 4.6 and Algorithm 3. As the final step, Algorithm 4 assigns the load of users individually to the edge or cloud.

The algorithm receives the list of elite solutions returned by Algorithm 3, the response time $\bar{R}$, the users' demands $D^{(k)}_{j,i}$, $j \in \{I, II\}$, and their network bandwidths $\bar{B}_i$ and B_i as inputs. For each candidate solution in the list, users' loads are properly assigned to the edge/cloud to achieve a near-optimal and feasible solution to the Stackelberg game. The necessary information is extracted from each candidate solution (line 4), and the indices of the users who select an offloading deployment are stored in $N^{(k)}$ (line 5). For each deployment $k \in \mathcal{D}'$, the vector $V^{(k)}$ is initialized (line 6). $V^{(k)}_i$ stores the *local execution time* of user i who selected deployment k, which is the sum of the demanding times on their local devices, plus the transfer times between the local devices and between the smartphone and the edge platform. The vector $V^{(k)}$ is filled within COMPUTEEDGESERVERS. This procedure computes the number of edge servers needed for each deployment to satisfy the users' time constraints, considering the user (assigned to the edge) with the shortest remaining time (lines 7-9). In this way, if the user with the most restrictive constraint is satisfied, all other users are satisfied as well. Note that, if Algorithm 2 estimates $n^{(k)}_c > 0$, the procedure assigns users to the cloud for that deployment k according to the computed load $\lambda^{(k)}_c$. If the constraint of the maximum number of edge servers is not satisfied (line 10), deployments are assigned first to the edge and then to the cloud, following the same order as in Algorithm 3. The deployment that has to be served by both edge and cloud is identified (lines 11-15), and the users with the lowest $V^{(k)}_i$ who chose that deployment are moved one-by-one from edge to cloud (lines 17-19), through MOVEUSER. The procedure recomputes the optimal number of edge servers needed for that deployment each time a user is moved. The remaining deployments are fully served by the cloud, so they are directly assigned there (lines 20-25), through MOVEDEPLOYMENT. Once edge resources have been allocated, the optimal number of cloud VMs to meet user time constraints is computed (lines 27-29). The procedure COMPUTECLOUDVMS calculates the optimal number of cloud VMs by considering, for each deployment, the user (assigned to the cloud) with the least amount of time remaining, similar to how it was done for edge servers. Finally, the profit is calculated for each candidate solution according to the resource allocation (lines 30-34), and the best one is returned. The pseudocode for the procedures is provided in the Appendix. In summary, the heuristic approach combines Algorithms 3 and 4. The complexity of the algorithm depends on the number of users N running the application and the number of offloading deployments $|\mathcal{D}'|$. The most computationally intensive operation is the sorting of candidate solutions, performed in Algorithm 3 at line 19. Given the loops over users (line 3) and candidate discontinuity points (line 5), the number of candidate solutions is proportional to $N|\mathcal{D}'|$. Therefore, the overall complexity of the algorithm is $O\left(N|\mathcal{D}'|\log(N|\mathcal{D}'|)\right)$.

4.5 Partial knowledge of users' parameters

So far, we have considered the case where the platform knows all users' parameters through the controller agent. In a more realistic scenario, the provider does not know sensitive parameters such as the costs β_i, ζ_i, and the value of running the application U_i. In this section, we describe a procedure to estimate the best platform price with partial knowledge of users' parameters. Specifically, we propose an algorithm that allows the platform to explore the most promising prices r to maximize its profit. We call this scenario "partial knowledge" since the platform needs to know $D^{(k)}_{j,i}$, with $j \in \{I, II\}$, B_i, and $\bar{B}_i$, which are used in resource allocation, and also T_i, necessary to compute the platform profit (17a). While delays due to execution on local devices and data transmission can be easily measured through runtime measurements and/or code instrumentation, the assumption of knowing users' activation time is not restrictive when considering applications such as gaming or map navigation. In this scenario, the platform cannot anticipate users' decisions or calculate discontinuity points. Hence, the Stackelberg game cannot be reformulated as a single MINLP problem, and a heuristic approach is required to find an equilibrium of the problem. Our approach is to

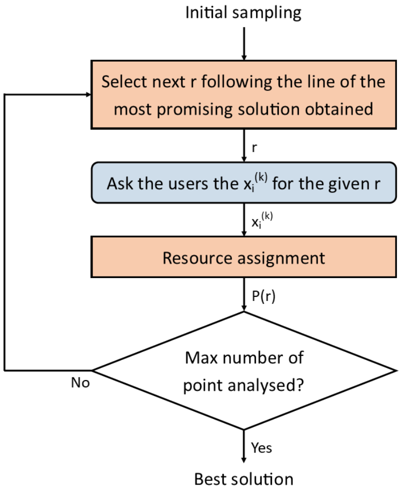

Initial sampling
Select next r following the line of the most promising solution obtained
r
Ask the users the $x_i^{(k)}$ for the given r
$x_i^{(k)}$
Resource assignment
P(r)
Max number of point analysed?
No
Yes
Best solution

Figure 7: Follow the Line approach in the partial knowledge scenario.

propose a set of prices r to the users, who then respond by selecting a deployment. Based on users' choices $x_i^{(k)}$, the platform calculates its profit using the same procedure as in the heuristic algorithm under full knowledge. The search for the best price r is conducted in a greedy manner, evaluating each proposed price based on previous ones. Figure 7 illustrates how the platform estimates the optimal r to maximize its profit. From an implementation perspective, we assume that the mobile application includes a settings panel where users can configure parameters related to their costs and the economic value they assign to running the application. To protect privacy, this data is not transmitted to the platform directly. Instead, the control agent can autonomously respond to proposed prices r by solving the user's optimization problem (18a)–(18c) to maximize their profit. Moreover, the number of distinct r values analyzed is scaled with the number of concurrent users. This approach aims to minimize interactions with users while ensuring high-quality service. Note that the data transferred for this initial setup is negligible compared to the data transferred $\delta_j^{(k)}$ during application runtime.

We call Algorithm 5 "Follow the line" because its objective is to improve the solution by moving along the lines of the platform profit function. Specifically, the algorithm advances from the most promising r observed, i.e., the point yielding the highest profit. For each r analyzed, the platform queries the controller agent for the deployment choice $x_i^{(k)}$ and compares the new user response with that of the previous r point. This allows the algorithm to determine if the new point lies on the same line as its predecessor, where discontinuities indicate changes in user deployment choices or dropout from the system. When consecutive points lie on the same trajectory, meaning no user has changed their choice, the algorithm increases the step size to speed-up the search for maximum profit, as depicted in Figure 8.

Algorithm 5 requires five parameters to be set:

- Cutting fraction cut: this parameter reduces the initial sampling domain to $[r_{\min} + cut \cdot L, r_{\max} - cut \cdot L]$, where $cut \in [0, 0.5)$ and $L = r_{\max} - r_{\min}$. By doing so, the initial points are selected within a central and narrower interval of the r-domain, as it is expected that the maximum profit does not occur at the extremes of the interval. This does not prevent the algorithm from considering points outside this range.
- First sampling approach: the initial points are selected either randomly or equidistantly.
- Epsilon scale ϵ_{scale}: scaling factor which determines the initial step size, calculated as $\epsilon = (r_{\max} - r_{\min})/N$.
- Fraction of initial points η_{init}: points picked in the initial sampling with respect to N.
- Fraction of total points η_{tot}: maximum samples and interactions with users with respect to N.

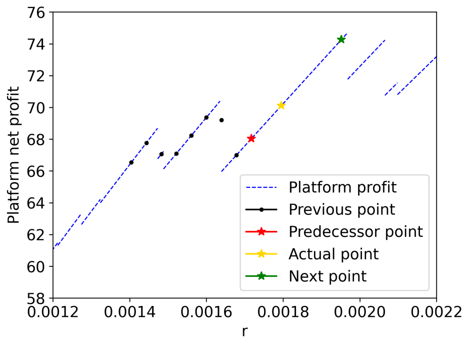


Figure 8: Intuition behind the Follow the Line approach: try to move forward along the line of the most promising r.

Figure 9 illustrates the parameter settings of Algorithm 5 in a simple scenario. Here, $cut = 0.3$, *First sampling approach* is "equispaced", $\epsilon_{scale} = 4$, $\eta_{init} = N/10$ and $\eta_{tot} = N/5$.

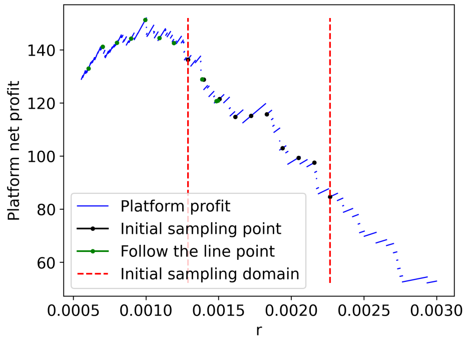


Figure 9: Application of Algorithm 5 in a simple scenario.

As the initial stage of Algorithm 5, the controller agents share the necessary parameters for computing the platform profit and estimating the number of edge servers and cloud VMs (line 3). The extremes of the initial sampling interval, r_{min} and r_{max}, along with the initial step ϵ and the initial points, are computed (lines 4-6). These initial points undergo analysis via Algorithm 6 (see Appendix), providing a starting point for optimization. For each initial point, the profit P and users' choices $x_i^{(k)}$ are collected along with candidate following points in *fp* (lines 7-11). Specifically, each item in *fp* includes the candidate r value being analyzed in the current step, the profit and users' choices stored from the previous point. This information is crucial for generating new candidates. Regarding the initial points, subsequent points are generated by adding and subtracting the initial step ϵ. Once all initial points have been evaluated, new candidates are sorted based on the profit of the points that generated them (line 13). Only the most promising candidates are retained for analysis, with each iteration extracting information from the first element of *fp* until the maximum number of points (max_num_points) to be visited is reached (lines 14-15). The estimated platform

Algorithm 5 Follow the line approach

```
1: Input: cut, first sampling approach, $\epsilon_{scale}$, $\eta_{init}$, $\eta_{tot}$, $N$, $r_{min}$, $r_{max}$
2: Initialization: fp ← [], r_analyzed ← [], solutions ← []
3: Collect $T_i$, $D_{j,i}^{(k)}$, $B_i$, $\bar{B}_i$ $\forall i \in \mathcal{U}$
4: $\epsilon \leftarrow [(r_{max} - r_{min})/N] \cdot \epsilon_{scale}$
5: $L \leftarrow r_{max} - r_{min}, r_{min} \leftarrow r_{min} + L \cdot cut, r_{max} \leftarrow r_{max} - L \cdot cut$
6: initial_points ← pick $N \cdot \eta_{init}$ points in a first sampling approach way in $[r_{min}, r_{max}]$
7: for $r$ in initial_points do
8:     sol ← UserRequestAndProfitEstimate($r$, $r_0$, $\gamma^{(k)}$, $\lambda$, ordering)
9:     Extract $P, x$ from sol and store $r$ in r_analyzed and sol in solutions
10:    Add $[r + \epsilon, P, \epsilon, x]$ and $[r - \epsilon, P, \epsilon, x]$ to fp
11: end for
12: max_num_points ← $N \cdot \eta_{init}$
13: Sort fp by P
14: while max_num_points < $N \cdot \eta_{tot}$ do
15:    r, old_P, step, old_x, ← First element of fp
16:    if $r$ not in r_analyzed then
17:        sol ← UserRequestAndProfitEstimate($r$, $r_0$, $\gamma^{(k)}$, $\lambda$, ordering)
18:        Extract $P, x$ from sol and store $r$ in r_analyzed and sol in solutions
19:        Compare $x$ with old_x to verify if the new point is on the same line
20:        if line changed then
21:            if |step| > $\epsilon$ then
22:                Insertion sort of $[r - step/2, P, -step/2, x]$ in fp
23:            end if
24:            Insertion sort of $[r + \text{sign}(step) \cdot \epsilon, P, \text{sign}(step) \cdot \epsilon, x]$ in fp
25:        else
26:            Insertion sort of $[r + 2 \cdot step, P, 2 \cdot step, x]$ in fp
27:        end if
28:        max_num_points ← max_num_points + 1
29:    end if
30:    Delete the actual point from fp
31: end while
32: BestSol ← Resource assignment through Algorithm 4, given the solution which maximize the profit in solutions
33: return (BestProfit, r)
```

profit P and user choices $x_i^{(k)}$ are gathered by Algorithm 6 and stored in the same manner as the initial points (lines 17-18). The choices of users for the selected r are compared with those from the point that generated it to determine if they lie on the same line (line 19). If they do, the current point generates a new candidate that doubles the distance from the previous step (line 26). Conversely, if at least one choice has changed, the step size is halved, and new candidates are selected to the left and right of the current point (lines 21-24). This process updates the list of candidates *fp* after each analysis by adding new candidates and removing items that have already been evaluated (line 30). Finally, the most promising solution is passed to Algorithm 4 as $EliteSol$, where resources are allocated to users and the best solution is returned (lines 32-33). Note that candidate points are sorted based on the profit estimate provided by Algorithm 6 (see Appendix), without proceeding to allocate resources to users using Algorithm 2. Experimental analysis has shown that, in addition to being faster, the profit difference obtained by Algorithm 4 typically ranges from -1% to +3.5% compared to attempting resource allocation at each point r. This suggests that the estimate is sufficiently accurate.

