Model Placement for Quality Inference of Video Streaming Traffic over a Cellular Network

Francescomaria Faticanti*, Loïc Desgeorges*, Rémi Watrigant*, Thomas Begin*, Francesco Bronzino*†
*ENS de Lyon, CNRS, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France
†Institut universitaire de France
Email: {firstname.lastname}@ens-lyon.fr

***Abstract*—Monitoring the quality of streaming video applications is important for Internet service providers (ISPs) to detect network issues and facilitate capacity planning. Machine Learning (ML) inference models have emerged as an effective solution to determine service quality using network traffic. However, while much focus has been on enhancing model performance, little attention has been given to deploying these models across entire networks. This paper introduces a new placement approach of quality inference models and their associated tasks to enhance the monitoring of video streaming applications over an entire mobile traffic network. Starting from the observation that inference tasks require the deployment of multiple components to, first, calculate input features from raw traffic, and then execute the inference models, we define the placement problem as an integer programming problem and, given its NP-hardness, we provide a heuristic solution, experimentally close to the optimum, based on the relaxation and the rounding of fractional solutions. We highlight that decoupling these components for the inference of network traffic can be beneficial in terms of total accuracy of the ML inference tasks. Finally, we experimentally show that our solution outperforms state-of-the-art placement techniques by ~30% of accuracy of the deployed inference models.**

I. Introduction

Video streaming is the dominant application on today's Internet, representing over 65% of all network traffic [1]. For Internet Service Providers (ISPs), monitoring video Quality of Experience (QoE) metrics is essential for making informed, fine-grained decisions about network operations, such as capacity planning and resource provisioning [2]. However, with the widespread adoption of end-to-end encryption, ISPs can no longer directly access video quality metrics from traffic data [3]. Recent research suggests that data-driven insights using Machine Learning (ML) models offer a promising alternative for monitoring traffic quality [4]. While significant efforts have focused on enhancing the predictive capabilities of these models, there has been limited consideration of how different model deployment strategies may impact overall model performance.

ML-based traffic analysis pipelines involve multiple functional stages, typically beginning with the ingestion of raw network traffic data and ending with a prediction of specific quality characteristics, such as resolution or startup time. In the initial stages, raw traffic data is captured and transformed to derive features and representations that serve as input for the ML model [5], [6]. Subsequently, these features are fed into inference models to predict the quality of the monitored video [4], [7]. Each stage in the pipeline has specific computational and networking requirements that depend on the chosen inference model. The demands for processing raw traffic vary with the type of representations required: simpler representations (e.g., packet counters or average throughput) are less resource-intensive than more complex metrics like tracking packet loss within a TCP flow [5]. Additionally, model execution entails memory and processing requirements that depend on the size and complexity of the ML model used for inference. When deploying such a pipeline, it is essential to account for all these requirements to ensure optimal performance and resource allocation. It is worth to stress, as we discuss in detail in Section IV, that such a monitoring process does not have a significant impact on the actual quality metrics that ISPs aim to infer.

To accurately monitor video quality metrics, the inference pipeline must be deployed within network infrastructure resources. Given the high volume of traffic passing through the network, a centralized processing approach (e.g., cloud computing) would quickly lead to network congestion, rendering necessary to solely leverage edge resources. Unfortunately, the limited computational capacity at the edge demands careful deployment strategies that starkly differ from the placement strategies present in the literature. Standard applications or virtual network placement solutions presented in the literature [8] are not suited to the large number of different model variants available (given by the model type, the hardware requirements, or the subset of hyperparameters chosen for training of the model [9]). This adds new levels of computational complexity to the search space of the optimal placement. There exists more recent work focused on the specific challenges of model placement [9]–[11]. However, these approaches primarily address the placement and execution of the inference model itself, without considering the essential steps required to preprocess and transform raw network traffic for input to the model. As a result, they overlook the added complexity of handling traffic data ingestion, feature extraction, and data-transformation processes that are critical for accurate inference but place additional demands on network and computational resources.

In this paper, we show that decoupling the stages of the inference pipeline leads to a higher deployment flexibility, enabling operators to achieve higher monitored accuracy in the estimation of video quality metrics from encrypted traffic. We formulate such a problem and prove that it is NP-hard. We also propose an integer linear programming formulation and design

an efficient heuristic that presents near-optimal performance. Overall, the novelty of this work resides in the introduction and the study of the inference placement problem for video streaming. Our main contributions are summarized as follows:
1) We introduce the problem of inference placement for the monitoring of video streaming traffic at the edge of the network and provide a formalization of such a problem and a characterization of its computational complexity (Section II).
2) We design a heuristic solution based on the relaxation of the integer problem and on a rounding procedure for the fractional variables (Section III).
3) We prove on realistic cellular data that such an approach outperforms the state-of-the-art idea of treating monitoring as a unique and atomic module, showing an improvement of ~30% of accuracy of the inference models deployed (Section IV).

II. Inference Placement Problem

A. Description

The goal of a video quality monitoring pipeline is to estimate with the highest possible accuracy the quality of the video flows received by the users of the network. Quality metrics such as startup delay and video resolution affect the users' experience and represent the main target for monitoring pipelines applied to video flows [4]. The monitoring is performed through two main components that correspond to the previously described stages: i) the capture, processing, and transformation of raw traffic into data representations; we call such component the feature extractor (FE). ii) The execution of the inference model; we call such component the inference model (IM). The final accuracy of the pipeline strictly relates to the accuracy of the model chosen to perform the inference. More complex models achieve better accuracy but might exceed available resources rendering impossible to process all video flows traversing the network.

Figure 1 depicts an example of the main problem applied to a cellular network. In this context, we have an infrastructure composed by different base stations (BSs). Such BSs are co-equipped with servers offering computational capabilities (CPU, RAM). Different BSs can be interconnected in different ways depending on the particular topology of the cellular network. In this work, we consider that BSs are divided into clusters. Within the cluster BSs are directly connected by a switch in a star topology (we show two clusters in Figure 1). This represents the typical structure of urban cellular networks [12], [13].

