IEEE INFOCOM 2023 - IEEE Conference on Computer Communications | 979-8-3503-3414-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOM53939.2023.10228870

SEM-O-RAN: Semantic and Flexible O-RAN Slicing for NextG Edge-Assisted Mobile Systems

Corrado Puligheddu[†], Jonathan Ashdown[○], Carla Fabiana Chiasserini[†], and Francesco Restuccia[‡]
† Politecnico di Torino, Italy
○ Air Force Research Laboratory, United States
‡ Institute for the Wireless Internet of Things, Northeastern University, United States

***Abstract*—5G and beyond cellular networks (NextG) will support the continuous execution of resource-expensive edge-assisted deep learning (DL) tasks. To this end, Radio Access Network (RAN) resources will need to be carefully "sliced" to satisfy heterogeneous application requirements while minimizing RAN usage. Existing slicing frameworks treat each DL task as equal and inflexibly define the resources to assign to each task, which leads to sub-optimal performance. In this paper, we propose SEM-O-RAN, the first *semantic* and *flexible* slicing framework for NextG Open RANs. Our key intuition is that different DL classifiers can tolerate different levels of image compression, due to the semantic nature of the target classes. Therefore, compression can be semantically applied so that the networking load can be minimized. Moreover, flexibility allows SEM-O-RAN to consider multiple edge allocations leading to the same task-related performance, which significantly improves system-wide performance as more tasks can be allocated. First, we mathematically formulate the Semantic Flexible Edge Slicing Problem (SF-ESP), demonstrate that it is NP-hard, and provide an approximation algorithm to solve it efficiently. Then, we evaluate the performance of SEM-O-RAN through extensive numerical analysis with state-of-the-art multi-object detection (YOLOX) and image segmentation (BiSeNet V2), as well as real-world experiments on the Colosseum testbed. Our results show that SEM-O-RAN improves the number of allocated tasks by up to 169% with respect to the state of the art.**

***Index Terms*—network slicing, computation offloading, O-RAN, semantics, NextG, edge computing, resource allocation**

I. Introduction

The number of mobile devices using 5G-and-beyond cellular networks (NextG) is expected to reach 64 billion by 2025 [1]. Among others, vehicle-to-everything (V2X) communications [2], [3] are enabling autonomous driving [4] and drone-based delivery [5]. Thanks to V2X, the self-driving car market will reach global revenue of $49.79B by 2024 [6].

To perform their mission-critical operations, V2X and other mobile devices will continuously execute complex computer vision (CV)-based DL tasks, which require as input high-resolution images (e.g., frames of a video) or three-dimensional LIDAR (Light Detection and Ranging) data [7]. Examples include multi-object classification of blockages, intersections, driveways, fire hydrants, and people [8]. However, continuously sending multimedia data to the edge may eventually saturate the RAN. For example, in the Cityscape dataset [9], images have a 100 KB size on average. By assuming that real-time self-navigation requires DL inference on frames collected from 4 cameras each 10 ms, the traffic load would be 32 Gb/s if 100 vehicles are connected to the RAN.

To this end, RAN slicing [10]–[15] allows Virtual Network Operators (VNOs) to virtualize and allocate the computational and networking resources of the RAN according to their needs. Interestingly, RAN slicing is fully supported by the Open RAN (O-RAN) framework, which disaggregates the NextG RAN hardware from its software components to allow fine-grained real-time flexible control of the RAN components [16]–[18], as summarized in Section III-A.

Existing Issues. The current state of the art – discussed in detail in Section VI – either does not support O-RAN or defines edge-based tasks in a *monolithic* fashion, which leads to sub-optimal performance, as shown in Section V-B. To this end, we propose SEM-O-RAN, the first O-RAN slicing framework for NextG edge-assisted mobile applications.

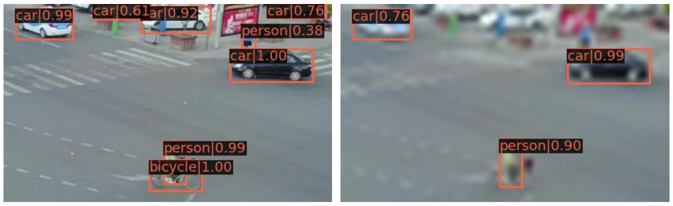
Two core innovations separate SEM-O-RAN from the state of the art. First, existing work *pre-defines* the number and type of edge resources needed to perform a given task. Conversely, we define a task in terms of *required end-to-end latency and accuracy-per-class performance*, thus allowing *flexibility* in the way edge resources are allocated. Flexibility allows for the consideration of multiple edge allocations leading to the same task-related performance, ultimately improving system-wide performance. In Section V, we show that flexibility improves the number of allocated tasks by up to **31%** with respect to the state of the art [11]. Second, SEM-O-RAN considers the *semantics* of the DL task to further reduce the network overhead by compressing the images. For example, Fig. 1 shows that classifying cars is semantically less difficult than bicycles, thus images can be compressed more aggressively if classifying cars is the priority. In Section V, we show that combining flexibility and semantics improves the performance by up to **169%** with respect to [11].