Finally, concerning the complexity of the "Follow the line" algorithm, the most computationally intensive operation is the insertion sort, which has quadratic complexity in the number of elements. Since this number is a fraction of the number of users N, the average complexity is $O(N \cdot \eta_{tot}(N^2 + |\mathcal{D}'|))$, where $|\mathcal{D}'|$ is the complexity of the function at line 17.

5 EXPERIMENTAL RESULTS

In this section, we present numerical experiments to evaluate the performance of the algorithms and approaches proposed in Section 4. First, in Section 5.1, we describe the experimental setup. Then, in Section 5.2, we assess the performance of the heuristic algorithm by comparing it with BARON [15], a state-of-the-art mixed-integer nonlinear global solver, and exxamine their scalability under the assumption of full knowledge of users' parameters. In Section 5.3, we analyze user behavior in specific scenarios. Additionally, in Section 5.4, we highlight the convenience of solving the Stackelberg game versus imposing a fixed price. Finally, in Section 5.5, we report the performance of the "Follow the Line" algorithm under the assumption of partial knowledge of users' parameters.

5.1 Experimental setting

In this section, we present the parameter settings for our numerical analysis, all of which are inspired by real scenarios. We assume that the first two deployments ($k = 1, 2$) split the inference of the DNN solely among local devices, differing in how the load is distributed. The other deployments ($k \in \mathcal{D}'$) involve a tripartition of the DNN, with the third part of the application being offloaded. For the transmission delay, we consider the upload bandwidth B_i of 5G networks in Italy [25]. The electricity cost β_i is based on the cost in Italy in October 2022 [26], ranging from 0.491 to 0.521 \$/kWh. The cloud VM cost per second c_c is determined by considering current costs from major cloud providers for medium-range GPU-based VMs, such as AWS [18] and Azure [19]. The edge server cost per second c_e is set proportionally to the cloud VM cost. We assume, for simplicity, that users' devices have sufficient memory $\bar{M}_i$ to run any deployment. The energy $\bar{E}_i$ is set within a range of values between 90% of the energy required to run the last deployment ($k = |\mathcal{D}|$), i.e., the one with the minimum energy consumption on local devices, and 1.2 times the energy needed to run the application entirely on the device, inspired by [10]. This ensures a mix of users, some of whom lack sufficient energy to run the application, and others for whom all deployments are available. System parameters are inspired by real application data gathered from an extensive profiling campaign of DNN-based applications for object recognition and segmentation. In particular, we consider the YOLOv4 neural network [27]. The DNN was profiled on a Jetson TX2 Edge device, while the edge/cloud side processing times were obtained on an NVIDIA Pascal GPU server. Regarding the bandwidth between smart devices and phones $\bar{B}_i$, we estimate the network time $\delta_{gp}/\bar{B}_i$ by considering the average between the maximum Wi-Fi 7 and Wi-Fi 6 data rates [28], which is 27.8 Gbps. For the connectivity between smartphones and edge platforms, we use 25.1 Mbps, which is the maximum upload speed for 5G connections in Italy [25]. The results from the profiling of the YOLOv4 DNN are shown in Appendix (Figure 15). Regarding the execution time T_i, half of the users use the application for about ten minutes, and the other half for twenty minutes, while the time horizon T for the platform is set to one hour.

In the following analyses, we refer to the heuristic algorithm that considers all possible *orders* of assignment of the partitioned deployment as *combinatorial*. We use this version of the heuristic algorithm as the reference solution for our experiments. To compare the different approaches, we consider the Profit Ratio, defined as:

$$\text{Profit ratio} = \frac{\text{Profit proposed approach} - \text{Profit reference}}{\text{Profit reference}}, \quad (29)$$

All experiments were performed on a Linux server with a 40-core Intel(R) Xeon(R) CPU, running at 2.20 GHz, and 64 GB of memory. We used the BARON 22.3.21 solver [15] with its multithreading option enabled to maximize performance. The source code for reproducing our numerical analysis is available on Zenodo [22]. To speed up execution, the proposed algorithms were parallelized. For Algorithm 3, the *for* loop between lines 3 and 18 was distributed across the available cores. In Algorithm 5, the *for* loop between lines 7 and 11 was parallelized to process all initial r prices simultaneously, while the *while* loop between lines 14 and 31 was executed in a single thread due to the "Insertion sort" operations performed when new candidates are generated.

To consider different scenarios, users and edge platform parameters, as summarized in Tables 5 and 6 (see Appendix), are adjusted randomly within a range of $\pm 5\%$. Note that in all the following experiments, the results are averaged over ten different instances.

5.2 Performance evaluation of the heuristic algorithm under full knowledge of users' parameters

In this section, we evaluate the performance of the heuristic algorithm under the assumption of full knowledge (Algorithms 3 and 4). As previously discussed, the algorithm requires the parameter *ordering*, which specifies the sequence for assigning deployments first to the edge and then to the cloud. We test several settings, such as ordering based on energy consumption on the edge, inverse sorting (assigning the most expensive deployments to the edge first), and an ordering that considers the number of extra cloud VMs needed to meet the demand for new users. Sorting the deployments in descending order with respect to the demanding time $D_e^{(k)}$ proves to be the most effective. We refer to this setting as the *chosen order*. Figure 10 illustrates the execution times of the chosen order and the combinatorial approach, alongside their respective profit ratios, assuming $|\mathcal{D}| = 5$. The figure shows the expected trends for the execution time, as discussed in Section 4.4. In this scenario, the execution time of the chosen order is approximately 45% of that of the combinatorial approach. The profit ratio shows a difference of less than 0.2% for a higher number of users, and around 1.2% for $N = 50$. This shows that while the chosen order is slightly less accurate, it is significantly faster (2x), especially with increasing numbers of users.

To validate the heuristic approach, we compare the performance of our algorithm with the BARON solver [15]. Problem (19a)–(19e) has a non-concave objective function and non-convex constraints, requiring a global MINLP solver. BARON achieves nearly global optimal solutions using interval analysis and range reduction techniques within a branch-and-bound framework. We evaluate both algorithms based on profit and execution time. The maximum runtime for BARON is set to two hours. We consider $N = 10, 25, 50, 100, 200$ and $|\mathcal{D}| = 3, 4, 5$.

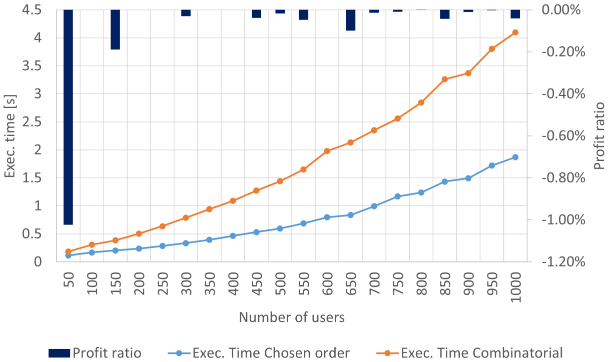


Figure 10: Profit ratio and Average execution time of the chosen order and the combinatorial, 5 deployments scenario.

First, Tables 8 and 9 in the Appendix show the profit ratio with the combinatorial approach and the execution time of the chosen order compared to BARON. The results indicate that BARON fails to converge to the optimal solution for more than 100 users. Consequently, the average profit obtained using BARON is lower than that of the combinatorial approach. The profit ratio is based on the best feasible solution found within the two-hour limit, which may not be the globally optimal solution. Negative profit ratio values indicate that both BARON and the heuristic algorithm with the chosen order perform worse than the combinatorial approach. For a small number of users, BARON achieves slightly better profits, up to 4.31% (with $N = 25$ and $|D| = 4$), but at a significantly longer computational time compared to the chosen heuristic approach (up to 50,000 times longer). In Table 10, we investigate whether BARON can enhance the best profit achieved by the combinatorial approach by initializing the solver with that solution. This initialization bypasses the need for an initial search for a feasible solution and prunes nodes with upper bounds lower than the profit of the initial solution. The results show that while BARON often converges when initialized, it improves the profit over the combinatorial algorithm in only a few instances (20% of cases at most). However, the profit ratio achieved by BARON does not exceed 1.66% in scenarios with fewer users. For a larger number of users, the profit ratio does not exceed 0.29%. This means a profit difference between BARON and the heuristic with the chosen order of less than 0.30%, despite a huge overhead in terms of execution time.

5.3 Users and application provider behaviour

In this section, we analyze the behavior of the edge platform by varying individual user workloads to observe how resources are allocated across the system. We begin by considering a scenario with $|\mathcal{D}| = 3$ and two local deployments, assuming all users can run all deployments within their energy constraints. No users drop out of the system, and the platform profit function exhibits only changing deployment points. For higher prices r, all users opt for the local deployment 1 or 2, whose costs are independent of r and always lower than the economic value U_i assigned to the application. In our analysis, we vary the number of users N within the range of [50, 1000], with a step size of 50 users, and the incoming workload λ within the range of [0.5, 4.0] requests/s, with a step size of 0.5 requests/s. Specifically, Figures 11a and 11b illustrate the number of edge servers n_e and cloud VMs n_c used for different values of N and λ, while Figure 11c analyzes the fraction of users choosing the partitioned deployment at the optimal price r. From the figures, we observe that for $\lambda < 1$ requests/s, all users select the first two deployments. This is because at this incoming workload λ, the power consumption required to run the application entirely on the local device is sufficiently low, making it consistently the most economical option. As λ increases to ≥ 1.5 requests/s, we notice that the edge servers are always saturated. Conversely, the number of cloud VMs increases with both N and λ.

As a second analysis, we examine three specific scenarios. In Figure 12a, we observe that in the *AllLocal* scenario, where all users select local deployments, the average platform profit is a linear increasing function. Figure 12b illustrates that an *OnlyEdge* solution would be infeasible in a scenario where all users select the third deployment (*AllDep3*), due to the required number of edge servers exceeding the available ones. Finally, in Figure 12c, we observe that in an *AllDep3* scenario, the *OnlyCloud* solution is less profitable than the general solution, mainly because the cost of edge infrastructure is lower than that of cloud VMs.

5.4 Gain in the Stackelberg game resolution

To illustrate the rationale for modeling the problem as a Stackelberg game (19a)–(19e), rather than imposing a fixed price r, we compute the profit ratio between the optimal solution and three specific points: r_{min}, $r_{mean} = 2.5r_{min}$, and $r_{max} = 4r_{min}$. This analysis considers $N = 10, 25, 50, 100, 200$ and $|\mathcal{D}| = 3, 4, 5$. For consistency, we set $T = 1320$ s, which is the maximum execution time of the application for users. Figure 13 illustrates that solving the Stackelberg game generally leads to higher profitability compared to fixing r at a specific value. Moreover, due to parameter variability, there is no guaranteed superiority of one fixed point over another. Experimental results show an average profit ratio of 40% higher than r_{min}, 16% higher than r_{mean}, and 66% higher than r_{max}.

5.5 Performance evaluation of the follow the line algorithm under partial knowledge of users' parameters

As discussed in Section 4.5, in a more realistic scenario, the platform provider lacks knowledge of sensitive user parameters such as β_i, ζ_i, and U_i. This section presents the results of the "Follow the Line" Algorithm 5. Specifically, Section 5.5.1 evaluates the performance, while Section 5.5.2 analyzes its sensitivity to parameter settings.

5.5.1 *Performance evaluation*

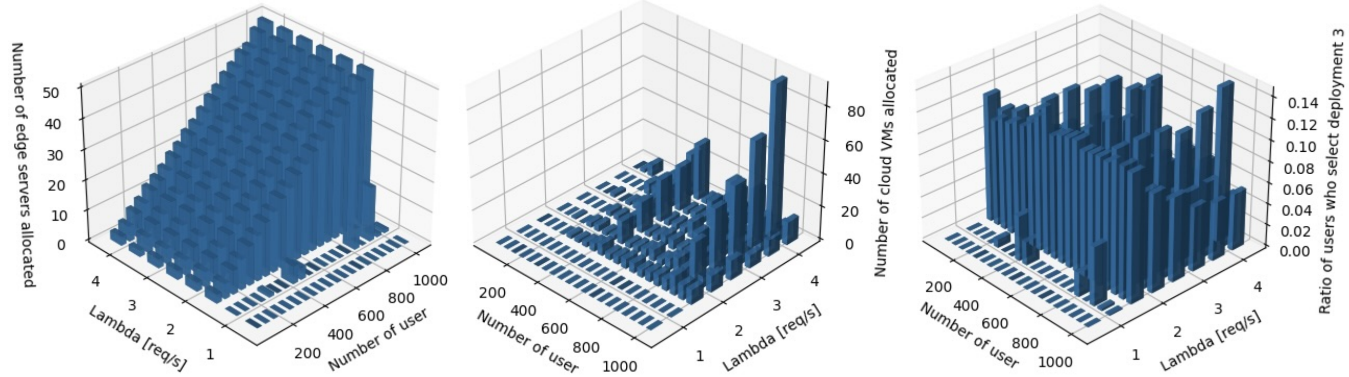
In the early development phase, we fine-tuned the parameters of Algorithm 5 to achieve the best settings in terms of both profit estimation and execution time. Table 7 in the Appendix reports the values we identified. Next, we attempt to reduce communication with users and consequently the number of total points analyzed, aiming to speed up the algorithm while observing whether the estimate becomes poor below a certain threshold. For this purpose, only the fractions of initial points and total points are varied. In the following analysis, we consider $N = 100, 250, 500, 750, 1000$ and individually $|\mathcal{D}| = 3, 4, 5$. Specifically, Table 3 illustrates the profit ratio of the follow the line algorithm (with chosen order) compared to the combinatorial approach with full knowledge, as well as the computation time, varying the number of points analyzed. Negative profit ratio values imply that the follow the line approach achieves a worse result on average than the combinatorial one. Results indicate that with fewer users, a larger fraction of points is needed to achieve a low profit ratio, i.e., a more precise estimate of the best profit, such as for $N = 100$. Conversely, for a large N, it is sufficient to pick $N/20$ points to maintain a profit ratio lower than 1%. Note that for large N, analyzing a large number of points does not guarantee a better profit estimate due to local maxima in the platform profit function. Regarding execution time, it decreases with the number of analyzed points and, in the worst case, is just over 4 s (see Table 3).