In this scenario, a video service provider delivers video flows (illustrated by the red line in the figure) to users via the cellular network. Each flow reaches a BS for delivery to users. To monitor these flows, the network operator deploys the pipeline comprising the Feature Extractor and Inference Model on the network. The FE of a given flow can be placed on any BS within the same cluster of the BS it arrives to. If the FE is located on a different BS from the one receiving the original traffic, the flow must be mirrored from the switch to the designated BS, incurring bandwidth consumption on the link between the switch and the BS (see the green line in

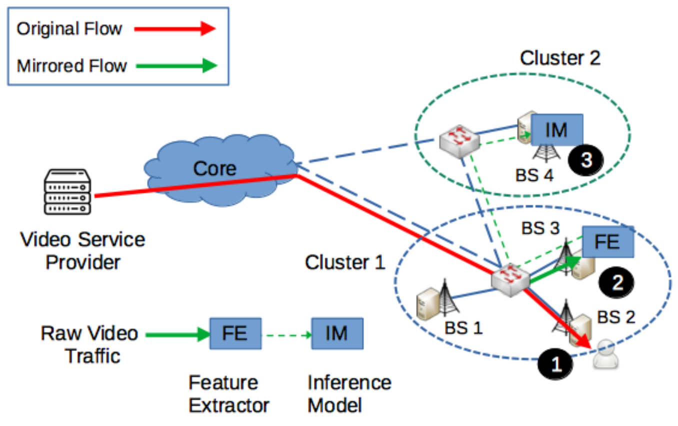


Fig. 1: To monitor the video quality of the original video flow (❶), the ISP must deploy an inference pipeline that consists of a feature extractor (❷) and an inference model (❸) with a reduced traffic between the two components (dashed green line). Since the FE is deployed on a different BS with respect to the BS where the original traffic arrives (red line), the video flow is mirrored (green line) on the BS where the FE is placed.

the Figure 1). The IM, in contrast, can be deployed on any BS, as the traffic representations produced by the FE render the data transfer negligible in terms of bandwidth [5]. If these components are deployed over the infrastructure, i.e., there are enough resources to host them, the monitoring process will have a negligible impact on the quality of videos received by the users as we discuss in Section IV. Note that in this work we do not account for the possibility of processing traffic features on the network fabric itself (e.g., programmable switches), leaving it for future work.

As we will demonstrate, separating raw traffic processing from inference offers advantages in terms of i) flexibility in resource utilization and ii) accuracy of the model estimates. Note that this setup has no latency requirements for pipeline operations, as the ISP's goal is to monitor traffic flows to assess their quality. Furthermore, in the following section, we formulate an offline problem for the monitoring of video streaming traffic. Indeed, the decision on the placement of inference pipelines is based on i) the current traffic observed at the time of decision, and ii) the historical data concerning the resource usage of the pipeline components needed for monitoring each flow. The online setting (i.e., with no assumption or prior knowledge on the resource usage of video flows and their FEs and IMs) with forecasting is left to future work.

B. Model

Infrastructure. We consider an infrastructure consisting of a set of BSs. Each BS is associated with an edge server with a fixed computational capacity as shown in [14]. BSs are divided in clusters, and, within each cluster, each BS is connected to a switch through a direct link. Each link connecting the BSs has a given residual bandwidth capacity [15]. The topology of each cluster is a star topology where the switch is the center of the star, as shown in Figure 1. The raw video traffic arrives at the switch and then it is routed to the desired BS. Hence, in case of mirroring the original traffic of a given flow, we can assume that the first point of mirroring is the switch and there is a consumption of bandwidth between the switch and the BS where the FE of the flow is placed. Indeed, to avoid bandwidth consumption across the core network, we require that the FE for each flow is deployed within the cluster of the BS where the original video flow arrives. Given that there is only one link between the switch and each BS within each cluster (in a star topology), the bandwidth availability can be seen as a property (or capacity) of each BS. This observation allows us to model the network simply as a set of nodes $\mathcal{N}$, i.e., the set of BSs (or nodes). Each node $n \in \mathcal{N}$ has a memory, a processing and a bandwidth capacity: $C_n^{mem}, C_n^{cpu}, B_n \in \mathbb{R}^+$, respectively. In what follows, we indicate with "capacity" the residual availability of resources in a given node (in terms of CPU, RAM or bandwidth).

Video Flows. Video flows (or streams) are sent by the service provider, through the core network, to the BSs. The set of flows is indicated with $\mathcal{F}$. Each flow $f \in \mathcal{F}$ is characterized by its throughput λ_f and by the *original* BS where the raw video traffic arrives, denoted by s_f (e.g, in Figure 1 the original BS of the red flow is $BS2$). Further, we indicate with $C_f \subseteq \mathcal{N}$ the cluster s_f belongs to. As stated before, for each flow f, we require the FE for f to be placed within C_f. In this manner, if the FE for f is placed on a node $n \in C_f$ different from s_f, the ISP must mirror the traffic from s_f to n, leading to a bandwidth consumption.

Inference Pipeline. The inference process, for each flow $f \in \mathcal{F}$, consists of two components to be placed on the infrastructure. The first one is the FE which takes in input the raw traffic and extracts all the features needed for the inference. It is characterized by CPU and memory requirements, μ_f^{cpu} (which depends on λ_f) and μ_f^{mem}, respectively. This component significantly reduces the traffic that is sent to the IM. The second component is the IM that performs the inference task on the data received by the FE. For each flow $f \in \mathcal{F}$, the inference can be executed by a model chosen from a set $\mathcal{M}_f$ of possible models. Each model $m \in \mathcal{M}_f$ is defined by the memory occupancy (or size) μ_m^{mem}, the CPU requirements μ_m^{cpu}, and the level of accuracy α_m at the training phase. From the modelling perspective, two different configurations of the same model for the inference of a flow f can be considered as two distinct models in $\mathcal{M}_f$. We assume, for each inference model, that the level of accuracy is an increasing function of the size since, in general, the bigger is the size of the model and the higher is its accuracy. Furthermore, we assume that the throughput between the FE and IM of each pipeline is negligible since the raw traffic is significantly reduced after being processed by the FE module. Bronzino et al. [5] showed that the amount of traffic is reduced by an order of magnitude of 10^4 for video inference, leading to a negligible throughput on a network link with 10 Gbps of bandwidth. In what follows we will say that a flow is "*monitored*" if both the FE and the IM for that flow are placed on the network. Given the model described above, the *Inference Placement Problem* (IPP) is described in Figure 2.

Inference Placement Problem (IPP)

Input
(I1) Set of video flows $\mathcal{F}$.
(I2) Set of ML models $\{\mathcal{M}_f\}_{f\in\mathcal{F}}$ for IMs.
(I3) Network $\mathcal{N}$.
Output
(O1) Subset $\mathcal{F}' \subseteq \mathcal{F}$ of video flows monitored through the deployment of FE and IM modules.
(O2) One FE placed on a node in C_f, $\forall f \in \mathcal{F}'$.
(O3) One IM in $\{\mathcal{M}_f\}_{f\in\mathcal{F}}$ placed on a node in $\mathcal{N}, \forall f \in \mathcal{F}'$.
Constraints
(C1) *Computational capacity*. The total resource occupation of all the FEs and IMs placed on a node $n \in \mathcal{N}$ must not exceed the capacity of n.
(C2) *IM and FE*. Each flow belongs to $\mathcal{F}'$ if and only if both the FE and the IM are deployed on the infrastructure.
(C3) *Bandwidth*. The total amount of throughput of flows mirrored to a BS must not exceed the bandwidth capacity of the BS.
Optimization goal
(G) *Maximize the total accuracy* given by the sum of the accuracies of the IMs instantiated for the flows in $\mathcal{F}'$.