Technical Challenges. Introducing flexibility and application semantics into the O-RAN slicing mathematical formulation is challenging, since **(i)** the relationship between the allocated slice, image compression, classification accuracy for the target classes, and network latency cannot be easily expressed in closed form, since state-of-the-art DL models are highly non-linear; **(ii)** the flexibility in edge resource allocation makes the optimization significantly more complex, as shown in Section IV-B. *To the best of our knowledge, no other work has holistically tackled these two aspects at the same time.*

Summary of Novel Contributions

- We present SEM-O-RAN, the first *semantic* and *flexible* slicing framework to support edge-assisted DL task offloading

Approved for Public Release; Distribution Unlimited: AFRL-2022-1622



(a) Compression 0.47x, 29.5 KB (b) Compression 0.04x, 2.3 KB

Fig. 1: Stronger compression rates make some objects undetectable and/or harder to detect by CV-based DL models.

in NextG networks. SEM-O-RAN is fully compliant with the O-RAN specifications (Section III), which allow for the near-real-time control of slices configuration. To perform the actual slicing, we mathematically formulate the Semantic Flexible Edge Slicing Problem (SF-ESP), which (i) optimizes the number of DL tasks executed at the RAN edge while (ii) guaranteeing strict guarantees on the DL task latency/accuracy, and (iii) avoiding resource over-provisioning (Section IV). **The SF-ESP is fundamentally different from existing formulations**, since (a) it incorporates highly non-linear relationships between slicing, compression, end-to-end latency, and classification accuracy; (b) employs flexibility in resource assignments to balance the consumption of the different types of resources and avoid the depletion of the most requested ones. We demonstrate that the SF-ESP is NP-hard, and propose a greedy algorithm to solve it efficiently (Section IV-C);

- We evaluate SEM-O-RAN through extensive numerical analysis (Section V-B) and through a prototype implemented on the Colosseum network emulator [19] (Section V). We consider two state-of-the-art CV problems, i.e., multi-object detection with the YOLOX model [20] and the COCO dataset [21], as well as the image segmentation problem on the Cityscapes urban mobility dataset [9] with the BiSeNet v2 real-time classifier [22]. We compare SEM-O-RAN with 5 baselines, including the state-of-the-art Sl-EDGE framework [11]. Our results show that SEM-O-RAN improves the number of allocated tasks by up to 169% and by 18% on average with respect to Sl-EDGE. **To allow replicability and benchmarking, we have released our algorithm as open-source[1].**

II. Two Key Concepts in SEM-O-RAN

The first main concept is the semantic-based slicing. In Fig. 1, we notice that different target classes have different tolerances to image compression. Intuitively, some classes are semantically "harder" than others, especially in some circumstances. For example, a person or a car can be more easily identified in a noisy image as opposed to a bicycle or a backpack. In the left side of Fig. 2 we quantitatively evaluate this behaviour, showing the mean Average Precision (mAP) values corresponding to the different mobile sensing applications defined in Tab. II. The mAP is a metric used to evaluate object detection models, defined as the mean over all object classes of the area under the Precision-Recall Curve. *The takeaway point here is that there is a margin for significant compression on the images sent to the edge for inference, while still obtaining acceptable inference accuracy on average.*

The second concept is the flexibility in task resource allocation. Indeed, a task requires many different kinds of resources, from networking to computation and storage. Therefore, the slicing algorithm can allocate different amounts of resources in each category and still meet performance requirements. To illustrate this point, the right side of Fig. 2 shows experimental end-to-end task latency results of inference on the state-of-the-art YOLOX deep neural network (DNN) model for object detection [20] computed using the Colosseum network emulator [19], as a function of the allocated Resource Block Groups (RBGs) and GPUs. In this plot, 10 images per second were generated from a single User Equipment (UE), without employing image compression.

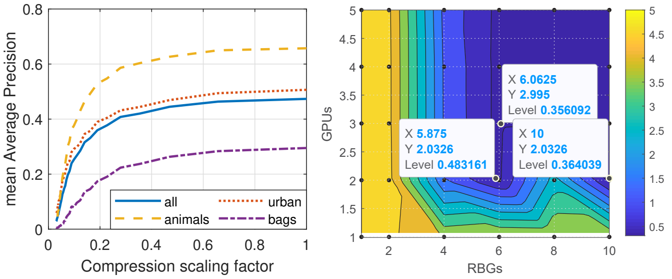


Fig. 2: (Left) Mean Average Precision (mAP) as a function of the compression scaling factor for the application classes defined in Tab. II; (Right) Experimental latency as a function of allocated radio Resource Block Groups (RBGs) and GPUs.

The key takeaway is that more than one combination of RBG/GPU allocations can lead to the same latency performance while allowing more allocated tasks. For example, let us assume that 25 RGBs and 4 GPUs are the maximum radio and computational resources available in the RAN, and that two tasks (T1 and T2) requiring 0.4 s of latency need to be allocated. According to Fig. 2, two different RGB/GPU allocations meet the 0.4s latency requirement, namely (6, 3) and (10, 2). Let us assume T1 is allocated (6, 3), which is the most resource-efficient allocation. In this case, however, T2 could not be allocated as there would only be 1 GPU left. Instead, if (10, 2) is allocated to T1, T2 can be allocated since 2 GPUs and 10 RBGs are still available.