5.5.2 *Sensitivity analysis on parameters*

In this section, we conduct a sensitivity analysis on certain parameters to examine how the follow the line algorithm responds to variations in these settings. As in previous comparisons, the reference solution is the one obtained by the full knowledge combinatorial algorithm. The analysis is carried out by fixing all parameters as in Table 7, except for the considered parameter, which is varied in a range of ±50%. Results, shown in Figure 14, are obtained by averaging over the number of users, with $N = 100, 250, 500, 750, 1000$, and over deployments, $|\mathcal{D}| = 3, 4, 5$. Specifically, we vary the *Cutting Fraction* (Figure 14a), *Fraction of Initial Points* (Figure 14b), and *Epsilon Scale* (Figure 14c). In all figures, the red bar represents the value obtained with the settings shown in Table 7. In general, we observe that the algorithm is not strongly dependent on the parameter settings. Additionally, for some specific parameter values, the algorithm appears to achieve better profit ratios than with the fixed settings. However, these variations are not significant and may depend on the particular instances considered. The results demonstrate that further fine-tuning is unnecessary, indicating the robustness of the algorithm, as parameters can be chosen within ranges.

6 RELATED WORK

MEC systems have gained considerable interest in the literature. In particular, [11] focused on a multi-Base Station (BS) and multi-service edge-cloud-assisted IoT environment, where both the BSs (with edge servers deployed) and the cloud assist IoT devices in processing multi-type DL tasks via task offloading. The authors formulated a joint optimization problem aimed at minimizing the total delay of all DL tasks and concentrated on the partition point selection for DNN models. In [29], a user job model and a multi-resource allocation (MRA) framework were introduced for efficient processing of user requests in a wireless communication, edge, and cloud computing environment. The framework utilizes a deep-deterministic policy gradient-based approach with convolutional residual learning units and attention layers to address the MRA problem. The proposed method achieved better conver-

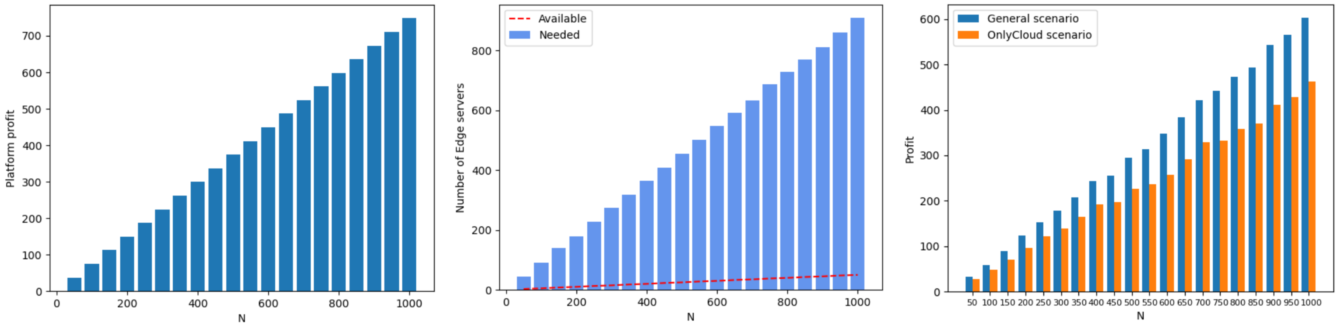


(a) Number of edge servers used.

(b) Number of cloud VMs used.

(c) Ratio of users who select the third deployment.

Figure 11: Users and application behaviour at the optimal price r as N and λ vary, with 3 deployments and maximum energy for all the users.



(a) Platform profit, *AllLocal*.

(b) Edge servers, *AllDep3* OnlyEdge.

(c) Cloud VMs, *AllDep3* OnlyCloud.

Figure 12: Particular scenarios, whose results validate our approach.

N	**Profit ratio (%)**											
	$\lvert\mathcal{D}\rvert = 3$				$\lvert\mathcal{D}\rvert = 4$				$\lvert\mathcal{D}\rvert = 5$			
	N/5	N/10	N/20	N/40	N/5	N/10	N/20	N/40	N/5	N/10	N/20	N/40
100	-1.88	-3.91	-7.32	-62.07	-2.73	-4.22	-12.20	-11.35	-1.96	-1.65	-10.42	-12.40
250	-0.38	-0.87	-1.15	-4.36	-1.04	-0.93	-2.50	-11.11	-0.78	-0.90	-1.01	-8.92
500	-0.70	-0.60	-0.42	-1.57	-1.14	-0.72	-0.39	-0.22	-0.32	-0.25	-0.66	-1.75
750	-0.17	-0.19	-0.82	-0.26	-0.23	-0.37	-0.29	-1.62	-0.81	-0.12	-0.33	-0.59
1000	-0.13	-0.09	-0.13	-0.22	-0.11	-0.13	-0.48	-0.27	-0.13	-0.18	-0.42	-0.21
Execution time (s)												
100	0.10	0.06	0.04	0.06	0.11	0.06	0.04	0.05	0.11	0.06	0.04	0.07
250	0.33	0.18	0.11	0.08	0.36	0.20	0.13	0.09	0.41	0.22	0.14	0.10
500	0.87	0.50	0.29	0.20	1.02	0.62	0.33	0.22	1.19	0.69	0.39	0.24
750	1.70	0.97	0.60	0.33	2.01	1.12	0.69	0.39	2.36	1.31	0.80	0.46
1000	3.01	1.62	0.93	0.53	3.56	1.98	1.16	0.64	4.15	2.20	1.31	0.69

Table 3: Execution time and profit ratio of the Follow the line Algorithm (partial knowledge).

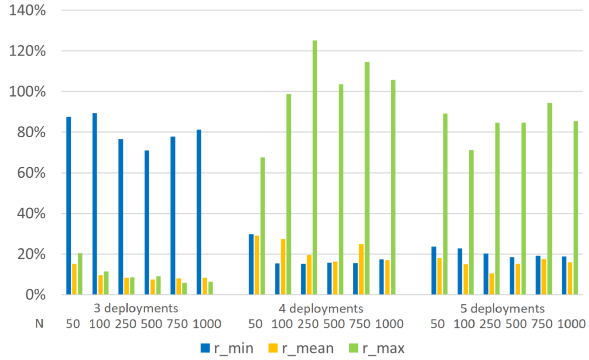
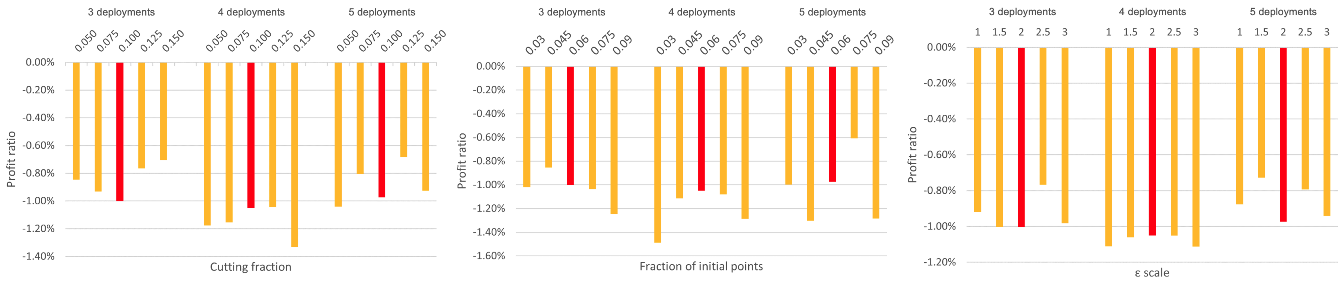


Figure 13: Profit ratios between the optimal solution and those with a fixed cost r.

gence, lower rejection rates, and improved Quality of Experience compared to existing approaches. The authors in [30] investigated the dynamic offloading of finite block length packets in an edge-cloud collaborative MEC system. The system comprises multiple mobile IoT devices (MIDs) with energy harvesting capabilities, multiple edge servers, and a single cloud server, all operating within a dynamic environment. Their objective was to minimize the average long-term service cost, which combines the weighted sum of MID energy consumption and service delay. The optimization problem takes into account various constraints, including available resources, energy causality, allowable service delay, and maximum decoding error probability. [31] proposed an algorithm for optimizing offloading decisions in heterogeneous edge/cloud computing environments. The approach combines meta-reinforcement learning (RL) and deep RL to improve offloading performance in terms of response time and energy consumption.

Many other works have focused on offloading and dynamic resource allocation methods. [32] introduced a three-tier wireless powered mobile edge computing (WPMEC) system, comprising cloud, MEC servers, and IoT devices.



(a) Profit ratio vs *Cutting fraction*. (b) Profit ratio vs *Fraction of initial points*. (c) Profit ratio vs $\epsilon\ scale$.

Figure 14: Sensitivity analysis of the Follow the Line Algorithm. The red bars are the values obtained with the setting shown in Table 7

The authors formulated a combinatorial optimization problem to minimize wireless energy transmission and evaluated the performance of their proposed mechanism. Their approach achieved a significant reduction in the energy required for IoT device operation compared to various offloading policies. The authors in [7] aimed to develop an adaptive and priority-based resource allocation mechanism for efficient resource management in time-constrained edge computing applications. Their objectives included designing a technique for allocating resources based on application time constraints, developing an adaptive resource management system considering priorities and urgency, and optimizing resource utilization to reduce wastage. [33] presented a solution to the challenge of reducing network delay and minimizing the number of edge servers in MEC design, aiming for cost efficiency. This design encompasses the placement of edge servers and the allocation of base stations, creating a combined combinatorial optimization problem (COP). While RL has shown promise in addressing COPs, its application to real-world problems with large state and action spaces requires further investigation. To address this, the authors proposed a novel RL framework that effectively represents and models the state space, action space, and penalty function within the underlying Markov decision process to solve the problem.

Another part of the literature has focused on the partition placement of DNNs. In particular, [34] assumed a single partitioning point in their scenario, with the DNN model partitioned into two parts for executing DNN inference on the mobile web and the edge, respectively. They formulated the partition placement problem as a multi-objective optimization with latency and mobile energy as the two objectives, guaranteeing an upper bound constraint for each objective. They used the weighted sum of latency and mobile energy to solve the problem with linear complexity and find the optimal partition point. The authors in [35] proposed a distributed DNN computing framework for IoT systems, focusing on optimizing latency. They introduced a scheme for collaborative inference on heterogeneous devices using containerization. The researchers developed a problem formulation considering factors like end-to-end delay, service probability, and energy consumption, and employed Deep RL for resource allocation. Finally, the authors in [36] modeled the application components as a Directed Acyclic Graph (DAG) and considered heterogeneous resources in edge and cloud environments. They formulated the component placement and resource selection problem as a Mixed Integer Non Linear Program (MINLP) and proposed a randomized greedy algorithm to solve it. Subsequently, in [37], they extended their work to include DNN components partitioning as an optimization knob.

To the best of our knowledge, this is the first paper addressing the resource allocation problem of AI applications running in next-generation smart eyewear systems from the perspective of platform providers, while ensuring compliance with response time constraints. In a previous work [38], we considered a mobile crowdsensing system (MCS) based on smart eyewear, where users sell their AI sensing task data to the MCS provider. In that work, we assumed only two deployments (a full DNN and a single two-partition DNN) for the AI sensing task. In this paper, however, end-users pay for accessing the application, leading to a different profit function, and the problem involves multiple deployments, including three DNN partitions. Additionally, we consider the more realistic scenario where the application provider has partial knowledge of the users' sensitive parameters.