Fig. 2: IPP Statement

C. Computational Complexity of IPP

Proposition 1. *IPP is NP-hard.*

Proof: The proof follows by reduction from the well-known Multiple Knapsack Problem (MKP) [16]. Given an instance of MKP with m knapsacks and n objects where each item i has profit p_i and weight w_i, and each knapsack j has capacity C_j, we can build an instance of IPP with $|\mathcal{N}| = m$, $|\mathcal{F}| = n$ and $|\mathcal{M}_i| = 1$ for each $i \in \mathcal{F}$. Further, we set $\mu_i^{cpu} = 0$, $\mu_i^{mem} = 0$ for each $i \in \mathcal{F}$, $\mu_m^{mem} = w_i$ and $\alpha_m = p_i$ for each $m \in \mathcal{M}_i$, and $\lambda_i = 0$ for each $i \in \mathcal{F}$. The described mapping is the desired polynomial reduction. ■

III. Proposed Approach

A. Integer Linear Programming Formulation

The following integer linear programming (ILP) problem formulates the problem described in Figure 2.

Variables. We define two kinds of binary variables. The first one for the placement of the FE of each flow among the set of nodes (or BSs); the second one represents the instantiation of the IM of each flow on a given node (or BS) of the network.

1) FE placement. We define $y_{f,n} \in \{0, 1\}$, $\forall f \in \mathcal{F}, \forall n \in \mathcal{N}$. Such a variable is equal to 1 if the FE for f is placed on n. Since we only deploy the FE of a given flow within its cluster, we force $y_{f,n}$ to 0 whenever $n \notin C_f$ (as expressed later in constraint (5)).

2) IM instantiation. The IM instantiation is represented by the definition of $x_{f,m,n} \in \{0, 1\}$, $\forall f \in \mathcal{F}, \forall m \in \mathcal{M}_f, \forall n \in \mathcal{N}$. The variable is set to 1 if model m is placed on n for flow f.

Constraints. The problem's constraints are formally defined as follows.

C1. Server capacity:

$$\sum_{m\in\mathcal{M}_f}\mu^r_m\sum_{f\in\mathcal{F}}x_{f,m,n}+\sum_{f\in\mathcal{F}}\mu^r_f y_{f,n}\le C^r_n,$$
$$\forall n\in\mathcal{N},\forall r\in\{cpu,mem\}. \quad (1)$$

C2. If the IM is instantiated for flow $f\in\mathcal{F}$ then also the FE is deployed and vice versa:

$$\sum_{n\in\mathcal{N}}y_{f,n}=\sum_{m\in\mathcal{M}_f}\sum_{n\in\mathcal{N}}x_{f,m,n},\quad\forall f\in\mathcal{F}. \quad (2)$$

At most one model is selected for each stream (flows can be discarded):

$$\sum_{m\in\mathcal{M}_f}\sum_{n\in\mathcal{N}}x_{f,m,n}\le 1,\quad\forall f\in\mathcal{F}. \quad (3)$$

C3. Bandwidth constraints in case of mirroring video traffic from the source of the flow to the BS where the feature extractor for the flow is placed. The constraint only applies for the flows whose source is not the considered BS:

$$\sum_{f\in\mathcal{F}|s_f\neq n}\lambda_f y_{f,n}\le B_n,\quad\forall n\in\mathcal{N}. \quad (4)$$

The FE of each flow f can only be placed within C_f. Such a requirement is expressed by the following constraint:

$$y_{f,n}\le 0,\quad\forall f\in\mathcal{F},\forall n\in\mathcal{N}\setminus C_f. \quad (5)$$

Objective (G). The main metric that ISPs aim to maximize is the total accuracy achieved by the models placed on the infrastructure to monitor the traffic flows, i.e.,

$$\sum_{f\in\mathcal{F}}\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}_f}\alpha_m x_{f,m,n}. \quad (6)$$

B. Inference Rounding Algorithm (IRA)

Given the NP-hardness of IPP, the objective is to design an efficient heuristic solution. Indeed, IPP shares similarities with Multiple Multidimensional Knapsack problems [16]. Such problems have been proved hard to approximate and heuristic solutions have been provided [17]. In particular, our solution is based on i) the relaxation of the ILP described in Section III-A, and ii) the rounding of the fractional values obtained from the relaxation. The reason behind such an algorithmic choice is twofold. First, as it will be shown in Section IV, the relaxation of the ILP represents a good starting point for the design of a heuristic solution in terms of both accuracy and computational complexity. Secondly, such a technique is one of the most adopted to provide efficient heuristic algorithms for NP-hard problems that present similarities with Multiple Multidimensional Knapsack problems [16].

Main idea. As we will show in Section IV, if we solve the relaxed version of IPP in practice we obtain a solution with a low percentage of fractional variables. This means that, by simply solving the linear program (LP) obtained by the relaxation, we already have a valid heuristic solution that is not far from the optimal one. Hence, the main idea of the solution is to solve the LP using off-the-shelf solvers, and then rounding the fractional values of the decision variables. After solving the relaxation, all the non-zero resulting variables are either set to 1 or to a fractional value in $(0,1)$. More formally, we indicate with $\hat{X}$ and $\hat{Y}$ all the variables that are set to a value different from zero by the resolution of the LP. In particular, $\hat{X}=\hat{X}_I\cup\hat{X}_F$ and $\hat{Y}=\hat{Y}_I\cup\hat{Y}_F$ where $\hat{X}_I$ (resp. $\hat{Y}_I$) is the set of all x's (resp. y's') variables set to 1, and $\hat{X}_F$ (resp. $\hat{Y}_F$) is the set of all x (resp. y) variables whose value lies in $(0,1)$. With an abuse of notation we indicate with $f\in\hat{X}_I$ (resp. $f\in\hat{X}_F$) if there exist a node n and a model m such that $\hat{x}_{f,m,n}=1$ (resp. $\hat{x}_{f,m,n}\in(0,1)$). The same notation applies for $\hat{Y}_I$ and $\hat{Y}_F$. The problem formulation of IPP implies that if a flow $f\in V_F=\hat{X}_F\cup\hat{Y}_F$ then there are three possible cases for the decision variables concerning f: *case i)* $f\in\hat{X}_I$ and $f\in\hat{Y}_F$, i.e., the place of the IM for f has been fixed and the FE's placement is left; *case ii)* $f\in\hat{X}_F$ and $f\in\hat{Y}_F$, i.e., both the placements of IM and FE are left; *case iii)* $f\in\hat{X}_F$ and $f\in\hat{Y}_I$, i.e., the placement of FE is fixed and the placement and the model for the inference should be determined.