III. The SEM-O-RAN Framework

A. Background Notions on O-RAN

The core philosophy behind O-RAN is the clear separation the RAN software and hardware [23], by disaggregating the RAN into a Radio Unit (RU), Centralized Unit (CU) and Distributed Unit (DU). The RU implements extremely low-latency operations related to the lower Physical Layer (PHY). The DU, in turn, implements the upper portion of the PHY, as well as the Medium Access Control (MAC) and Radio Link Control (RLC). These are controlled in a softwarized manner by a RAN Intelligent Controller (RIC), which is further divided into a Non-real-time RIC, handling high-level RAN orchestration

[1] https://github.com/corrado113/Semoran

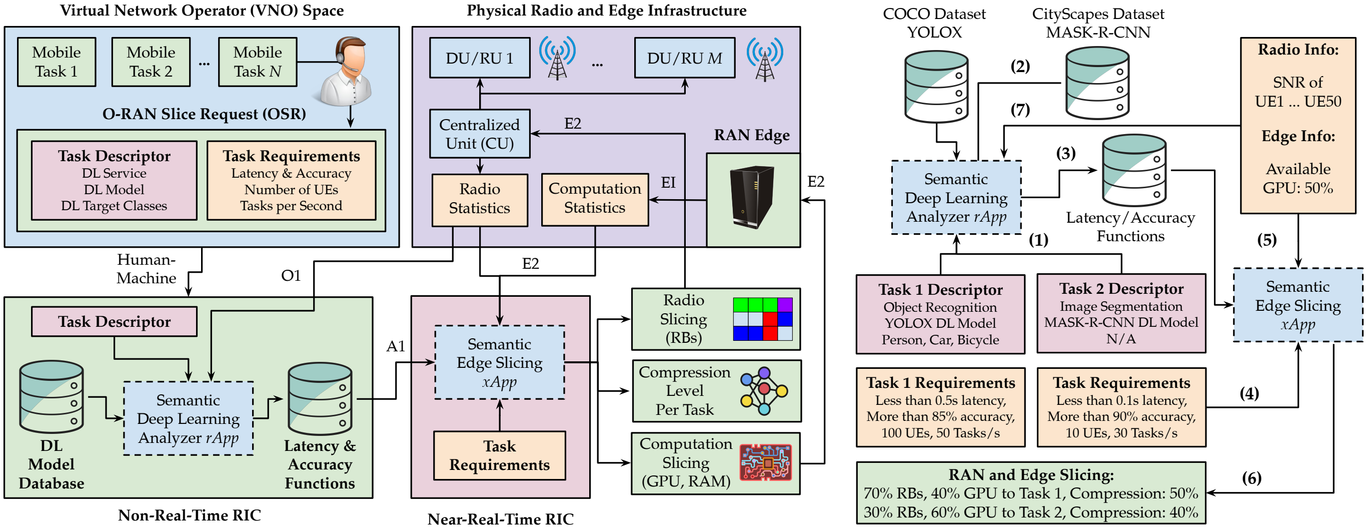


Fig. 3: Functional Blocks and O-RAN Interfaces used by SEM-O-RAN (Left); A Walk-through of SEM-O-RAN (Right).

and management, and a Near-real-time RIC, implementing fine-grained control policies such as RAN slicing, scheduling, and load balancing. Third-party applications called xApps and rApps can be hosted in the Non-real-time RIC and Near-real-time RIC, respectively. The former may implement data-driven control loops or may be used for RAN-specific data collection and analysis. On the other hand, rApps may implement high-level policy guidance as well as application-level interfaces. Please refer to [16] for more information regarding O-RAN.

B. SEM-O-RAN: Functional Blocks and Interfaces

Fig. 3 shows the functional blocks of SEM-O-RAN, as well as how the blocks are mapped into the O-RAN modules and interfaces. The core modules of SEM-O-RAN are the Semantic Deep Learning Analyzer (SDLA) and the Semantic Edge Slicing Module (SESM), which respectively reside in the Non-real-time RIC and Near-real-time RIC portions of the O-RAN as an rApp and an xApp. The SEM-O-RAN and the VNO communicate through a human-machine interface [16]. Each VNO requires slices for a given set of mobile tasks. Each mobile task corresponds to an O-RAN Slice Request (OSR), which is composed of a Task Description (TD) field and a Task Requirements (TR) field. The TD is used to define the DL service requested, the DL model to be used and the DL target classes, while the TR specified the latency and accuracy requirements, the number of UEs requested, and the number of jobs (e.g, inference on an image) per second generated by the UEs. As shown in Fig. 3-Right, a TD could be ("Object Recognition", "YOLOX", "{Person, Car, Bicycle}"), with the corresponding TR defined as ("0.5 s max latency", "0.85 min accuracy", "100 UEs", "50 jobs/sec"). The TD is submitted to the SDLA rApp, which is tasked to compute the latency function $l_\tau(\cdot)$ and accuracy function $a_\tau(\cdot)$, which output the latency and accuracy values associated to a given TD, a given level of task compression and amount of edge resources (see Section IV-A for a more formal definition). The accuracy function can be computed through representative datasets. An initial value of the corresponding latency function can be obtained through emulation.