7 CONCLUSION

In this work, we modeled a MEC system to support AI applications based on a single DNN. We formulated the interaction among the users and the edge platform as a Stackelberg game with one leader and multiple followers, resulting in a non-convex MINLP problem. We then developed a heuristic algorithm that solves the problem very quickly, in both scenarios where the platform knows or does not know the users' sensitive parameters. The results showed that our approach outperforms baseline methods, being orders of magnitude faster than the BARON SOTA commercial solver. Moreover, in the absence of knowledge of the sensitive parameters, the loss in platform profit is around 1%. We have also demonstrated the advantage for the application provider in solving the Stackelberg game rather than setting a constant price.

Future work will focus on testing our solution in a prototype environment. Additionally, the model will be extended to include re-configuration overhead on the user side.

ACKNOWLEDGMENTS

This work has been sponsored by the EssilorLuxottica Smart Eyewear Lab.

REFERENCES

[1] D. Schubmehl, "Worldwide Artificial Intelligence Software Platforms Forecast, 2019–2023," https://www.idc.com/getdoc.jsp?containerId=US49571222, 2019.

[2] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Collaborative vehicular edge computing networks: Architecture design and research challenges," *IEEE Access*, vol. 7, pp. 178 942–178 952, 2019.

[3] P. Fraga-Lamas, T. M. Fernandez-Carames, O. Blanco-Novoa, and M. A. Vilar-Montesinos, "A review on industrial augmented reality systems for the industry 4.0 shipyard," *Ieee Access*, vol. 6, pp. 13 358–13 375, 2018.

[4] A. B. De Souza, P. A. Rego, T. Carneiro, J. D. C. Rodrigues, P. P. Reboucas Filho, J. N. De Souza, V. Chamola, V. H. C. De Albuquerque, and B. Sikdar, "Computation offloading for vehicular environments: A survey," *IEEE Access*, vol. 8, pp. 214–243, 2020.

[5] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on sarsa," *IEEE Access*, vol. 8, pp. 54 074–54 084, 2020.

[6] S. Al-Sarawi, M. Anbar, R. Abdullah, and A. B. Al Hawari, "Internet of things market analysis forecasts, 2020–2030," in *WorldS4*. IEEE, 2020, pp. 449–453.

[7] Z. Sharif, L. T. Jung, I. Razzak, and M. Alazab, "Adaptive and priority-based resource allocation for efficient resources utilization in mobile edge computing," *IEEE Internet of Things Journal*, 2021.

[8] Y. Huang, X. Qiao, P. Ren, L. Liu, C. Pu, S. Dustdar, and J. Chen, "A lightweight collaborative deep neural network for the mobile web in edge cloud," *IEEE Transactions on Mobile Computing*, 2020.

[9] J. S. Devagiri, S. Paheding, Q. Niyaz, X. Yang, and S. Smith, "Augmented reality and artificial intelligence in industry: Trends, tools, and future challenges," *Expert Systems with Applications*, vol. 207, p. 118002, 2022.

[10] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge," in *ACM ASPLOS '17*, 2017.

[11] W. Fan, L. Gao, Y. Su, F. Wu, and Y. Liu, "Joint dnn partition and resource allocation for task offloading in edge-cloud-assisted iot environments," *IEEE Internet of Things Journal*, 2023.

[12] F. Mireshghallah, M. Taram, P. Vepakomma, A. Singh, R. Raskar, and H. Esmaeilzadeh, "Privacy in deep learning: A survey," *arXiv preprint arXiv:2004.12254*, 2020.

[13] K. Rajasekar, R. Loh, K. W. Fok, and V. L. Thing, "Privacy preserving layer partitioning for deep neural network models," *arXiv preprint arXiv:2404.07437*, 2024.

[14] Q. Liu, Q. Huang, X. Chen, S. Wang, W. Wang, S. Han, and P. P. Lee, "Pp-stream: Toward high-performance privacy-preserving neural network inference via distributed stream processing," in *Proceedings of the 40th IEEE International Conference on Data Engineering (ICDE 2024)*, 2024.

[15] "MINLP: BARON," https://minlp.com/baron-solver.

[16] U. Tadakamalla and D. A. Menasce, "Autonomic resource management for fog computing," *IEEE TCC*, pp. 1–1, 2021.

[17] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik., *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, 1984.

[18] "Amazon ec2," https://aws.amazon.com/ec2/pricing/.

[19] "Azure," https://azure.microsoft.com/en-us/pricing/.

[20] A. W. Kambale, H. Sedghani, F. Filippini, G. Verticale, and D. Ardagna, "Runtime management of artificial intelligence applications for smart eyewears," in *Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing*, 2023, pp. 1–8.

[21] H. Sedghani, D. Ardagna, M. Passacantando, M. Z. Lighvan, and H. S. Aghdasi, "An incentive mechanism based on a Stackelberg game for mobile crowdsensing systems with budget constraint," *Ad Hoc Networks*, vol. 123, p. 102626, 2021.

[22] "Zenodo," https://doi.org/10.5281/zenodo.13682044.

[23] D. Ardagna, B. Panicucci, and M. Passacantando, "Generalized Nash equilibria for the service provisioning problem in cloud systems," *IEEE Transactions on Services Computing*, vol. 6, pp. 429–442, 2013.

[24] M. Passacantando, D. Ardagna, and A. Savi, "Service provisioning problem in cloud and multi-cloud systems," *INFORMS Journal on Computing*, vol. 28, pp. 265–277, 2016.

[25] "Open Signal, Italy, May 2022, 5G Experience Report," https://www.opensignal.com/reports/2022/05/italy/mobile-network-experience-5g.

[26] "Average monthly electricity wholesale price in italy from january 2019 to january 2023," https://www.statista.com/statistics/1267548/italy-monthly-wholesale-electricity-price/.

[27] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[28] C. Chen, X. Chen, D. Das, D. Akhmetov, and C. Cordeiro, "Overview and performance evaluation of wi-fi 7," *IEEE Communications Standards Magazine*, vol. 6, no. 2, pp. 12–18, 2022.

[29] A. Qadeer and M. J. Lee, "Deep-deterministic policy gradient based multi-resource allocation in edge-cloud system: A distributed approach," *IEEE Access*, vol. 11, pp. 20 381–20 398, 2023.

[30] H. Hu, D. Wu, F. Zhou, X. Zhu, R. Q. Hu, and H. Zhu, "Intelligent resource allocation for edge-cloud collaborative networks: A hybrid ddpg-d3qn approach," *IEEE Transactions on Vehicular Technology*, 2023.

[31] Z. Zhang, N. Wang, H. Wu, C. Tang, and R. Li, "Mr-dro: A fast and efficient task offloading algorithm in heterogeneous edge/cloud computing environments," *IEEE Internet of Things Journal*, 2021.

[32] M. Bolourian and H. Shah-Mansouri, "Energy-efficient task offloading for three-tier wireless powered mobile edge computing," *IEEE Internet of Things Journal*, 2023.

[33] A. Mazloomi, H. Sami, J. Bentahar, H. Otrok, and A. Mourad, "Reinforcement learning framework for server placement and workload allocation in multiaccess edge computing," *IEEE Internet of Things Journal*, vol. 10, no. 2, pp. 1376–1390, 2022.

[34] Y. Huang, X. Qiao, S. Dustdar, and Y. Li, "AoDNN: An Auto-Offloading Approach to Optimize Deep Inference for Fostering Mobile Web," in *IEEE INFOCOM*, 2022.

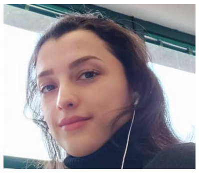
[35] T. Kim, H. Park, Y. Jin, S.-s. Lee, and S. Lee, "Partition placement and resource allocation for multiple dnn-based applications in heterogeneous iot environments," *IEEE Internet of Things Journal*, 2023.

[36] H. Sedghani, F. Filippini, and D. Ardagna, "A Randomized Greedy Method for AI Applications Component Placement and Resource Selection in Computing Continua," in *IEEE JCC*, 2021.

[37] ——, "A Random Greedy based Design Time Tool for AI Applications Component Placement and Resource Selection in Computing Continua," in *IEEE EDGE*, 2021.

[38] H. Sedghani, M. Z. Lighvan, H. S. Aghdasi, M. Passacantando, G. Verticale, and D. Ardagna, "A Stackelberg Game Approach for Managing AI Sensing Tasks in Mobile Crowdsensing," *IEEE Access*, vol. 10, pp. 91 524–91 544, 2022.

Roberto Sala received his M.S degree in mathematical engineering from the Politecnico di Milano, Italy, in 2023. Currently, he is a Ph.D. researcher in Computer Science and Engineering at Politecnico di Milano. His research interests focus on modelling and performance management of HPC-cloud-edge systems and Bayesian optimisation methods for auto-tuning these infrastructures.



Hamta Sedghani received her B.Sc. degree from Iran University of Science and Technology, Tehran, Iran and her M.S and Ph.D. degree from University of Tabriz, Tabriz, Iran in Computer Engineering. Currently, she is post doc researcher at Politecnico di Milano. Her current interests include game theory and optimization for resource management in computing continuum.

Mauro Passacantando received the M.S. and Ph.D. degrees in Mathematics from University of Pisa, Italy, in 2000 and 2005, respectively. He is currently Associate Professor of Operations Research (qualified for Full Professorship) at the Department of Business and Law at University of Milan-Bicocca. His research is mainly devoted to variational inequalities and equilibrium problems. In the last years, his work focused on game theoretic models applied to service provisioning problems in cloud and multicloud systems and infrastructure and spectrum sharing in mobile networks.

Giacomo Verticale received the Ph.D. degree in telecommunications engineering from the Politecnico di Milano, Italy, in 2003. He is currently an Associate Professor at the Politecnico di Milano. He was involved in several research projects on fixed and wireless broadband access technologies and promoting the smart grid. His current interests focus on the security issues of the smart grid, on network function virtualization, and on edge computing in 5G.

Danilo Ardagna is Associate Professor at the Dipartimento di Elettronica Informazione and Bioingegneria at Politecnico di Milano. He received a Ph.D. degree in computer engineering in 2004 from Politecnico di Milano from which he also graduated in December 2000. His work focuses on the design, prototype and evaluation of optimization algorithms for resource management of cloud computing and big data systems.

APPENDIX

For the sake of space, we provide proofs of the theorems, details of the functions embedded in the algorithms, and some tables of the experimental results in the following subsections.

Proof of the theorems

In this section, we provide the proofs of the theorems stated in Section 4. We recall that Theorems 4.1 and 4.3 establish the convexity of the response time constraints on the platform side for the single and multiple deployment offloading scenarios, respectively. Theorem 4.2 provides an exact solution to the platform problem for the single deployment offloading scenario, without considering the delay term due to data transfer between edge and cloud. Furthermore, we show that there is no closed-form solution for the problem when including this delay term. Theorems 4.4 and 4.5 estimate the optimal number of edge servers and cloud VMs required to meet user demand and maintain an average response time in the Only Edge and Only Cloud scenarios, respectively. Finally, Theorem 4.6 proves that the maximum of the platform profit function lies at a point of discontinuity or in a left neighborhood of it.

Proof of Theorems 4.1 and 4.3

Proof. We denote the response time of edge resources for deployment k as follows:

$$f(n_e^{(k)}, \lambda_e^{(k)}) = \frac{\lambda_e^{(k)}}{\lambda^{(k)}} \cdot \frac{D_e^{(k)} n_e^{(k)}}{n_e^{(k)} - D_e^{(k)} \lambda_e^{(k)}}. \tag{30}$$

The first derivative of f respect to $n_e^{(k)}$ is

$$\frac{\partial f(n_e^{(k)}, \lambda_e^{(k)})}{\partial n_e^{(k)}} = -\frac{1}{\lambda^{(k)}} \frac{(\lambda_e^{(k)})^2 (D_e^{(k)})^2}{\left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^2}, \tag{31}$$

the first derivative of f respect to $\lambda_e^{(k)}$ is

$$\frac{\partial f(n_e^{(k)}, \lambda_e^{(k)})}{\partial \lambda_e^{(k)}} = \frac{1}{\lambda^{(k)}} \cdot \frac{D_e^{(k)} (n_e^{(k)})^2}{\left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^2}. \tag{32}$$

The second derivative of f respect to $n_e^{(k)}$ is

$$\frac{\partial^2 f(n_e^{(k)}, \lambda_e^{(k)})}{\partial {n_e^{(k)}}^2} = \frac{1}{\lambda^{(k)}} \frac{2(\lambda_e^{(k)})^2 (D_e^{(k)})^2}{\left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^3}, \tag{33}$$

the second derivative of f respect to $\lambda_e^{(k)}$ is

$$\frac{\partial^2 f(n_e^{(k)}, \lambda_e^{(k)})}{\partial {\lambda_e^{(k)}}^2} = \frac{1}{\lambda^{(k)}} \cdot \frac{2(D_e^{(k)})^2 (n_e^{(k)})^2}{\left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^3}, \tag{34}$$

and the mixed derivative is

$$\begin{aligned} \frac{\partial^2 f(n_e^{(k)}, \lambda_e^{(k)})}{\partial \lambda_e^{(k)} \partial n_e^{(k)}} &= \frac{\partial^2 f(n_e^{(k)}, \lambda_e^{(k)})}{\partial n_e^{(k)} \partial \lambda_e^{(k)}} \\ &= -\frac{1}{\lambda^{(k)}} \left(\frac{2\lambda_e^{(k)} (D_e^{(k)})^2 n_e^{(k)}}{\left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^3} \right). \end{aligned} \tag{35}$$