Algorithmic solution. The Inference Rounding Algorithm (IRA) follows the observations made above and it consists of two parts: i) the resolution of the LP of the relaxed version of IPP; ii) the rounding procedure based on the three possible cases for the fractional decision variables. The relaxed version of IPP is obtained by simply substituting the integrality constraints defined above (*FE Placement* and *IM instantiation*) with $x_{f,m,n}\in[0,1]$ and $y_{f,n}\in[0,1]$, respectively. The pseudocode of IRA is shown in Algorithm 1.

Starting from the fractional solution of the linear program $(\hat{\mathbf{x}},\hat{\mathbf{y}})$ (line 3 of Algorithm 1), the algorithm monitors each flow $f\in\hat{X}_I\cap\hat{Y}_I$, i.e., all the flows for which both the FE and the IM are entirely placed by the resolution of the LP, are *monitored* by IRA. The set of such flows represents the starting solution. Then, the algorithm analyses all the fractional variables in V_F trying to improve the total accuracy of the starting solution by increasing the number of monitored flows. According to *case i)*, it first selects all the flows for which $\hat{x}$ is set to 1 and $\hat{y}$ has fractional values (line 6 in Algorithm 1). The algorithm starts from this subset of variables since it is the subset that can certainly increase the objective function. For the fractional values of $\hat{y}$ the algorithm attempts to deploy entirely the FE on the location with the highest fractional value on $\hat{y}$ (line 10 in Algorithm 1). After this step, IRA (line 11 in Algorithm 1) analyses the subset of flows for which both $\hat{x}$ and $\hat{y}$ are fractional (second case). Finally, it proceeds with the subset where $\hat{x}$ is fractional and $\hat{y}$ is set to 1. For the model selection and placement, the algorithm chooses the couple (model and node) with the highest value of the fractional variable $\hat{x}$ (line 15 and line 20 in Algorithm 1). Indeed, the fractional values (of both $\hat{x}$ and $\hat{y}$) are interpreted as probabilities so that FEs and IMs are placed in the most likely location and with the most likely model. The procedure ALLOCATE simply checks for the availability of resources to entirely place the IM or the FE for a given flow f. In case of enough space for the component, it allocates the IM or the FE on the established node, updates the residual resources and the flow is monitored. Otherwise the flow is discarded.

Algorithm 1: Inference Rounding Algorithm (IRA)

```
1: Input: Set of video flows $\mathcal{F}$, family of models $\{M_f\}_{f\in\mathcal{F}}$, network graph $G=(\mathcal{N},\mathcal{E})$.
2: Output: Inference placement $(\mathbf{x},\mathbf{y})$.
3: $\hat{\mathbf{x}},\hat{\mathbf{y}} \leftarrow$ solve LP of IPP
4: $x_{f,m,n}, y_{f,n} \leftarrow \hat{x}_{f,m,n}, \hat{y}_{f,n}, \forall n\in\mathcal{N}, m\in\mathcal{M}_f$
5: for $f\in V_F$ do
6:     if $f\in\hat{X}_I \wedge f\in\hat{Y}_F$ then
7:         $E \leftarrow \{n\in\mathcal{N}|0<\hat{y}_{f,n}<1\}$
8:         $\tilde{n} \leftarrow \arg\max_{n\in E}\{\hat{y}_{f,n}\}$
9:         $m,n \leftarrow m,n$ s.t. $\hat{x}_{f,m,n}=1$
10:        $x_{f,m,n}, y_{f,n}, G \leftarrow$ ALLOCATE$(G,f,\hat{x},\hat{y},\tilde{n},m,n)$
11:    else if $f\in\hat{X}_F \wedge f\in\hat{Y}_F$ then
12:        $E \leftarrow \{n\in\mathcal{N}|0<\hat{y}_{f,n}<1\}$
13:        $\tilde{n} \leftarrow \arg\max_{n\in E}\{\hat{y}_{f,n}\}$
14:        $L \leftarrow \{(n,m)\in\mathcal{N}\times\mathcal{M}_f|0<\hat{x}_{f,m,n}<1\}$
15:        $n,m \leftarrow \arg\max_{(n,m)\in L}\{\hat{x}_{f,m,n}\}$
16:        $x_{f,m,n}, y_{f,n}, G \leftarrow$ ALLOCATE$(G,f,\hat{x},\hat{y},\tilde{n},m,n)$
17:    else
18:        $\tilde{n} \leftarrow n$ s.t. $\hat{y}_{f,n}=1$
19:        $L \leftarrow \{(n,m)\in\mathcal{N}\times\mathcal{M}_f|0<\hat{x}_{f,m,n}<1\}$
20:        $n,m \leftarrow \arg\max_{(n,m)\in L}\{\hat{x}_{f,m,n}\}$
21:        $x_{f,m,n}, y_{f,n}, G \leftarrow$ ALLOCATE$(G,f,\hat{x},\hat{y},\tilde{n},m,n)$
22:    end if
23: end for
24: return $(\mathbf{x},\mathbf{y})$
```

Time Complexity. The complexity of IRA is dominated by the complexity to solve the LP relaxation of IPP, i.e., polynomial in the input size. Indeed, the algorithm performs constant operations on the number of fractional variables and, as we will show in Section IV, such a number represents a small percentage of the total number of variables.

IV. Evaluation

A. Experimental Settings

We evaluate the proposed solution using the NetMob dataset described in [15], which contains mobile network traffic volumes generated by various services. The experimental analysis includes: i) a comparison of our solution with baselines, including the state-of-the-art approach that treats the FE and IM as a single component [10], [13], and ii) an assessment of the optimality and efficiency of our rounding algorithm relative to the integer optimal solution. Both this optimal solution and the relaxed version of IPP are computed using Gurobi [18].