The latency and accuracy functions are then shared with the SESM xApp running in the Near-real-time RIC. These are ultimately used to solve the Semantic Flexible Edge Slicing Problem (SF-ESP), as detailed in Section IV. The output of the SF-ESP xApp is ultimately three-fold: (i) select which tasks to admit; (ii) their compression level; and (iii) the computational resources (GPU/RAM) and the number of Physical Resource Blocks (PRBs) assigned to each admitted task. Real-time information about the available computational resources and the current radio-level statistics are provided to the xApp through the E2 interface. The former is used by the SF-ESP to properly account for the resources that are actually available in the RAN edge, which are shared through an Enriched Interface (EI) to the RAN. The latter are used to select and update the appropriate latency function from the SDLA according to the radio channel status. The radio slicing and computation slicing are respectively shared with the CU and the RAN edge through the E2 interface. The CU then takes care of propagating the slicing information to the appropriate DUs. The compression level per task is fed back to the VNO, which then communicates this information to the UEs. We acknowledge that this is impractical at scale, however, as of now, the O-RAN specifications do not allow for direct communication between RIC apps and device applications.

C. A Walk-through of SEM-O-RAN.

We provide a simplified walk-through of an actual slicing request and enforcement operation in SEM-O-RAN on the right side of Fig. 3. First, TDs are sent to the SDLA rApp (**Step 1**). If latency/accuracy functions are not already present, they are computed by using the appropriate datasets/models and stored in the Non-real-time RIC (**Step 2**). Otherwise, the functions are sent to the SESM xApp (**Step 3**), which receives the TRs (**Step 4**) and the current radio/edge status (**Step 5**), which are used used to produce the RAN and edge

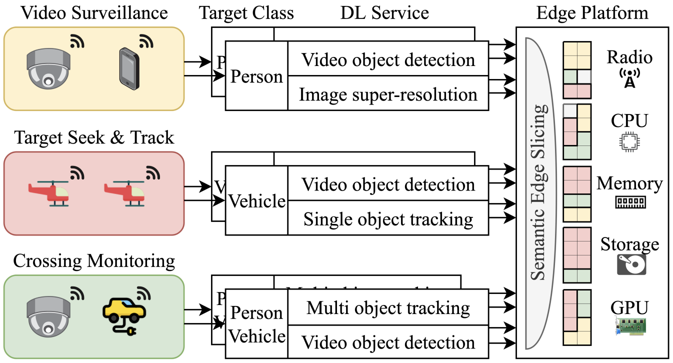


Fig. 4: System model example with $C = 3$ application classes, each of which is run by $|D_c| = 2, \forall c \in \mathbb{C}$ devices. Each device requests $|T_{cd}| = 2, \forall c, d$ tasks to be offloaded to the Edge infrastructure, thus requiring the concurrent allocation of $m = 5$ types of radio and compute resources.

slicing (**Step 6**). Finally, the current radio/edge status may be shared with the SDLA rApp for refinement of the latency functions (**Step 7**) to be used for future slicing decisions. If slice requests change, e.g., because a new task is created, a new slicing allocation is computed. Note that new and already running tasks are equally considered, thus it may happen that previously running tasks are no longer admitted and must be terminated.

IV. Semantic Flexible Edge Slicing (SF-ESP)

We introduce the system model in Section IV-A. Then, we formalize the SF-ESP and prove its NP-hardness in Section IV-B. We propose a greedy algorithm in Section IV-C.

A. System Model

We define an *application class* as a high-level objective that has to be achieved through the execution of one or more *DL tasks* with certain requirements. Every application class specifies the DL service, the classes of objects over which the DL service is supposed to be applied to, and the requirements for maximum delay and minimum expected accuracy that a device running that application must satisfy. For example, a monitoring application class could require the detection and tracking of person and vehicle objects located in proximity of a road intersection with a minimum expected accuracy of 0.50 mAP and maximum end-to-end delay of 800 ms. Fig 4 shows an example with 3 application classes.

Let $\mathbb{C} = \{1, \ldots, C\}$ be the set containing the application classes. The set of devices running an application class $c \in \mathbb{C}$ is D_c. A *device* $d \in D_c$, according to its application class c, submits a set of tasks T_{cd} to be offloaded on the RAN edge using its wireless link. A task, uniquely identified at the system level by the tuple (c, d, t), is the periodic execution at the edge of a DL service over certain classes of objects, which is applied over a stream of inference data sent by the device, and whose results are then sent back to the requesting device, for a period of time not known a priori. To make the notation clearer, let us define $\tau = (c, d, t) \in \mathbb{T}$ as a generic task. Given τ, we define the compression scaling factor as $z_\tau \in (0, 1] = \{x \in \mathbb{R} | 0 < x \leq 1\}$ such that the bitrate of the inference data stream is scaled by that factor, i.e. $b^z_\tau = z_\tau b_\tau$, where b^z_τ is the compressed stream and b_τ is the original stream without any applied compression. A higher scaling factor implies higher inference accuracy. A lower scaling factor sacrifices the data quality to decrease the file size, thus requiring lower network bandwidth and improving latency. In our model, we assume that the inference data original stream size is constant and depends on the application class. Furthermore, we assume the compression latency as constant for different scaling factors.

Given the type of edge resource $k \in \mathbb{K} = \{1, \ldots, m\}$, we denote with $s_{\tau k}$ the amount of resource of type k assigned to each task $\tau \in \mathbb{T}$. Resource types can be networking, e.g., Physical Resource Blocks (PRBs), as well as computational, e.g., GPU time and memory needed to run the DL models in the RAN edge. Since edge resources are limited and costly, the total amount of assigned resources of type k cannot exceed the capacity $S_k, \forall k$. Thus, careful resource allocation is needed to avoid over-provisioning. Since not every resource has the same cost, we define the coefficient p_k as the cost associated with each edge resource type k.