Hence, the Hessian matrix of f is

$$\begin{aligned} &\nabla^2 f(n_e^{(k)}, \lambda_e^{(k)}) = \\ &\frac{2(D_e^{(k)})^2}{\lambda^{(k)} \left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^3} \begin{bmatrix} (\lambda_e^{(k)})^2 & -\lambda_e^{(k)} n_e^{(k)} \\ -\lambda_e^{(k)} n_e^{(k)} & (n_e^{(k)})^2 \end{bmatrix}. \end{aligned} \tag{36}$$

Notice that the determinant of the Hessian matrix (i.e., the product of eigenvalues) is equal to zero and the trace of the Hessian matrix (i.e., the sum of the eigenvalues) is equal to

$$\frac{2(D_e^{(k)})^2 \left[(\lambda_e^{(k)})^2 + (n_e^{(k)})^2\right]}{\left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^3} > 0. \tag{37}$$

Hence, the Hessian matrix is positive semi-definite and f is convex. Considering the response time of cloud resources for deployment k, i.e., the function

$$g(n_c^{(k)}, \lambda_c^{(k)}) = \frac{\lambda_c^{(k)}}{\lambda^{(k)}} \cdot \frac{D_c^{(k)} n_c^{(k)}}{n_c^{(k)} - D_c^{(k)} \lambda_c^{(k)}}, \tag{38}$$

the result is analogous. We denote, now, the data transfer time for deployments k as follows:

$$h(n_c^{(k)}, \lambda_c^{(k)}) = \frac{\lambda_c^{(k)}}{\lambda^{(k)}} \cdot \frac{\delta_{II}^{(k)}}{\bar{B}}. \tag{39}$$

The Hessian of h is the null matrix, so its eigenvalues are zeros. Since the response time constraints (25) are the sum of multiple convex functions, they are convex. □

Proof of Theorem 4.2

Proof. In the following, since the only non-local deployment is $\bar{k}$, the index $(\bar{k})$ is omitted. Therefore, $n_e = n_e^{(\bar{k})}$, $n_c = n_c^{(\bar{k})}$, $D_e = D_e^{(\bar{k})}$, $D_c = D_c^{(\bar{k})}$, $\lambda_e = \lambda_e^{(\bar{k})}$, $\lambda_c = \lambda_c^{(\bar{k})}$ and $\lambda = \lambda^{(\bar{k})}$. The edge platform problem, where no cloud resource is used, can be formulated as follows:

$$\begin{aligned} &\min c_e n_e \\ \text{subject to: } & n_e \le N_e \\ & \frac{D_e n_e}{n_e - D_e \lambda} \le \bar{\bar{R}} \\ & n_e - D_e \lambda > 0 \end{aligned}$$

that is equivalent to

$$\begin{aligned} &\min c_e n_e \\ \text{s.t. } & n_e \le N_e \\ & n_e \ge \frac{\bar{\bar{R}} D_e \lambda}{\bar{\bar{R}} - D_e} \end{aligned}$$

whose optimal solution is

$$n_e^* = \frac{\bar{\bar{R}} D_e \lambda}{\bar{\bar{R}} - D_e},$$

provided that the feasible region of the latter problem is nonempty, i.e.,

$$\lambda \le \frac{N_e(\bar{\bar{R}} - D_e)}{\bar{\bar{R}} D_e},$$

that is the total load is small enough.

If the total load does not satisfy the above condition, then edge resources are saturated and also cloud resources have to be used. Thus, the edge platform problem is:

$$\begin{aligned}
&\min_{(\lambda_e, n_c)} \; c_c n_c \\
&\text{s.t. } \frac{D_e N_e \lambda_e}{N_e - D_e \lambda_e} + \frac{n_c D_c(\lambda - \lambda_e)}{n_c - D_c(\lambda - \lambda_e)} \le \bar{\bar{R}}\lambda \\
&\quad 0 < \lambda_e < \lambda \\
&\quad N_e - D_e \lambda_e > 0 \\
&\quad n_c - D_c(\lambda - \lambda_e) > 0.
\end{aligned}$$

The latter problem is convex and standard constraint qualifications hold (e.g., Slater condition is satisfied), hence it is equivalent to the corresponding KKT system:

$$\begin{aligned}
&\mu_1 \left[\frac{D_e N_e^2}{(N_e - D_e\lambda_e)^2} - \frac{D_c n_c^2}{(n_c - D_c(\lambda - \lambda_e))^2} \right] \\
&\quad - \mu_2 + \mu_3 + D_e \mu_4 - D_c \mu_5 = 0 \\
&c_c - \mu_1 \frac{D_c^2(\lambda - \lambda_e)^2}{(n_c - D_c(\lambda - \lambda_e))^2} - \mu_5 = 0 \\
&\frac{D_e N_e \lambda_e}{N_e - D_e \lambda_e} + \frac{n_c D_c(\lambda - \lambda_e)}{n_c - D_c(\lambda - \lambda_e)} \le \bar{\bar{R}}\lambda \\
&\mu_1 \ge 0, \quad \mu_1 \left[\frac{D_e N_e \lambda_e}{N_e - D_e \lambda_e} + \frac{n_c D_c(\lambda - \lambda_e)}{n_c - D_c(\lambda - \lambda_e)} - \bar{\bar{R}}\lambda \right] = 0 \\
&\lambda_e > 0, \quad \mu_2 \ge 0, \quad \mu_2 \lambda_e = 0 \\
&\lambda_e < \lambda, \quad \mu_3 \ge 0, \quad \mu_3(\lambda - \lambda_e) = 0 \\
&N_e - D_e\lambda_e > 0, \quad \mu_4 \ge 0, \quad \mu_4(N_e - D_e\lambda_e) = 0 \\
&n_c - D_c(\lambda - \lambda_e) > 0, \quad \mu_5 \ge 0, \quad \mu_5[n_c - D_c(\lambda - \lambda_e)] = 0
\end{aligned}$$

Notice that constraints imply $\mu_2 = \mu_3 = \mu_4 = \mu_5 = 0$. Moreover, it follows from second equation that $\mu_1 > 0$, thus first constraint holds as an equality, i.e.,

$$\frac{D_e N_e \lambda_e}{N_e - D_e \lambda_e} + \frac{n_c D_c(\lambda - \lambda_e)}{n_c - D_c(\lambda - \lambda_e)} = \bar{\bar{R}}\lambda,$$

that is equivalent to

$$n_c = \frac{\left[\bar{\bar{R}}\lambda - \frac{D_e N_e \lambda_e}{N_e - D_e\lambda_e}\right] D_c(\lambda - \lambda_e)}{\bar{\bar{R}}\lambda - \frac{D_e N_e \lambda_e}{N_e - D_e\lambda_e} - D_c(\lambda - \lambda_e)}. \tag{40}$$

Since $\mu_1 > 0$, first equation implies that

$$\frac{D_e N_e^2}{(N_e - D_e\lambda_e)^2} = \frac{D_c n_c^2}{(n_c - D_c(\lambda - \lambda_e))^2},$$

hence

$$\frac{\sqrt{D_e} N_e}{N_e - D_e \lambda_e} = \frac{\sqrt{D_c} n_c}{n_c - D_c(\lambda - \lambda_e)},$$

that is equivalent to

$$n_c = \frac{\sqrt{D_e} D_c N_e (\lambda - \lambda_e)}{N_e(\sqrt{D_e} - \sqrt{D_c}) + D_e \sqrt{D_c} \lambda_e}. \tag{41}$$

It follows from (40)–(41) that

$$\begin{aligned}
&\sqrt{D_c}(N_e - D_e\lambda_e) \Big[(N_e D_e + \bar{\bar{R}}\lambda D_e - N_e \sqrt{D_c D_e})\lambda_e \\
&\quad - N_e \bar{\bar{R}}\lambda + N_e \lambda \sqrt{D_c D_e} \Big] = 0.
\end{aligned}$$

Since $N_e - D_e\lambda_e > 0$, we get that the optimal edge load is

$$\lambda_e^* = \frac{N_e \lambda (\bar{\bar{R}} - \sqrt{D_c D_e})}{N_e D_e + \bar{\bar{R}}\lambda D_e - N_e \sqrt{D_c D_e}}.$$

Finally, we get from (40) the optimal number of cloud resources:

$$n_c^* = \frac{D_c \lambda [\bar{\bar{R}} D_e \lambda - N_e(\bar{\bar{R}} - D_e)]}{N_e(\sqrt{D_e} - \sqrt{D_c})^2 + D_e \lambda (\bar{\bar{R}} - D_c)}.$$

□

Proof of Theorem 4.2 with the additional delay term $\delta_{II}/\tilde{B}$

Following the very same steps of the Proof of Theorem 4.2, we obtain that the solution for the edge platform problem where no cloud resource is used is the same. If the edge resources are saturated and also cloud resources have to be used, the edge platform problem is:

$$\begin{aligned}
&\min_{(\lambda_e, n_c)} \; c_c n_c \\
&\text{s. t. } \frac{D_e N_e \lambda_e}{N_e - D_e \lambda_e} + (\lambda - \lambda_e)\left[\frac{\delta_{II}}{\tilde{B}} + \frac{D_c n_c}{n_c - D_c(\lambda - \lambda_e)} \right] \le \bar{\bar{R}}\lambda \\
&\quad 0 < \lambda_e < \lambda \\
&\quad N_e - D_e \lambda_e > 0 \\
&\quad n_c - D_c(\lambda - \lambda_e) > 0
\end{aligned}$$

By proceeding in the same way as the Proof of Theorem 4.2, the associated KKT system is the following:

$$\begin{cases}
\dfrac{D_e N_e^2}{(N_e - D_e\lambda_e)^2} - \dfrac{\delta_{II}}{\tilde{B}} - \dfrac{D_c n_c^2}{(n_c - D_c(\lambda - \lambda_e))^2} = 0, \\[2ex]
\dfrac{D_e N_e \lambda_e}{N_e - D_e\lambda_e} + (\lambda - \lambda_e)\left(\dfrac{\delta_{II}}{\tilde{B}} + \dfrac{n_c D_c}{n_c - D_c(\lambda - \lambda_e)} \right) = \bar{\bar{R}}\lambda.
\end{cases}$$

The latter system of equations does not admit solutions in closed form since developing the calculations yields a fourth-degree equation.

Proof of Theorem 4.4

Proof. Let $D = |\mathcal{D}|$, and assume, without loss of generality, $\mathcal{D} \setminus \mathcal{D}' = \{1, 2\}$. The minimization problem is convex and the Slater constraint qualification holds, thus the problem is equivalent to the corresponding KKT system:

$$
\begin{aligned}
&c_e + \mu_1 - \mu_2 \frac{{D_e^{(3)}}^2 \lambda^{(3)}}{(n_e^{(3)} - D_e^{(3)}\lambda^{(3)})^2} - \mu_3 = 0\\
&c_e + \mu_1 - \mu_2 \frac{{D_e^{(4)}}^2 \lambda^{(4)}}{(n_e^{(4)} - D_e^{(4)}\lambda^{(4)})^2} - \mu_4 = 0\\
&\vdots\\
&c_e + \mu_1 - \mu_2 \frac{{D_e^{(D)}}^2 \lambda^{(D)}}{(n_e^{(D)} - D_e^{(D)}\lambda^{(D)})^2} - \mu_D = 0\\
&\mu_1(n_e^{(3)} + n_e^{(4)} + \cdots + n_e^{(D)} - N_e) = 0\\
&\mu_2\Big(\sum_{k\in\mathcal{D}'} \frac{D_e^{(k)} n_e^{(k)}}{n_e^{(k)} - D_e^{(k)}\lambda^{(k)}} - \bar{\bar{R}}\Big) = 0\\
&\mu_3(n_e^{(3)} - D_e^{(3)}\lambda^{(3)}) = 0\\
&\mu_4(n_e^{(4)} - D_e^{(4)}\lambda^{(4)}) = 0\\
&\vdots\\
&\mu_D(n_e^{(D)} - D_e^{(D)}\lambda^{(D)}) = 0
\end{aligned}
$$

Constraints imply $\mu_1 = \mu_3 = \mu_4 = \cdots = \mu_D = 0$ and $\mu_2 > 0$, thus

$$\sum_{k\in\mathcal{D}'} \frac{D_e^{(k)} n_e^{(k)}}{n_e^{(k)} - D_e^{(k)}\lambda^{(k)}} - \bar{\bar{R}} = 0. \tag{42}$$