Data. The NetMob dataset contains traffic volumes generated by 68 different mobile services (e.g., Netflix, Facebook Live) over 20 metropolitan areas in France monitoring during 77 consecutive days in 2019 [15]. We focus our evaluation on the area of Lyon and select Neflix traffic as a representative scenario. The dataset only reports traffic volumes generated from each tile of the metropolitan area, where a tile represents an area of $100\times 100\ m^2$, without mentioning the position of the BSs associated to each tile. Hence, we use the data from the French National Agency of Frequencies [19] to retrieve the coordinates of each BS of the area. Then we associate each tile t to the BS b that minimizes Euclidean distance between t and b. To obtain the total traffic of a given BS, we sum all the traffic coming from each tile associated to that BS.

Network and topology. We have $|\mathcal{N}|=418$ BSs connected according to the topology discussed in [12]. To create the star-like subgraphs that connect the BSs we create $\lceil\frac{|\mathcal{N}|}{6}\rceil$ clusters of BSs using the k-means algorithm based on the Euclidean distance between BSs. In each cluster, the bandwidth between the central switch and each BS is uniformly generated from the set $\{100,400,600,1000,5000,10000\}$ Mbps. We assume that each BS is equipped with a server with computational capabilities. The memory capacity of each BS is randomly selected from the set $\{4000,8000,12000,16000\}$ MB, and the CPU capacity is randomly sampled from $\{1,2,3\}$ GHz.

Video flows. Given the total amount of traffic on each BS we separate video flows f in the BS by randomly picking a rate λ_f from $\{3,5,10,15\}$ Mbps that corresponds to the average throughput at different video resolutions (480p, 720p, 1080p and 4K, respectively) [4]. To determine the total CPU requirement (in terms of number of packets per second) for each flow f we, first, randomly generate the CPU cycles τ_f required to process 1 packet by the FE from the set $[100,400]$ CPU cycles, in line with the values shown in Wan et al. [6]). Then we get the number of packets to be processed by the FE per second $\mu_f^{cpu}=\lceil\frac{\tau_f}{\iota}\rceil$, where ι represents the length of a packet in bits. In the evaluation we set $|\mathcal{M}_f|=10$ models for each flow f with accuracy and size $(\alpha_m,\mu_{f,m}^{mem})$ uniformly sampled from $\{(98,2000),(90,1500),(85,1000),(80,950),(70,700),(65,600),(60,500),(40,300),(30,100),(20,50)\}$ (%,MB). These values are in line with the accuracy values and sizes presented in the literature [5], [10], [13].

Baselines. Given the lack of solutions in traffic monitoring that can be used as a direct comparison, as it will be discussed in Section V, we compare our solution against two baselines.

NoSplit (SotA). To highlight the advantages of splitting the FE and the IM, we compare it against a solution that constrains these two components to be located on the same BS. This represents the common assumption from the state-of-the-art to consider the FE and the IM as a unique component deployed on the infrastructure [10], [13]. To implement such a solution we simply modify the problem defined in Section III-A by adding the following constraints:

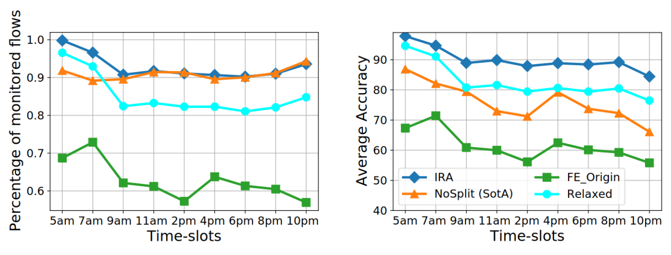
$$\sum_{m\in\mathcal{M}_f} x_{f,m,n}=y_{f,n},\quad \forall n\in\mathcal{N},\forall f\in\mathcal{F}. \tag{7}$$

We solve the relaxation of the program and then we round according to all the constraints.

FE_Origin. This represents an extreme solution where the FE of each flow f is constrained to be placed on the original BS s_f of flow f. In this way, no bandwidth is consumed to monitor each flow. The solution is obtained by adding the following constraint to the formulation of IPP:

$$y_{f,n}\le 0,\quad \forall f\in\mathcal{F},\forall n\in\mathcal{N}\setminus\{s_f\}. \tag{8}$$

Metrics. We compare all the solutions with respect to two main metrics: i) the *Average Accuracy* per flow defined as



(a) % Monitored flows (b) Average Accuracy

Fig. 3: `IRA` vs. Baselines.

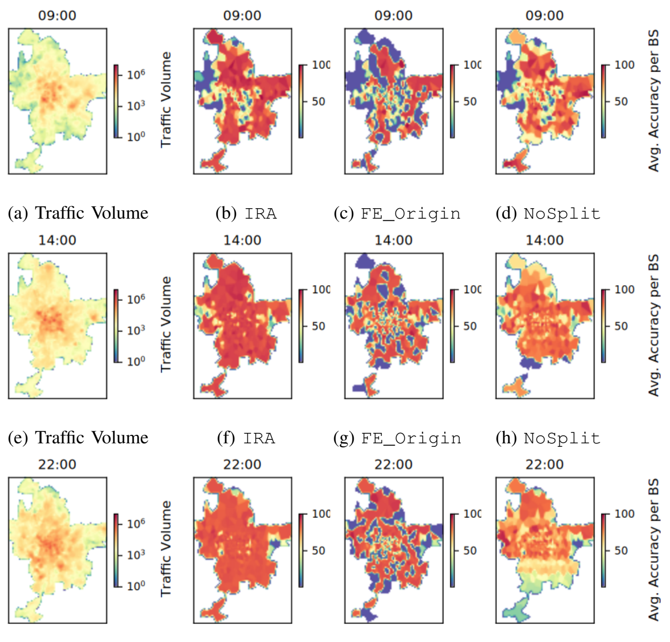
$(\sum_{f\in\mathcal{F}}\sum n\in\mathcal{N}\sum_{m\in\mathcal{M}_f}\alpha_m\cdot x_{f,m,n})/|\mathcal{F}|$; ii) the *Percentage of monitored flows* by each solution with respect to the total number of flows $|\mathcal{F}|$, defined as $(\sum_{f\in\mathcal{F}}\sum_{n\in\mathcal{N}}y_{f,n})/|\mathcal{F}|$. Note that the two metrics relate closely as unprocessed flows account for 0% accuracy, as presented in Section II. In all the figures of the experiments we report the average for the metric of interest as computed over 30 different runs. In each run a new set of flows is generated as described above.