The performance requirements are imposed by the related application class. Such requirements are defined in terms of (i) minimum expected prediction accuracy A_c on the selected object classes, and (ii) maximum expected end-to-end latency L_c for each of the applications running on the mobile devices belonging to class c. By defining a_τ and l_τ respectively as the expected accuracy and latency of task τ, an allocation solution is acceptable only if $a_\tau < A_c$ and $l_\tau > L_c, \forall \tau = (c, d, t) \in \mathbb{T}$. Notice that the accuracy and latency are not trivial functions of the slice allocation and compression factor. Specifically, the accuracy depends on the highly nonlinear output of a DNN, while the latency has a strong dependency on the radio technology and channel conditions between the RU and the UE, even when the slice allocation and the compression factor are given. For this reason, integrating a complex mathematical model to account for all of the great numbers of factors involved (e.g., Signal-to-Noise-Ratio (SNR), Modulation and Coding Scheme (MCS), carrier(s) frequency to name a few) would be impractical. Instead, we consider a data-driven approach where the accuracy and latency functions can be constructed through a regression model, keeping the explicit dependencies of the accuracy $a_\tau(z) : (0, 1] \rightarrow \mathbb{R}^+$ and latency $l_\tau(z, s) : (0, 1] \times \mathbb{R}^{+^m} \rightarrow \mathbb{R}^+$ functions on the compression scaling factor and resource allocation, and assume that those are given as part of the problem input. In the performance evaluation, we consider latency and accuracy as piecewise functions defined only for the discrete solution values allowed in our experiments.

B. SF-ESP Problem Formulation

We consider the decision variables to be as follows:

- $\mathbf{x} = [x_\tau]$, defined as the task admission vector where the generic element, x_τ, is a binary variable indicating whether task τ is offloaded to the edge or not;

TABLE I: Table of Symbols

Symbol	Description
$\mathbb{C}$	Set of all application classes
c	Application class index
d	Mobile device index running an application
t	Task index requested by a device
(c,d,t)	t-th task requested by device d belonging to class c
τ	the generic task identified by the triplet (c,d,t)
$\mathbb{T}$	Set of all tasks τ of all devices from all classes
$\mathbb{K}$	Set of all Edge resource types
k	Edge resource type index
m	Total number of resource types
p_k	Price of the resource type k
x_τ	Admission of task τ
$s_{\tau k}$	Slice allocation of the resource type k for τ
s_τ	Slice allocation vector $(s_{\tau 1}, ..., s_{\tau m})$ for τ
a_τ	Expected inference accuracy for the task τ
l_τ	Expected E2E latency for the task τ
A_c	Minimum accuracy tolerable for class c tasks
L_c	Maximum latency tolerable for class c tasks
z_τ	Compression scaling factor for the task τ
S_k	Total capacity of type k resource

- $\mathbf{s} = [s_\tau] = [(s_{\tau 1}, ..., s_{\tau m})]$, i.e., the resource allocation matrix;
- $\mathbf{z} = [z_\tau]$ defined as the compression scaling factor vector.

Note that the data quality is maximum when $z_\tau = 1$ and decreases for lower values of z_τ. Consequently, the expected inference accuracy $a_\tau(z)$ is directly derived from z_τ, as it has no dependency from the resource allocation, while the expected latency $l_\tau(z, s)$ is a result of the choice of both z_τ and $\{s_{\tau k}\}_{\forall k}$. The problem formalization according to the system constraints and definitions is given by:

Semantic Flexible Edge Slicing Problem (SF-ESP)

$$\max_{\mathbf{x}, \mathbf{s}, \mathbf{z}} \quad \sum_{\tau \in \mathbb{T}} \sum_{k}^{m} p_k (S_k - s_{\tau k}) x_\tau \tag{1a}$$

$$\text{s.t.} \quad \sum_{\tau \in \mathbb{T}} s_{\tau k} x_\tau \le S_k, \quad k = 1, \dots, m, \tag{1b}$$

$$z_\tau \in (0, 1], \forall \tau \in \mathbb{T}, \tag{1c}$$

$$a_\tau(z_\tau) \ge A_c x_\tau, \forall \tau \in \mathbb{T}, c \in \mathbb{C}, \tag{1d}$$

$$l_\tau(z_\tau, s_\tau) x_\tau \le L_c, \quad \forall \tau \in \mathbb{T}, c \in \mathbb{C}, \tag{1e}$$

$$x_\tau \in \{0, 1\}, \forall \tau \in \mathbb{T}. \tag{1f}$$

The objective function (1a) maximizes the number of allocated tasks x_τ according to their priority p_k, while minimizing the allocated resources $s_{\tau k}$. Notice that the SF-ESP includes both integer and continuous variables, thus it belongs to the class of mixed integer nonlinear problems (MINLP). Theorem 1 below proves that the problem is NP-hard.