Moreover, it follows that the first $D-2$ equations become

$$
\begin{aligned}
&c_e - \mu_2 \frac{{D_e^{(3)}}^2 \lambda^{(3)}}{(n_e^{(3)} - D_e^{(3)}\lambda^{(3)})^2} = 0\\
&c_e - \mu_2 \frac{{D_e^{(4)}}^2 \lambda^{(4)}}{(n_e^{(4)} - D_e^{(4)}\lambda^{(4)})^2} = 0\\
&\vdots\\
&c_e - \mu_2 \frac{{D_e^{(D)}}^2 \lambda^{(D)}}{(n_e^{(D)} - D_e^{(D)}\lambda^{(D)})^2} = 0
\end{aligned}
$$

from which, by isolating μ_2/c_e,

$$\mu_2/c_e = \frac{(n_e^{(k)} - D_e^{(k)}\lambda^{(k)})^2}{{D_e^{(k)}}^2 \lambda^{(k)}} \qquad \forall\, k \in \mathcal{D}',$$

which implies the following $D-3$ equations:

$$n_e^{(k)} - D_e^{(k)}\lambda^{(k)} = \frac{D_e^{(k)}}{D_e^{(3)}}\sqrt{\frac{\lambda^{(k)}}{\lambda^{(3)}}} \cdot (n_e^{(3)} - D_e^{(3)}\lambda^{(3)}) \qquad \forall k = 4, \ldots, D. \tag{43}$$

Solving the system of equations (42)-(43) implies

$$n_e^{(k)} = D_e^{(k)}\lambda^{(k)} + D_e^{(k)}\sqrt{\lambda^{(k)}} \cdot \frac{\sum_{m\in\mathcal{D}'} D_e^{(m)}\sqrt{\lambda^{(m)}}}{\bar{\bar{R}} - \sum_{m\in\mathcal{D}'} D_e^{(m)}}$$

holds for any $k \in \mathcal{D}'$. Note that in the OnlyEdge scenario all the resources are in the edge, so

$$\sum_{k\in\mathcal{D}'} n_e^{(k)} < N_e,$$

and

$$
\begin{aligned}
\sum_{k\in\mathcal{D}'} n_e^{(k)} = \sum_{k\in\mathcal{D}'} D_e^{(k)}\lambda^{(k)} &\\
+ \frac{\sum_{m\in\mathcal{D}'} D_e^{(m)}\sqrt{\lambda^{(m)}}}{\bar{\bar{R}} - \sum_{m\in\mathcal{D}'} D_e^{(m)}} &\cdot \sum_{k\in\mathcal{D}'} D_e^{(k)}\sqrt{\lambda^{(k)}}
\end{aligned}
$$

□

Proof of Theorem 4.5

Proof. Let $D = |\mathcal{D}|$, and assume, without loss of generality, $\mathcal{D} \setminus \mathcal{D}' = \{1, 2\}$. The minimization problem is convex and the Slater constraint qualification holds, thus the problem is equivalent to the corresponding KKT system:

$$
\begin{aligned}
&c_c - \mu_1 \frac{{D_c^{(3)}}^2 \lambda^{(3)}}{(n_c^{(3)} - D_c^{(3)}\lambda^{(3)})^2} - \mu_2 = 0\\
&c_c - \mu_1 \frac{{D_c^{(4)}}^2 \lambda^{(4)}}{(n_c^{(4)} - D_c^{(4)}\lambda^{(4)})^2} - \mu_3 = 0\\
&\vdots\\
&c_c - \mu_1 \frac{{D_c^{(D)}}^2 \lambda^{(D)}}{(n_c^{(D)} - D_c^{(D)}\lambda^{(D)})^2} - \mu_{D-1} = 0\\
&\mu_1\left(\sum_{k\in\mathcal{D}'}\left(\frac{\delta_{II}^{(k)}}{\tilde{B}} + \frac{D_c^{(k)} n_c^{(k)}}{n_c^{(k)} - D_c^{(k)}\lambda^{(k)}}\right) - \bar{\bar{R}}\right) = 0\\
&\mu_2(n_c^{(3)} - D_c^{(3)}\lambda^{(3)}) = 0\\
&\mu_3(n_c^{(4)} - D_c^{(4)}\lambda^{(4)}) = 0\\
&\vdots\\
&\mu_{D-1}(n_c^{(D)} - D_c^{(D)}\lambda^{(D)}) = 0
\end{aligned}
$$

Constraints imply $\mu_2 = \mu_3 = \cdots = \mu_{D-1} = 0$ and $\mu_1 > 0$, thus

$$\sum_{k\in\mathcal{D}'}\left(\frac{\delta_{II}^{(k)}}{\tilde{B}} + \frac{D_c^{(k)} n_c^{(k)}}{n_c^{(k)} - D_c^{(k)}\lambda^{(k)}}\right) - \bar{\bar{R}} = 0. \tag{44}$$

Moreover, it follows that the first $D-2$ equations become

$$
\begin{aligned}
&c_c - \mu_1 \frac{{D_c^{(3)}}^2 \lambda^{(3)}}{(n_c^{(3)} - D_c^{(3)}\lambda^{(3)})^2} = 0\\
&c_c - \mu_1 \frac{{D_c^{(4)}}^2 \lambda^{(4)}}{(n_c^{(4)} - D_c^{(4)}\lambda^{(4)})^2} = 0\\
&\vdots\\
&c_c - \mu_1 \frac{{D_c^{(D)}}^2 \lambda^{(D)}}{(n_c^{(D)} - D_c^{(D)}\lambda^{(D)})^2} = 0
\end{aligned}
$$

from which, by isolating μ_1/c_c, we get

$$\mu_1/c_c = \frac{(n_c^{(k)} - D_c^{(k)}\lambda^{(k)})^2}{{D_c^{(k)}}^2 \lambda^{(k)}} \qquad \forall\, k \in \mathcal{D}',$$

which implies the following $D-3$ equations:

$$n_c^{(k)} - D_c^{(k)}\lambda^{(k)} = \frac{D_c^{(k)}}{D_c^{(3)}}\sqrt{\frac{\lambda^{(k)}}{\lambda^{(3)}}} \cdot (n_c^{(3)} - D_c^{(3)}\lambda^{(3)}) \quad \forall\, k = 4, \dots, D. \tag{45}$$

Solving the system of equations (44)-(45) implies

$$n_c^{(k)} = D_c^{(k)}\lambda^{(k)} + D_c^{(k)}\sqrt{\lambda^{(k)}} \cdot \frac{\sum_{m\in\mathcal{D}'} D_c^{(m)}\sqrt{\lambda^{(m)}}}{\bar{\bar{R}} - \left(\sum_{m\in\mathcal{D}'} D_c^{(m)} + \sum_{m\in\mathcal{D}'} \frac{\delta_{II}^{(m)}}{\tilde{B}}\right)}$$

holds for any $k \in \mathcal{D}'$. □

Proof of Theorem 4.6

Proof. Assuming the users' decisions $x_i^{(k)}$ are fixed given the platform profit function (17a), the only part of the platform profit function that changes with increasing r is $r^{(k)}x_i^{(k)}$, which causes an increase of the platform profit. So, in each interval in which the users' decisions are fixed, the maximum point is at the end of the interval. On the other hand, we know that users' behaviours impose the platform's profit and the behaviour of a user will change when they make a decision to participate in the MEC system or not and change the selected deployment from one to another. This behaviour changes happen only immediately before the points of discontinuity and the platform profit can increase or decrease as shown in Figure 5. Therefore, in a specific interval between two points of discontinuity where the users' decisions are fixed, the optimal solution is either obtained by decreasing the price by ϵ, by setting $r = change_{jl} - \epsilon$ or $r = drop_k - \epsilon$, if the platform profit is also increased.

Otherwise, the optimal solution in that interval is either $r = change_{jl}$ or $r = drop_k$. Accordingly, it can be concluded that the optimal platform profit will happen immediately before or exactly at these points of discontinuity. □

Linearization of users' problem

In this section, we present the linearization of the optimality constraint (19b) related to the user-side problem.

Assume, without loss of generality, $\mathcal{D} \setminus \mathcal{D}' = \{1, 2\}$ The cost for user i to run deployment k is computed as

$$C_i^{(k)} = T_i\left(\alpha_i r^{(k)} + (1-\alpha_i)\beta_i\left(p_{I,i}^{(k)} + p_{II,i}^{(k)}\right)\lambda T_i + \zeta_i\delta_{II}^{(k)}\lambda\right), \tag{46}$$

with $\delta_{II}^{(1)} = \delta_{II}^{(2)} = 0$. Since memory and energy constraints are independent of other variables, these conditions are checked before solving the problem and embedded as binary parameter $s_i^{(k)}$ in the optimization problem. Therefore, the binary parameter $s_i^{(k)} = 1$ if user i has both enough energy and memory to run deployment k on both its smart device and phone, while $s_i^{(k)} = 0$ otherwise. The linear constraints equivalent to (19b) are the following:

$$-(1 - t_i^{(k)})M \le U_i - C_i^{(k)} \le t_i^{(k)}M \quad \forall\, k \in \mathcal{D} \tag{47}$$

$$-(1 - z_i^{(jl)})M \le C_i^{(l)} - C_i^{(j)} \le z_i^{(jl)}M \quad \forall\, j, l \in \mathcal{D},\ j \ne l \tag{48}$$

$$z_i^{(jl)} + z_i^{(lj)} = 1 \quad \forall\, j, l \in \mathcal{D},\ j \ne l \tag{49}$$

$$x_i^{(k)} \le s_i^{(k)}t_i^{(k)} \quad \forall\, k \in \mathcal{D} \tag{50}$$

$$x_i^{(k)} \ge s_i^{(k)}t_i^{(k)} + \sum_{l\in\mathcal{D}\setminus\{k\}} s_i^{(l)}(z_i^{(kl)} - 1) \quad \forall\, k \in \mathcal{D} \tag{51}$$

$$\sum_{k\in\mathcal{D}} x_i^{(k)} \le 1 \tag{52}$$

$$x_i^{(k)}, t_i^{(k)}, z_i^{(jl)} \in \{0, 1\} \quad \forall\, k, j, l \in \mathcal{D},\ j \ne l \tag{53}$$

where M is a large enough parameter, variable $t_i^{(k)}$ indicates whether $U_i \ge C_i^{(k)}(r)$ and variable $z_i^{(jl)}$ denotes if $C_i^{(j)}(r) \le C_i^{(l)}(r)$. Note that the dependence of cost on price r makes $t_i^{(k)}$ and $z_i^{(jl)}$ variables.

Support functions of Algorithms

We report here the algorithms related to the support functions of Algorithms 4 and 5. Recall that Algorithm 4 performs the allocation of resources to the users, for a list of elite candidate solutions returned by Algorithm 3 (see Section 4.4). This allocation is done to satisfy the users' time constraints and the constraint on the maximum number of edge servers. The following procedures are used within Algorithm 4. First, the procedure MOVEUSER moves the user j who selected the deployment k, assigned to the edge ($y_j^e = 1$) and with the lowest local execution time $V_k^{(k)}$, from the edge to the cloud. Once the user assignment variables, y_i^e and y_i^c, and deployment loads, $\lambda_e^{(k)}$ and $\lambda_c^{(k)}$, have been updated (lines 2-4), the procedure recomputes the optimal number of edge servers $\bar{n}_e^{(k)}$ needed to serve the load $\lambda_e^{(k)}$ (lines 5-7). Specifically, the calculation of $\bar{n}_e^{(k)}$ is done by restricting the resource allocation problem to only the load $\lambda_e^{(k)}$, relative to deployment k. Consider, therefore, constraint (25). The optimal number of edge servers is obtained by considering as the response time of the platform ($\bar{\bar{R}}$) the smallest residual time, LD, among those of the users who have selected deployment k and are assigned to the edge. Calculations follow:

$$\frac{D_e^{(k)}\bar{n}_e^{(k)}}{\bar{n}_e^{(k)} - D_e^{(k)}\lambda_e^{(k)}} = LD \tag{54}$$

$$\bar{n}_e^{(k)} = D_e^{(k)}\lambda_e^{(k)}\frac{LD}{LD - D_e^{(k)}} \tag{55}$$

$\bar{n}_e^{(k)}$ is then rounded to the next integer. The procedure MOVEDEPLOYMENT moves all users who have chosen a given deployment from edge to cloud, setting the optimal number of edge servers $\bar{n}_e^{(k)}$ for that deployment to zero. COMPUTEEDGESERVERS calculates the optimal number of edge servers $\bar{n}_e^{(k)}$ for a given deployment, based on the load $\lambda_e^{(k)}$ assigned to the edge by Algorithm 3. Specifically, the procedure first calculates the *local execution time* for each user i selecting deployment k and stores it in $V_i^{(k)}$ (line 15). If Algorithm 3 allocated only edge resources to deployment k (line 16), all users are initially assigned to the edge (line 17). If the sum of the local execution time and the edge service time is less than the maximum response time $\bar{R}$ (line 18), the optimal number of edge resources is the one estimated by Algorithm 3 (line 19). Otherwise, the optimal number of edge servers is calculated based on the shortest residual time among the users who selected the given deployment (lines 20-23), similar to the MOVEUSER procedure. If Algorithm 3 assigned cloud resources to deployment k, users are first split between edge and cloud, assigning those with the lowest local execution time to the cloud (lines 27-31). Next, the optimal number of edge servers to satisfy the load $\lambda_e^{(k)}$ is calculated as in the previous case (lines 32-34). Finally, the procedure COMPUTECLOUDVMS calculates the optimal number of cloud VMs to satisfy the load $\lambda_c^{(k)}$ on the cloud. Similar to the edge side, the cloud VMs are calculated based on the smallest remaining time LD among the users who selected deployment k and were allocated on the cloud (lines 39-40). Note that the delay due to data transmission between edge and cloud is also considered in the calculation of LD. The final calculation is similar to (54)-(55) (line 41), substituting the cloud (subscript c) for the edge (subscript e).