B. Results

Comparison against SotA. We compare `IRA` against `NoSplit` and `FE_Origin` in terms of the average accuracy achieved by each solution in different time-slots of the day. We also depict the `Relaxed` solution that represents solution obtained by solving the relaxation of IPP and considering solely the variables set to 1. For all the experiments we consider the first 15 minutes of traffic of each hour (the time series collected in the dataset, for each application, is divided in 15-minute time steps [15]). Figure 3 reports the percentage of monitored flows and the average accuracy (Figure 3a and Figure 3b, respectively). First, we observe that the number of monitored flows of `Relaxed` is already high and `Relaxed` is a good starting point for a heuristic solution. This means that the number of fractional variables obtained by relaxing problem IPP is low compared to the total number of variables. This confirms the goodness of `IRA` that starts from the solution of `Relaxed` and rounds the remaining fractional variables.

In Figure 3b we notice the benefit of splitting the FE and the IM for the monitoring of flows with `IRA` outperforming `NoSplit` with a gap of ~30% of accuracy. Indeed, constraint (7) forces the IM and the FE to be deployed on the same node consuming more resources on the node and hindering the deployment of more accurate models. Furthermore, the percentage of monitored flows reached by `Relaxed` and `NoSplit` in Figure 3a suggests that is not always necessary to monitor a greater number of flows to have higher accuracy.

The choice of deploying FEs in different places with respect to the original BSs represents an issue in terms of consumed bandwidth across the network. However, Figure 3 highlights the limitation of deploying all the FEs in the original BSs. This leads to a lower percentage of monitored flows (Figure 3a), thus lowering average accuracy (Figure 3b). Indeed, constraint (8) of `FE_Origin` leads to saturate all the original BSs with the FE components and hinders the deployment of high-accuracy IMs. This behaviour is better highlighted in



(a) Traffic Volume (b) `IRA` (c) `FE_Origin` (d) `NoSplit`
(e) Traffic Volume (f) `IRA` (g) `FE_Origin` (h) `NoSplit`
(i) Traffic Volume (j) `IRA` (k) `FE_Origin` (l) `NoSplit`

Fig. 4: Traffic Volume vs. AVG Accuracy per BS

Figure 4 where we report, for each considered solution, the heat maps of the average accuracy of the monitored flows for each BS during different times of the day (9 am, 2 pm, 10 pm) and, hence, different traffic volumes. Considering peak times of the day (i.e., 2 pm and 10 pm) we can observe how `FE_Origin` fails in serving several traffic-intensive areas. The requirement of placing FEs in the original BSs often hinders the placement of IMs and, hence, the monitoring of flows (blue holes in Figure 4g). The situation is better with `NoSplit (SotA)` that is more flexible in the placement of FEs within each cluster. In this way, the monitoring of flows is not hampered, however, requiring the FE and the IM of each flow to be placed in the same BS lows down the accuracy of the IMs. We can notice it in Figure 4h and Figure 4l where the distribution of accuracy is significantly more homogeneous with respect to the scattered situation of `FE_Origin`. Finally, higher and homogeneous accuracy is reached in `IRA` with the maximum level of flexibility for the placement of FEs and IMs, i.e., by allowing them to be placed in different BSs. This behaviour is further exacerbated in the peak time of 10 pm where the traffic volume is higher. Constraint (7) limits the average accuracy of `No_Split` although the number of monitored flows is slightly higher than the number of flows monitored by `IRA` (as shown in Figure 3b).

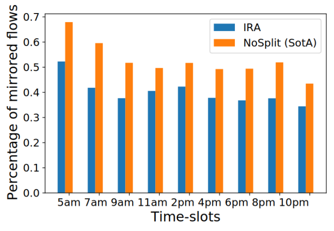
Takeaway. Decoupling the FE and IM in `IRA` increases flexibility, leading to better resource utilization and more evenly distributed accuracy in flow monitoring. Moreover, monitoring more flows does not always result in higher accuracy.

Limiting the available bandwidth. The previous results describe a case where the available bandwidth is not a bottleneck to the inference placement problem, i.e., the entire bandwidth of each link, generated as described above, is entirely dedicated to the placement of inference pipelines.

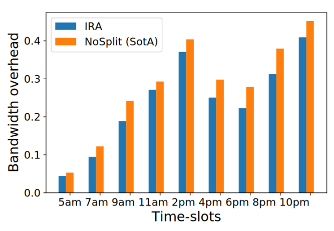
However, such an availability of bandwidth might not be always the most common case for two main reasons: i) the estimation of the available bandwidth can be noisy leading to errors in the residual bandwidth used as an upper bound of constraint (4), and, hence, in the following placement of the two components of each pipeline; ii) the existing traffic of other services across the network should be taken into account in the estimation of the residual available bandwidth. To consider this aspect in the experiments, we simulate a scenario where only 1% of the total bandwidth is available for mirroring the Netflix traffic to place inference pipelines. In this way, we consider a worst-case scenario equivalent to having a 99% error in the bandwidth availability estimation. This translates to considering 1% of B_n in constraint (4). Furthermore, this case also captures the worst-case scenario where just 1% of the total residual bandwidth is devoted to video traffic mirroring in the placement of inference pipelines. Finally, this ensures that bandwidth estimation errors do not cause mirrored traffic to interfere with the original traffic, thus preserving the video quality perceived by users.

Figure 5 presents the performance of the various solutions under these bandwidth restrictions. Note that `FE_Origin` is not included, as it does not involve any mirroring and, therefore, does not contribute to additional bandwidth usage. This is due to the constraint of placing the FEs on the original BSs where the video flows originate. Figure 5a illustrates the percentage of monitored flows that have been mirrored by each solution for the purpose of placing inference pipelines for monitoring. A *mirrored flow* is defined as a flow that has been replicated on a different link within the network for monitoring purposes. Both `IRA` and `NoSplit` solution mirror a significant portion of the original video traffic, although `IRA` performs less mirroring compared to the traditional solution. This behaviour is a consequence of the limited bandwidth resources available. Indeed, as shown in Figure 5b, the bandwidth consumption (bandwidth overhead) never exceeds 42% (achieved during the peak times, i.e., 2 pm and 10 pm, respectively) of the available bandwidth. However, this influences the percentage of monitored flows, as shown in Figure 5c, and subsequently affects the average accuracy achieved by each solution. Indeed, as depicted in Figure 5d, every solution achieves a lower average accuracy compared to what was attained in the scenario illustrated in Figure 3b. Note that, although this limitation, the performance loss in terms of accuracy, in the case of `IRA`, is only ~10% compared to the case shown in Figure 3b.