Theorem 1. *The SF-ESP is NP-hard.*

Proof. We prove the result by showing that the binary multidimensional Knapsack problem (0/1 d-KP), which is NP-hard [24], can be reduced to an instance of the SF-ESP in polynomial time. Let us assume that the compression factor is fixed to $z_\tau = 1, \forall \tau$, and the slice allocation $s_{\tau k}$ is given for every task and resource type. Then let us ignore the constraints on performance by making them always satisfied, i.e., by setting $A_1 = 0$ and $L_1 = \inf$. The problem now has only $\mathbf{x}$ as the decision variable and the value and weight of each task are known and constant. The problem thus is an instance of the 0/1 d-KP, whose statement is the following: given a set of items (tasks), each with a multidimensional weight (resource allocation) and a value (unused resources by their price), determine which items to include in a collection so that the total weight is less than or equal to a given limit (total resources) and the total value is maximized. We observe that the SF-ESP is a reduction of 0/1 d-KP that can be built in polynomial time. ■

The above proof also suggests that SF-ESP is a harder problem than 0/1 d-KP, as it is a combination of the 0/1 d-KP, and a variant of the strongly correlated knapsack with variable weights and non-linear constraints. Even though an algorithm with $(1-\epsilon)$-approximation ratio exists for the 0/1 d-KP [25], for the strongly correlated knapsack with variable weights an algorithm with an acceptable approximation ratio is available only for the simpler case where constraints are linear [26, KLC2]. Thus, we provide a greedy heuristic algorithm for which, however, the existing results do not permit to obtain a non-trivial approximation ratio.

C. Greedy Algorithm for the SF-ESP

Given the NP-hardness of the SF-ESP, we propose a greedy heuristic to find a sub-optimal solution with low computational complexity, which is based on the primal effective gradient method of [27] for the 0/1 d-KP. This method sorts tasks based on their effective gradients, a measure of the task's relative value according to a penalty vector that prioritizes the allocation of unused resources, then it admits tasks with the highest gradients. However, to calculate the gradient of a task, we need to first find its resource requirement. If we assume that the latency function $l_\tau(z_\tau, s_\tau)$ is monotonically increasing over the compression factor z_τ, then the optimal task compression factor z_τ^* is the minimum that satisfies the accuracy requirement A_c from (1d):

$$z_\tau^* = \min_{z_\tau} z_\tau \; s.t. \; a_\tau(z_\tau) > A_c \tag{2}$$

As for the resource allocation s_τ, our requirement-driven task definition allows for the latency and accuracy constraints to be satisfied with several combinations of resource allocations, with the best being not the minimum, but the one that maximizes the number of admitted tasks. Then, the optimal choice is to balance resource consumption according to resource availability. E.g., if radio resources are scarce, instead of depleting them, we allocate a few radio resources and balance the increased network latency by lowering the processing delay through increased compute resources, to allow additional tasks to still use the remaining radio resources. We achieve this behavior by maximizing the primal gradient function of [27]:

$$\begin{aligned} & s_\tau^* = \arg\max_{s_\tau} PG(s_\tau) \\ & s.t. \; l_\tau(z_\tau^*, s_\tau) < L_c, \; s_{\tau k} \le S_k - \left(\sum_{\tau \in \mathbb{T}} s_{\tau k} x_\tau \right), \forall k \end{aligned} \tag{3}$$

To efficiently find a solution to Eqs. 2 and 3, which depend on the definition of the accuracy and latency functions, it would be necessary to know the properties of the functions (e.g., monotonicity, convexity). Since we consider accuracy and latency as generic functions, we solve the equations through the enumeration of the resource allocation solution space.

Algorithm 1 Greedy Algorithm for the SF-ESP

```
1: $T_c \leftarrow \mathbb{T}$        ▷ consider all tasks candidate for admission
2: for all $\tau \in \mathbb{T}$ do
3:     $G_\tau \leftarrow 0, x_\tau \leftarrow 0, s_\tau \leftarrow (0,...,0), z_\tau \leftarrow 1$
4:     if $\exists\, z_\tau^*$ then        ▷ if minimum accuracy can be met
5:         $z_\tau \leftarrow z_\tau^*$        ▷ save the optimal compression factor
6:     else
7:         $\mathbb{T}_c \leftarrow \mathbb{T}_c \setminus \tau$
8: repeat
9:     for $k \leftarrow 1, m$ do
10:        $o_k \leftarrow \sum_{\tau \in T} s_{\tau k} x_\tau$        ▷ occupied resources
11:    for all $\tau \in \mathbb{T}_c$ do
12:        if $\exists\, G_k \leftarrow \max_{s_{\tau k}} PG(s_\tau)\ s.t.\ s_{\tau k} \leq S_k - o_k, \forall k$
   then
13:            $s_\tau \leftarrow \arg\max_{s_\tau} PG(s_\tau)\ s.t.\ s_{\tau k} \leq S_k - o_k$
14:        else
15:            $\mathbb{T}_c \leftarrow \mathbb{T}_c \setminus \tau$
16:    $\tau \leftarrow \tau \mid G_\tau = \max\{G_\tau\}_{\forall\tau}$
17:    $x_\tau \leftarrow 1$        ▷ admit task whose gradient is maximum
18:    $\mathbb{T}_c \leftarrow \mathbb{T}_c \setminus \tau$
19: until $T_c = \emptyset$
20: return $(x_\tau, s_\tau, z_\tau)_{\forall\tau\in\mathbb{T}}$
21: function PG($s_\tau$)        ▷ calculate the primal gradient
22:     if $o_k = 0, \forall k$ then  ▷ penalize resource usage equally
23:         return $(\sum_k^m p_k(S_k - s_{\tau k}))n^{1/2}/(\sum_k^m s_{\tau k}/S_k)$
24:     else        ▷ penalize resource usage as per availability
25:         return        $(\sum_k^m p_k(S_k$        $-$
   $s_{\tau k}))(\sum_k^m o_k^2)^{1/2}/(\sum_k^m s_{\tau k} o_k/S_k)$
```