Algorithm 5, follow the line, seeks a solution to the problem in the scenario with partial knowledge of user parameters. Specifically, the algorithm tries to improve the solution by moving along the lines of the platform profit function, starting from the most promising r observed, i.e., the one associated with the highest profit (see Section 4.5). Algorithm 6 serves as a support function for Algorithm 5. Given the price r and *ordering* as input, the algorithm asks the control agent for the deployment chosen by each user for that price (lines 3-4), stores the values of $x_i^{(k)}$ and computes the loads $\lambda^{(k)}$ for all deployments $k \in \mathcal{D}'$ (line 5). Subsequently, in analogy with Algorithm 3, the approximate $n_e^{(k)}$, $n_c^{(k)}$, $\lambda_e^{(k)}$, $\lambda_c^{(k)}$ are computed by Algorithm 2 for each order. The platform profit is estimated, and the solutions are stored (lines 7-11). Finally, the best solution in terms of profit is returned (lines 12-13).

Parameters setting

The parameters settings of experiments and corresponding figures and tables are reported in this section. First, Figure 15 shows the results of profiling the YoloV4 neural network, as described in Section 5.1. Each subplot refers to a specific splitting point (the x-axis) between the phone and edge. For example, the first subplot refers to the case in which only local devices are used. The bar at each split point shows the execution time of the smart device and phone if the DNN is split at the corresponding point. As expected, the transfer times from the smart device to the phone are negligible thanks to the WiFi 7 bandwidth. Following this analysis, we selected the splitting points with the lowest total execution time to determine the deployment-dependent system parameters.

Table 4 shows the indices of the selected splitting points relative to Figure 15. Note that the first two deployments have only the smart device-phone splitting point ($\mathcal{D} \setminus \mathcal{D}' = \{1, 2\}$). As for the case with $|\mathcal{D}| = 3$, only the third deployment assigns part of the computation to the edge/cloud, which means that in this scenario we solve the problem in the so-called "single deployment offloading scenario". Many of the parameters of the system are derived from the choice of the splitting points.

Procedures for users assignment

Move users

```
1: procedure MoveUser($V^{(k)}$, $N^{(k)}$, $\bar{R}$, $D_e^{(k)}$, $\lambda_e^{(k)}$, $\lambda_c^{(k)}$, $\lambda$, $y_i^e$, $y_i^c$, $\bar{n}_e^{(k)}$)
2:     $j \leftarrow \operatorname{argmin}_{i \in N^{(k)},\, y_i^e = 1} V_i^{(k)}$
3:     $y_j^e \leftarrow 0, y_j^c \leftarrow 1$
4:     $\lambda_c^{(k)} \leftarrow \lambda_c^{(k)} + \lambda$, $\lambda_e^{(k)} \leftarrow \lambda_e^{(k)} - \lambda$
5:     $j \leftarrow \operatorname{argmax}_{i \in N^{(k)},\, y_i^e = 1} V_i^{(k)}$
6:     $LD \leftarrow \bar{R} - V_j^{(k)}$
7:     $\bar{n}_e^{(k)} \leftarrow \lceil D_e^{(k)} \lambda_e^{(k)} \frac{LD}{LD - D_e^{(k)}} \rceil$
8: end procedure
```

Move deployments

```
9: procedure MoveDeployment($N^{(k)}$, $\lambda_e^{(k)}$, $\lambda_c^{(k)}$, $y_i^e$, $y_i^c$, $\bar{n}_e^{(k)}$)
10:     $y_i^e \leftarrow 0, y_i^c \leftarrow 1, \forall i \in N^{(k)}$
11:     $\lambda_c^{(k)} \leftarrow \lambda_c^{(k)} + \lambda_e^{(k)}$, $\lambda_e^{(k)} \leftarrow 0$
12:     $\bar{n}_e^{(k)} \leftarrow 0$
13: end procedure
```

Compute edge servers

```
14: procedure ComputeEdgeServers($V^{(k)}$, $N^{(k)}$, $\bar{R}$, $D_{j,i}^{(k)}$, $\delta_j^{(k)}$, $\bar{B}_i$, $B_i$, $D_e^{(k)}$, $n_e^{(k)}$, $n_c^{(k)}$, $\lambda_e^{(k)}$, $\lambda_c^{(k)}$, $y_i^e$, $y_i^c$, $\bar{n}_e^{(k)}$, $\bar{n}_c^{(k)}$)
15:     $V_i^{(k)} = D_{I,i}^{(k)} + \frac{\delta_I^{(k)}}{\bar{B}_i} + D_{II,i}^{(k)} + \frac{\delta_{II}^{(k)}}{B_i}, \forall i \in N^{(k)}$
16:     if $n_c^{(k)} = 0$ then
17:         $y_i^e \leftarrow 1, \forall i \in N^{(k)}$
18:         if $V_i^{(k)} + \frac{n_e^{(k)} D_e^{(k)}}{n_e^{(k)} - D_e^{(k)} \lambda_e^{(k)}} \leq \bar{R}\ \forall i \in N^{(k)}$ then
19:             $\bar{n}_e^{(k)} \leftarrow n_e^{(k)}$
20:         else
21:             $j \leftarrow \operatorname{argmax}_{i \in N^{(k)}} V_i^{(k)}$
22:             $LD \leftarrow \bar{R} - V_j^{(k)}$
23:             $\bar{n}_e^{(k)} \leftarrow \lceil D_e^{(k)} \lambda_e^{(k)} \frac{LD}{LD - D_e^{(k)}} \rceil$
24:         end if
25:     end if
26:     if $n_c^{(k)} > 0$ then
27:         sort the users by $V_i^{(k)}$ increasingly
28:         $\lambda^{(k)} \leftarrow \lambda_e^{(k)} + \lambda_c^{(k)}$
29:         $j \leftarrow \frac{\lambda_c^{(k)}}{\lambda} + 1$
30:         $y_i^c \leftarrow 1, \forall i \in N^{(k)} : i < j$
31:         $y_i^e \leftarrow 1, \forall i \in N^{(k)} : i \geq j$
32:         $j \leftarrow \operatorname{argmax}_{i \in N^{(k)},} V_i^{(k)}$
33:         $LD \leftarrow \bar{R} - V_j^{(k)}$
34:         $\bar{n}_e^{(k)} \leftarrow \lceil D_e^{(k)} \lambda_e^{(k)} \frac{LD}{LD - D_e^{(k)}} \rceil$
35:     end if
36: end procedure
```

Compute cloud VMs

```
37: procedure ComputeCloudVMs($V^{(k)}$, $N^{(k)}$, $\bar{R}$, $\delta_{II}^{(k)}$, $\tilde{B}$, $D_c^{(k)}$, $\lambda_c^{(k)}$, $y_i^c$, $\bar{n}_c^{(k)}$)
38:     if $\lambda_c^{(k)} > 0$ then
39:         $j \leftarrow \operatorname{argmax}_{i \in N^{(k)},\, y_i^c = 1} V_i^{(k)}$
40:         $LD \leftarrow \bar{R} - (V_j^{(k)} + \frac{\delta_{II}^{(k)}}{\tilde{B}})$
41:         $\bar{n}_c^{(k)} \leftarrow \lceil D_c^{(k)} \lambda_c^{(k)} \frac{LD}{LD - D_c^{(k)}} \rceil$
42:     end if
43: end procedure
```

Number of dep.	Dep. 1	Dep. 2	Dep. 3	Dep. 4	Dep. 5
$\lvert\mathcal{D}\rvert = 3$	3, -	1, -	1, 3		
$\lvert\mathcal{D}\rvert = 4$	3, -	1, -	1, 13	1, 3	
$\lvert\mathcal{D}\rvert = 5$	3, -	1, -	4, 13	1, 13	1, 3

Table 4: Selected splitting points for different number of deployments in the form *first splitting point, second splitting point.*

Once the deployment indices have been set, we can define all the parameters of the MEC system. In partic-

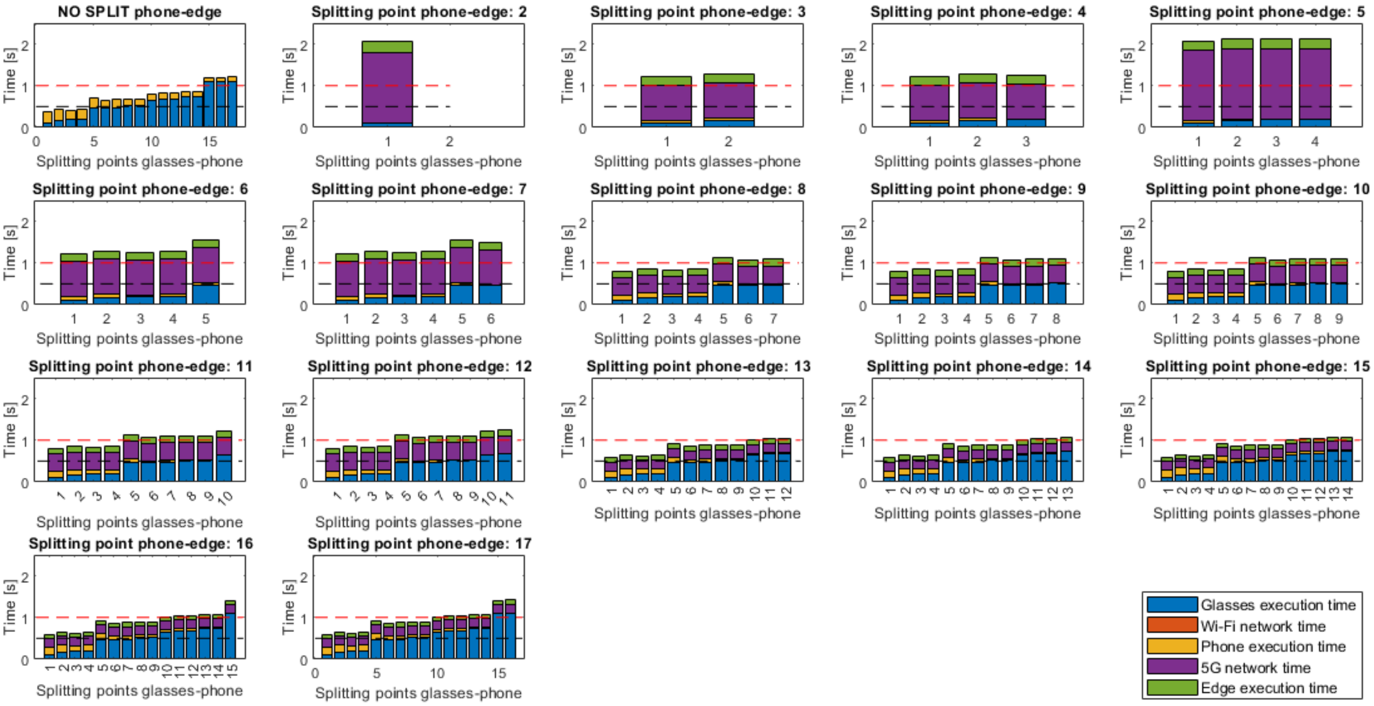


Figure 15: Execution time of a YOLOv4 neural network.

Algorithm 6 User Request and Profit Estimate

```
Input: $r$, $r_0$, $\gamma^{(k)}$, $\lambda$, $ordering$
Initialization: $Solutions \leftarrow []$
$r^{(k)} \leftarrow r_0 + \gamma^{(k)} r \ \forall k \in \mathcal{D}'$
$x \leftarrow$ Ask for chosen deployment given $r$
$\lambda^{(k)} \leftarrow \sum_{i \in \mathcal{U}} \lambda x_i^{(k)} \ \forall k \in \mathcal{D}'$
$orders \leftarrow$ Compute orders given $ordering$
for $order$ in $orders$ do
    $n_e^{(k)}, n_c^{(k)}, \lambda_e^{(k)}, \lambda_c^{(k)} \leftarrow$ OptimalResourceNumbers($order, \lambda^{(k)}, N_e$)
    $P \leftarrow$ Compute platform profit given $r, x, n_e^{(k)}, n_c^{(k)}$
    append $(P, r, x, n_e^{(k)}, n_c^{(k)}, \lambda_e^{(k)}, \lambda_c^{(k)}, order)$ to $Solutions$
end for
sort $Solutions$ by $P$ decreasingly
return the first element of $Solutions$
```

ular, Table 5 shows the settings for system parameters, including the number of users, the number of available edge servers, the average response time, execution time, and memory required per deployment, platform costs, and data size transferred between devices. Table 6 defines user parameters such as energy consumption and data transfer costs, execution times on local devices, power consumed when launching the application with a given deployment, application utilisation times, energy and memory available on devices.