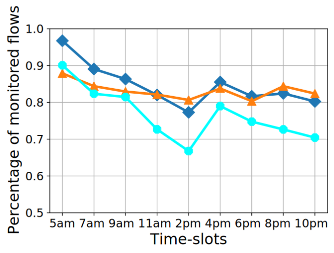
Takeaway. The `IRA` solution is robust in worst-case scenarios where just 1% of the total bandwidth is available for traffic mirroring, with only ~10% loss in average accuracy compared to the first case with complete availability of bandwidth, and it still outperforms the state-of-the-art solution. Furthermore, there is a trade-off between the accuracy and the percentage of mirrored flows for the placement of inference pipelines. However, this translates into less than 42% of the residual available bandwidth occupation.



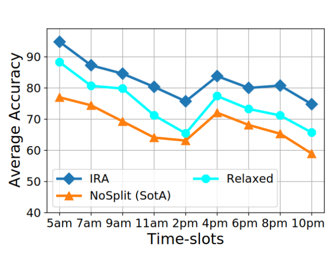
(a) % Mirrored flows



(b) Bandwidth overhead



(c) % Monitored flows

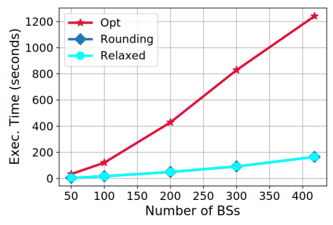


(d) Average Accuracy

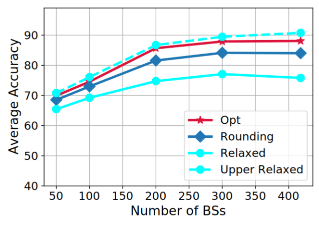
Fig. 5: Comparison vs. Baselines with restricted bandwidth for mirroring.

Optimality Evaluation. To study the trade-off between optimality efficiency we compare `IRA` with respect the optimal integer solution computed by the solver. To obtain such a solution the solver has been stopped after 60 minutes of computation and the best solution is considered. To show the trade-off between the scalability and the optimality of the proposed solution, the comparison exhibits the average execution time and the average accuracy as the number of BSs increases (increasing the number of flows as well). Figure 6 shows such a comparison when the time-slot is set to 10 pm. The dashed line in Figure 6b is the ceiling of the objective value achieved by solving the relaxation of IPP, hence representing a strict upper bound on the optimal value for IPP. From Figure 6a we immediately observe that the optimal solution results prohibitive for a practical utilization. The figure further shows that the overhead brought by the rounding procedure is negligible with respect to the time taken for solving the relaxed problem. This highlights that the complexity of `IRA` is dominated by the resolution of the linear program given by the relaxation of IPP. Finally, Figure 6b shows the proximity of `IRA` to the optimal solution and an improvement of ~17% of accuracy with respect to `Relaxed`. Such an improvement is due to the choice of variables to be rounded. Indeed, in case of fractional variables for the placement of IMs, `IRA` prioritizes the models and the nodes with the higher fractional values, leading to a certain improvement in terms of accuracy, in case of available resources. This result also suggests further insights for dynamic scenarios. As illustrated in Figure 6a, the proposed approach takes less than 200 seconds to compute a single placement of the monitoring pipelines with the maximum number of BSs. This implies that in a dynamic setting, the placement of monitoring pipelines can be re-computed every 5 or 10 minutes.

Takeaway. These observations confirm that the rounding procedure is an effective solution for the inference placement problem, both in terms of optimality and efficiency.



(a) Execution Time



(b) Average Accuracy

Fig. 6: Comparison vs. Integer Optimal solution

V. Related Work

Quality Inference of Video Streaming Traffic. Several studies have proposed using inference models to extract video quality metrics from encrypted traffic [4], [20]–[22]. These studies emphasize the necessity of injecting raw traffic into processing pipelines that consist of various components to perform traffic analysis. However, their primary focus is on the final step of these pipelines, specifically how to apply different models with varying levels of accuracy to estimate quality metrics of the traffic. In contrast, our work focuses on the placement of inference models together with the FE component to pre-process the incoming raw traffic. Our findings indicate that the arrangement of these monitoring pipelines over the network significantly impacts the average accuracy achieved in the estimation of video quality metrics.

Model Serving. The closest research line to our work is related to model-serving systems. In this context, a content provider receives a set of inference requests that need to be fulfilled. The main challenge is to place or schedule inference models across a given infrastructure while optimizing specific objectives, such as maximizing the total accuracy of the served models. Several inference-providing systems have been proposed in the literature, including Tensorflow Serving [23], Azure ML [24], and Clipper [25] where inference requests are served by a data center. Other studies have also explored inference provisioning within geographically distributed infrastructures, such as edge computing. [9]–[11], [13]. However, our work has significant differences from these approaches. In traffic monitoring, the placement of inference is driven by the necessity to monitor network flows to deduce video quality metrics, whereas in inference-serving systems, the requests are initiated by users. In the latter scenario, model placement directly affects user service quality. In contrast, the quality of the video streams each user receives in traffic monitoring is not influenced by inference placement. Moreover, these works consider just the deployment of ML models in the inference-serving process. In traffic monitoring, it is essential to consider the deployment of additional critical components, such as the FE for the traffic preprocessing, as this is fundamental for the effective functioning of the inference models. While the deployment of multi-component pipelines has been considered in video analytics [26], [27], and traffic analysis [28], the optimization problems formulated in these studies focus the placement and the variants of the inference models alone for each pipeline. They do not account for decisions related to other pipeline components. In our work, we take into account both the FE placement and the IM deployment (together with its variants) across a distributed edge infrastructure. In the cloud computing context, the deployment of inference pipelines has been considered in different works such as InferLine [29], Pretzel [30], and Nexus [31]. However, these works consider the deployment of inference pipelines on centralized infrastructures, e.g., data centers or GPUs' clusters. Our work, on the other hand, tackles the deployment problem over a distributed infrastructure such as a cellular network. The lack of a direct solution operating in the same context of this work justifies the choice of comparing `IRA` against baselines inspired by the main algorithmic ideas of the state of the art.

Content Placement Problem. From the optimization perspective, the IPP is close to the content placement problem for which a vast literature exists in different contexts of distributed systems, such as hybrid clouds or service caching [32]–[34]. However, there are two key differences between these problems. First, in the serving caching problem, each content can be placed or not, while in the IPP there is an additional decision, i.e., which variant must be chosen for each deployed model. Indeed, each model can have different variants to perform the same inference task, and different variants have different levels of accuracy and different resource demands. Second, in the content placement problem, there are no constraints between the placement of two different contents, while in our case the placement of the inference model should be coordinated with the placement of the feature extractor for each video flow. As shown in Section IV, such a coordination improves the average accuracy obtained by ISPs.