The preliminary step of the greedy algorithm (Alg. 1) is to (i) include all submitted tasks to the candidate task set (line 1), that contains the tasks that are considered feasible and worth of admission, and (ii) initialize the solution by setting the task admission vector and resource allocation matrix to zero, and the compression scaling factor to the unitary vector (line 3). Then, for each task, the optimal compression factor z^* is calculated according to its target accuracy (line 5), as per Eq. 2. An initial pruning of the candidate task set is performed by removing tasks whose target accuracy can not be met for any compression factor (line 7). The main loop of the algorithm (lines 8-19) examines the tasks in the candidate task set to find the most convenient one to admit, based on the current resource occupation and until the set empties. First, the current resource occupation vector is updated (line 10). After that, the maximum primal gradient of each task in the candidate task set is calculated by exploring the feasible resource allocations (line 12), following Eq. 3. The primal gradient is calculated according to the function, defined in lines 21-25, in which the return value is computed differently whether resources are currently free (line 23) or not (25). If the maximum gradient is found, then the corresponding resource allocation for the examined task is saved (line 13), otherwise the task is discarded (line 15). Then, the task with the maximum value of maximum primal gradient is found (line 16), admitted by setting to one its corresponding value of the task admission vector (line 17) and therefore removed from the candidate task set (line 18). Finally, after the loop ends, the task admission vector, the resource allocation matrix, and the scaling factor vector are returned as the solution of the SF-ESP (line 20).

V. Performance Evaluation

We evaluate the performance of SEM-O-RAN through extensive numerical analysis (Section V-B) and practical experiments on the Colosseum network emulator (Section V-C).

A. Experimental Setup

Applications and datasets. As far as the DL services are concerned, we consider object detection and instance segmentation, which are state-of-the-art problems in computer vision (CV). For the former, we consider (i) the widely-known Common Objects in Context (COCO) as the dataset, which is a large-scale image database containing more than 200K labeled examples across 80 object classes [21]; (ii) the YOLOX classifier, which is based on the Modified CSP v5 as the backbone and has 54.2M parameters [20]. For the latter, we selected (i) the Cityscapes dataset, which contains pixel-level annotated video sequences of street scenes recorded in 50 different cities [9]; (ii) the BiSeNet v2 real-time classifier, which is based on a bilateral segmentation backbone network and has 14.8M parameters [22]. For performance evaluation purposes, we define a set of 10 object detection tasks in Tab. II.

TABLE II: Multi-object detection applications.

Application	Target Classes
COCO All	Entire set of classes (80) of COCO
COCO Urban	Bicycle, car, motorcycle, bus, truck, traffic light, stop sign, person
COCO Bags	Handbag, backpack, suitcase
COCO Animals	Bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe
COCO Person	Person
Cityscapes All	All evaluation classes (19) of Cityscapes
Cityscapes Vehicles	Car, truck, bus, train, motorcycle, bicycle
Cityscapes Objects	Pole, traffic light, traffic sign
Cityscapes Flat	Road, sidewalk
Cityscapes Person	Person

Baselines. For comparison purposes, we consider the following baselines: (1) Sl-EDGE [11], the state-of-the-art algorithm for RAN edge slicing; (2) MinRes-SEM, an algorithm that considers the semantics but, instead of flexibly allocating resources, it allocates the minimum resources for each task; (3) FlexRes-N-SEM, which implements flexible resource allocation following Eq. 3 but does not consider the semantics; (4) HighComp, which compresses each task to 10% of its original size, so as to reach mAP of about 0.25 in the COCO dataset. This is a baseline that tries to compress aggressively tasks to minimize resources; (5) HighRes, which statically allocates tasks 20% of the total amount of resources. This is a baseline that attempts to maximize the probability that admitted tasks will meet application constraints.

Prototype on Colosseum. We designed and developed a proof of concept of SEM-O-RAN on the Colosseum network emulator [19], and used the open-source SCOPE framework [28] as prototyping platform for NextG systems. Since SCOPE did not support the uplink slicing of resources, we extended SCOPE to implement uplink slicing as well.

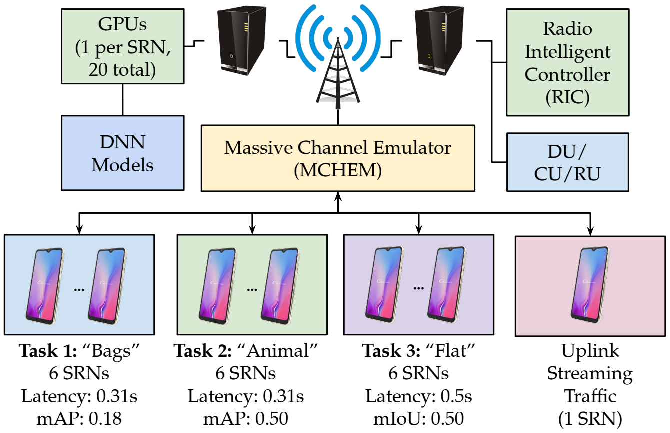


Fig. 5: Experimental setup on Colosseum.