Finally, Table 7 presents the fine-tuned parameters used in Algorithm 5 to achieve the results discussed in Section 5.5.1. These parameters, described in Section 4.5, were selected based on a grid search involving *Cutting fraction*, *ϵ scale* and *Fraction of initial points*, with the initial value of *Fraction of total points* set to 0.1. The search considered both possibilities of *First points approach* separately. It's worth noting that the sensitivity analysis in Section 5.5.2 demonstrates that while the the performance of the algorithm is robust against parameter variations, fine-tuning enhances its stability across different scenarios.

Experimental results

This section presents tables related to the experimental results discussed in Section 5. Tables 8, 9 and 10 depict the performance evaluation of the heuristic algorithm under full knowledge of users' parameters compared to the global solver BARON. Table 8 shows the profit ratio of the heuristic algorithm with the chosen order and that of BARON compared to the combinatorial heuristic algorithm. The "BARON solutions" column indicates the percentage of times BARON returns a feasible (not necessarily the best) solution within a maximum of two hours across 10 instances. The results indicate that for more than 100 users, the average profit of BARON is lower than that of the heuristic algorithm. This is due to its inability to converge to the optimal solution within the given time frame, and it sometimes fails to find a feasible solution for 200 users. Table 9 compares the execution times of the heuristic algorithm with that of BARON (limited to a maximum time of two hours), and shows in the column "BARON converged" the percentage of times BARON converges to the optimal solution (i.e., the times in which the solver stops before the maximum time set). BARON struggles to converge for more than 100 users, whereas the heuristic algorithm always returns the solution in less than one second (at most 0.25 seconds with the chosen order). To assess the optimality of the heuristic solutions, Table 10 compares the profits and convergence rates of BARON when initialized with solutions from the combinatorial algorithm. Specifically, the first column shows the profit ratio between the final solution returned by the global solver and that of the combinatorial algorithm. The second column reports the percentage of times the solver converges within two hours, while the last column indicates the percentage of cases where the solution

System Parameters	
N	In range $(50, 1000)$
λ	Uniform distribution in $[2, 4]$ req/s
N_e	$\frac{N}{20}$

$D_e^{(k)}$	D = 3: $D_e^{(3)} = 0.23$ s
	D = 4: $D_e^{(3)} = 0.12$ s, $D_e^{(4)} = 0.23$ s
	D = 5: $D_e^{(3)} = 0.12$ s, $D_e^{(4)} = 0.12$ s, $D_e^{(5)} = 0.23$ s

$D_c^{(k)}$	$\frac{5}{6} D_e^{(k)}\ \forall k \in \mathcal{D}'$
$m_g^{(k)}$	$5 - 4\frac{k}{\lvert\mathcal{D}\rvert}$ MB $\forall k \in \mathcal{D}$
$m_p^{(k)}$	$5 - 4\frac{k}{\lvert\mathcal{D}\rvert}$ MB $\forall k \in \mathcal{D}$
$\bar{R}$	Uniform distribution in $[2.75, 3.25]$ s
T	3600 s
c_c	Uniform distribution in $[1, 2]$ \$/h
c_e	$0.20\ c_c$
$\gamma^{(k)}$	$\gamma^{(1)} = \gamma^{(2)} = 0$, $\gamma^{(3)} = 1$, $\gamma^{(k)} = \frac{D_c^{(k)}}{D_c^{(k-1)}}\gamma^{(k-1)}$, for $k = 4, \ldots, \lvert\mathcal{D}\rvert$
ϵ	1×10^{-18}
$PopSize$	$10 * \lvert orders \rvert$
$\tilde{B}$	Uniform distribution in $[9216, 10240]$ Mbps

$\delta_{gp}^{(k)}$	D = 3: $\delta_{gp}^{(1)} = 2.64$ MB, $\delta_{gp}^{(2)} = 5.28$ MB, $\delta_{gp}^{(3)} = 5.28$ MB
	D = 4: $\delta_{gp}^{(1)} = 2.64$ MB, $\delta_{gp}^{(2)} = 5.28$ MB, $\delta_{gp}^{(3)} = 5.28$ MB, $\delta_{gp}^{(4)} = 5.28$ MB
	D = 5: $\delta_{gp}^{(1)} = 2.64$ MB, $\delta_{gp}^{(2)} = 5.28$ MB, $\delta_{gp}^{(3)} = 2.64$ MB, $\delta_{gp}^{(4)} = 5.28$ MB, $\delta_{gp}^{(5)} = 5.28$ MB

$\delta_{pe}^{(k)}$	D = 3: $\delta_{pe}^{(3)} = 2.64$ MB
	D = 4: $\delta_{pe}^{(3)} = 0.66$ MB, $\delta_{pe}^{(4)} = 2.64$ MB
	D = 5: $\delta_{pe}^{(3)} = 0.66$ MB, $\delta_{pe}^{(4)} = 0.66$ MB, $\delta_{pe}^{(5)} = 2.64$ MB

r_0	Uniform in $[8.30 \times 10^{-4}, 8.35 \times 10^{-4}]$ \$/s
r_{min}	5.56×10^{-4} \$/s
r_{max}	3×10^{-3} \$/s
M	100

Table 5: System parameters setting.

User Parameters	

$D_{g,i}^{(k)}$	D = 3: $D_{g,i}^{(1)} = 0.19$ s, $D_{g,i}^{(2)} = 0.10$ s, $D_{g,i}^{(3)} = 0.10$ s
	D = 4: $D_{g,i}^{(1)} = 0.19$ s, $D_{g,i}^{(2)} = 0.10$ s, $D_{g,i}^{(3)} = 0.10$ s, $D_{g,i}^{(4)} = 0.10$ s
	D = 5: $D_{g,i}^{(1)} = 0.19$ s, $D_{g,i}^{(2)} = 0.10$ s, $D_{g,i}^{(3)} = 0.21$ s, $D_{g,i}^{(4)} = 0.10$ s, $D_{g,i}^{(5)} = 0.10$ s

$D_{p,i}^{(k)}$	D = 3: $D_{p,i}^{(1)} = 0.23$ s, $D_{p,i}^{(2)} = 0.28$ s, $D_{p,i}^{(3)} = 0.57$ s
	D = 4: $D_{p,i}^{(1)} = 0.23$ s, $D_{p,i}^{(2)} = 0.28$ s, $D_{p,i}^{(3)} = 0.16$ s, $D_{p,i}^{(4)} = 0.06$ s
	D = 5: $D_{p,i}^{(1)} = 0.23$ s, $D_{p,i}^{(2)} = 0.28$ s, $D_{p,i}^{(3)} = 0.11$ s, $D_{p,i}^{(4)} = 0.16$ s, $D_{p,i}^{(5)} = 0.06$ s

$p_{g,i}^{(k)}$	$p_{g,i}^{(k)} = D_{g,i}^{(k)}$ W $\forall k \in \mathcal{D}$
$p_{p,i}^{(k)}$	$p_{p,i}^{(k)} = 40 D_{p,i}^{(k)}$ W $\forall k \in \mathcal{D}$
$\bar{B}_i$	Uniform distribution in $[9.6, 46.1]$ Gbps
B_i	Uniform distribution in $[9.9, 25.1]$ Mbps
$\bar{E}_{g,i}$	Uniform distribution in $[\min_k 0.9\lambda T_i^2 p_{g,i}^{(k)}, \max_k 1.2\lambda T_i^2 p_{g,i}^{(k)}]$ J
$\bar{E}_{p,i}$	Uniform distribution in $[\min_k 0.9\lambda T_i^2 p_{p,i}^{(k)}, \max_k 1.2\lambda T_i^2 p_{p,i}^{(k)}]$ J
$\bar{M}_{g,i}$	Uniform distribution in $[5\lvert\mathcal{D}\rvert, 6\lvert\mathcal{D}\rvert]$ MB
$\bar{M}_{p,i}$	Uniform distribution in $[5\lvert\mathcal{D}\rvert, 6\lvert\mathcal{D}\rvert]$ MB
T_i	50% of users uniform distribution in $[540, 660]$ s, 50% of users uniform distribution in $[1080, 1320]$ s
β_i	Uniform distribution in $[0.49, 0.52]$ \$/kWh
α_i	Energy dependent: $0.5+0.3\min\left(\frac{\bar{E}_{g,i} - \min_k 0.9\lambda T_i^2 p_{g,i}^{(k)}}{\max_k 1.2\lambda T_i^2 p_{g,i}^{(k)} - \min_k 0.9\lambda T_i^2 p_{g,i}^{(k)}}, \frac{\bar{E}_{p,i} - \min_k 0.9\lambda T_i^2 p_{p,i}^{(k)}}{\max_k 1.2\lambda T_i^2 p_{p,i}^{(k)} - \min_k 0.9\lambda T_i^2 p_{p,i}^{(k)}}\right)$, Fixed: 0.65, Random: uniform in $[0.5, 0.75]$
ζ_i	Uniform distribution in $[5.3 \times 10^{-5}, 6.0 \times 10^{-5}]$ \$/MB
U_i	Uniform distribution in $[8, 12]$ \$/h

Table 6: Users parameters setting.

returned by BARON is better than that of the heuristic algorithm. The results indicate that the solution returned by the combinatorial algorithm is improved in up to 20% of cases. On average, the highest improvement is observed in the scenario with 25 users and 4 deployments, where BARON achieves an average profit improvement of 1.66%.

Follow the line approach parameters	
Cutting fraction cut	0.1
First points approach	Equispaced
ϵ_{scale}	2
Fraction of initial points η_{init}	0.06
Fraction of total points η_{tot}	0.1

Table 7: Fine-tuned configuration of the Follow the line algorithm.

N	**Profit ratio Chosen order** (%)			**Profit ratio BARON** (%)			**BARON solutions** (%)		
	$\lvert\mathcal{D}\rvert= 3$	$\lvert\mathcal{D}\rvert= 4$	$\lvert\mathcal{D}\rvert= 5$	$\lvert\mathcal{D}\rvert= 3$	$\lvert\mathcal{D}\rvert= 4$	$\lvert\mathcal{D}\rvert= 5$	$\lvert\mathcal{D}\rvert= 3$	$\lvert\mathcal{D}\rvert= 4$	$\lvert\mathcal{D}\rvert= 5$
10	0.00	-1.07	-1.77	-0.07	1.34	0.00	100	100	100
25	0.00	-2.65	-2.77	0.40	1.66	-0.03	100	100	100
50	0.00	-0.70	-1.22	0.77	0.02	0.07	100	80	100
100	0.00	0.00	0.00	-0.42	-2.94	-7.24	100	100	100
200	0.00	-0.06	0.00	-3.92	-6.53	-9.64	80	30	70

Table 8: Performance of the heuristic algorithm and BARON without initialization.

N	**Execution time Combinatorial** (s)			**Execution time Chosen order** (s)			**Execution time BARON** (s)			**BARON converged** (%)		
	$\lvert\mathcal{D}\rvert= 3$	$\lvert\mathcal{D}\rvert= 4$	$\lvert\mathcal{D}\rvert= 5$	$\lvert\mathcal{D}\rvert= 3$	$\lvert\mathcal{D}\rvert= 4$	$\lvert\mathcal{D}\rvert= 5$	$\lvert\mathcal{D}\rvert= 3$	$\lvert\mathcal{D}\rvert= 4$	$\lvert\mathcal{D}\rvert= 5$	$\lvert\mathcal{D}\rvert= 3$	$\lvert\mathcal{D}\rvert= 4$	$\lvert\mathcal{D}\rvert= 5$
10	0.03	0.04	0.06	0.03	0.04	0.04	821.31	37.01	339.60	90	100	100
25	0.05	0.10	0.13	0.05	0.09	0.09	2895.69	3151.19	2665.85	60	70	70
50	0.10	0.13	0.18	0.10	0.11	0.10	5777.92	6419.36	6559.40	20	20	10
100	0.13	0.18	0.29	0.12	0.16	0.17	6891.10	7205.66	7207.90	10	0	0
200	0.18	0.29	0.51	0.18	0.23	0.25	7207.03	7213.40	7221.31	0	0	0

Table 9: Performance of the heuristic algorithm and BARON without initialization.

N	**Profit ratio BARON initialized** (%)			**BARON initialized converged** (%)			**Initial solution improved** (%)		
	$\lvert\mathcal{D}\rvert= 3$	$\lvert\mathcal{D}\rvert= 4$	$\lvert\mathcal{D}\rvert= 5$	$\lvert\mathcal{D}\rvert= 3$	$\lvert\mathcal{D}\rvert= 4$	$\lvert\mathcal{D}\rvert= 5$	$\lvert\mathcal{D}\rvert= 3$	$\lvert\mathcal{D}\rvert= 4$	$\lvert\mathcal{D}\rvert= 5$
10	0.32	1.34	0.00	100	100	100	20	20	10
25	0.35	1.66	0.00	100	100	90	10	20	0
50	0.79	0.23	0.17	100	80	100	10	10	20
100	0.00	0.00	0.29	100	90	90	0	0	20
200	0.00	0.00	0.00	100	100	90	0	0	0

Table 10: Performance of BARON with initialization.