VI. Conclusion and Discussion

In this paper, we introduced the inference placement problem in cellular networks for monitoring video traffic flows. We first analyzed the computational complexity of the problem and developed a heuristic solution, `IRA`, based on rounding the solution of the relaxed formulation. Our evaluation on real offline traces demonstrates that decoupling the FE and IM yields higher accuracy in video flow monitoring compared to standard methods. Interestingly, we found that monitoring more flows does not always enhance average accuracy, suggesting that sampling subsets of flows can achieve target accuracy levels in predicting video quality metrics. Designing optimal sampling strategies for these subsets remains an area for future work. The solution presented here is an offline approach for inference placement. However, as discussed in [9], inference models can be dynamically updated to adapt to traffic changes due to performance degradation over time. Future work will focus on developing online strategies that can predict traffic volume changes and perform adaptive inference placement accordingly.

Acknowledgments

This work was done using data made available by Orange within the Netmob 2023 Challenge [15]. This work was partially supported by the ANR Project No ANR-21-CE94-0001-01 (MINT).

References

[1] Sandvine, "Phenomena—THE GLOBAL INTERNET PHENOMENA REPORT JANUARY 2023," https://shorturl.at/7TZi2, 2023.

[2] I. Akbari, M. A. Salahuddin, L. Ven, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, and S. Tuffin, "A look behind the curtain: traffic classification in an increasingly encrypted web," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 5, no. 1, 2021.

[3] A. Saverimoutou, B. Mathieu, and S. Vaton, "Which secure transport protocol for a reliable http/2-based web service: Tls or quic?" in *IEEE symposium on computers and communications (ISCC)*. IEEE, 2017.

[4] F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, 2019.

[5] F. Bronzino, P. Schmitt, S. Ayoubi, H. Kim, R. Teixeira, and N. Feamster, "Traffic refinery: Cost-aware data representation for machine learning on network traffic," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 5, no. 3, 2021.

[6] G. Wan, F. Gong, T. Barbette, and Z. Durumeric, "Retina: analyzing 100gbe traffic on commodity hardware," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022.

[7] J. Holland, P. Schmitt, N. Feamster, and P. Mittal, "New directions in automated traffic analysis," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[8] R. Behravesh, D. Breitgand, D. H. Lorenz, and D. Raz, "A practical near optimal deployment of service function chains in edge-to-cloud networks," *arXiv preprint arXiv:2401.07611*, 2024.

[9] F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, "{INFaaS}: Automated model-less inference serving," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021.

[10] T. S. Salem, G. Castellano, G. Neglia, F. Pianese, and A. Araldo, "Toward inference delivery networks: Distributing machine learning with optimality guarantees," *IEEE/ACM Transactions on Networking*, 2023.

[11] T. da Silva Barros, F. Giroire, R. Aparicio-Pardo, S. Perennes, and E. Natale, "Scheduling with fully compressible tasks: Application to deep learning inference with neural network compression," in *The 24th IEEE/ACM international Symposium on Cluster, Cloud and Internet Computing (CCGRID 2024))*, 2024.

[12] J. Martín-Pérez, L. Cominardi, C. J. Bernardos, and A. Mourad, "5gen: A tool to generate 5g infrastructure graphs," in *IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE, 2019.

[13] Y. Jin, L. Jiao, Z. Qian, S. Zhang, N. Chen, S. Lu, and X. Wang, "Provisioning edge inference as a service via online learning," in *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2020.

[14] S. Maheshwari, D. Raychaudhuri, I. Seskar, and F. Bronzino, "Scalability and performance evaluation of edge cloud systems for latency constrained applications," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018.

[15] O. E. Martínez-Durive, S. Mishra, C. Ziemlicki, S. Rubrichi, Z. Smoreda, and M. Fiore, "The netmob23 dataset: A high-resolution multi-region service-level mobile data traffic cartography," *arXiv preprint arXiv:2305.06933*, 2023.

[16] H. Kellerer, U. Pferschy, D. Pisinger, H. Kellerer, U. Pferschy, and D. Pisinger, *Multidimensional knapsack problems*. Springer, 2004.

[17] V. Cacchiani, M. Iori, A. Locatelli, and S. Martello, "Knapsack problems—an overview of recent advances. part ii: Multiple, multidimensional, and quadratic knapsack problems," *Computers & Operations Research*, vol. 143, 2022.

[18] Gurobi Optimization, LLC. [Online]. Available: http://www.gurobi.com

[19] ANFR, "Agence National des Frequences," https://data.anfr.fr/portail, 2025, [Online; accessed January-2025].

[20] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for https and quic," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018.

[21] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, "Buffest: Predicting buffer conditions and real-time requirements of http (s) adaptive streaming clients," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017.

[22] C. Gutterman, K. Guo, S. Arora, T. Gilliland, X. Wang, L. Wu, E. Katz-Bassett, and G. Zussman, "Requet: Real-time qoe metric detection for encrypted youtube traffic," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 2s, 2020.

[23] C. Olston, N. Fiedel, K. Gorovoy, J. Harmsen, L. Lao, F. Li, V. Rajashekhar, S. Ramesh, and J. Soyke, "Tensorflow-serving: Flexible, high-performance ml serving," *arXiv preprint arXiv:1712.06139*, 2017.

[24] D. Chappell, "Introducing azure machine learning," *A guide for technical professionals, sponsored by microsoft corporation*, 2015.

[25] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A {Low-Latency} online prediction serving system," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017.

[26] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose, "Videoedge: Processing camera streams using hierarchical clusters," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018.

[27] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and {Delay-Tolerance}," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017.

[28] G. Wan, S. Liu, F. Bronzino, N. Feamster, and Z. Durumeric, "Cato: End-to-end optimization of ml-based traffic analysis pipelines," *arXiv preprint arXiv:2402.06099*, 2024.

[29] D. Crankshaw, G.-E. Sela, X. Mo, C. Zumar, I. Stoica, J. Gonzalez, and A. Tumanov, "Inferline: latency-aware provisioning and scaling for prediction serving pipelines," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, 2020.

[30] Y. Lee, A. Scolari, B.-G. Chun, M. D. Santambrogio, M. Weimer, and M. Interlandi, "{PRETZEL}: Opening the black box of machine learning prediction serving systems," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018.

[31] H. Shen, L. Chen, Y. Jin, L. Zhao, B. Kong, M. Philipose, A. Krishnamurthy, and R. Sundaram, "Nexus: A gpu cluster engine for accelerating dnn-based video analysis," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019.

[32] X. Qiu, H. Li, C. Wu, Z. Li, and F. C. Lau, "Cost-minimizing dynamic migration of content distribution services into hybrid clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, 2014.

[33] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018.

[34] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, 2016.