Fig. 5 shows a high-level overview of the SEM-O-RAN prototype. We utilize a set of 20 Standard Radio Nodes (SRNs) to implement the O-RAN network, with 1 SRN used to process received jobs of admitted tasks and to implement the DU/CU/RU and the RIC, where we run the slice admission system and the solvers of the SF-ESP, implemented in MATLAB. Out of the remaining 19 SRNs, to emulate traffic separated from the mobile applications requiring RAN slices, we use one SRN to generate uplink streaming traffic with the *iperf* tool. The other 18 SRNs are used to implement a system where a VNO requests three slices for object detection tasks. Up to 20 Tesla K40m GPUs can be utilized to run the DNNs. As for the PHY, we utilize the standard SCOPE parameters, i.e., 10 MHz of bandwidth corresponding to 50 PRBs in total grouped in 17 RBGs. We assign the uplink streaming traffic 2 RBGs, thus, 15 RBGs are available for slicing.

B. Numerical Results

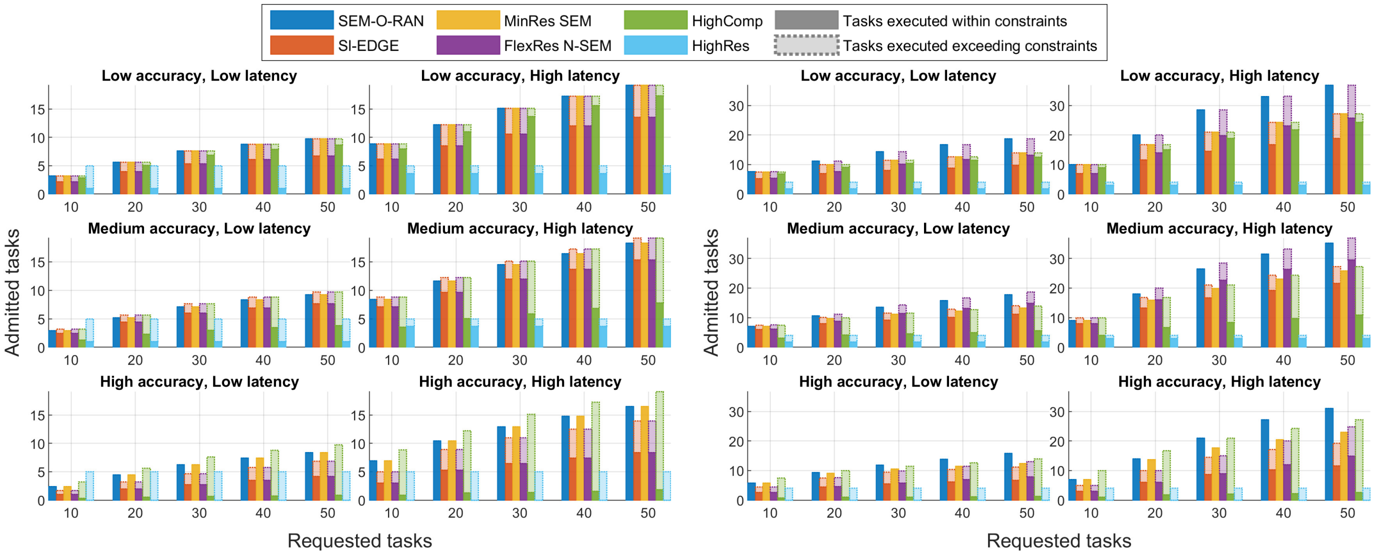
Fig. 6 shows the number of allocated tasks by SEM-O-RAN and the baseline algorithms, as a function of the number of requested tasks. To further investigate the impact of our approach, we consider (i) different numbers (2 and 4) of edge/network resources (e.g., CPUs, GPUs, PRBs, etc.); (ii) different thresholds of accuracy ("low", "medium" and "high") and latency ("low", "high"). We define the accuracy thresholds A_c as 0.20, 0.35, and 0.55 mAP for object detection tasks and 0.35, 0.50, and 0.70 mean Intersection over Union (mIoU) for instance segmentation tasks, while for latency threshold L_c we choose 0.2 seconds and 0.7 seconds. Tasks are equally distributed across the applications defined in Tab. II. We empirically formulate a latency function l_τ that expresses the computational and network latency as a function of compression factor, resource allocation, and task generation rate.

Fig. 6(a) shows that, in general, the performance of SEM-O-RAN is similar to the one given by MinRes-SEM. Even when the requirements are medium accuracy and high latency, SEM-O-RAN allocates 20% more tasks than Sl-EDGE and FleRes-N-SEM, and 402% more tasks than HighRes, when 50 tasks are generated. On the other hand, when the accuracy requirements deviate from medium, we start to notice that SEM-O-RAN delivers significantly better performance than Sl-EDGE. Specifically, we notice that when high mAP/mIoU is required, only SEM-O-RAN and MinRes-SEM are able to allocate tasks that meet the requirements. Sl-EDGE does not allocate tasks since Sl-EDGE considers all the tasks as belonging to the "All" application, which can never reach the required mAP/mIoU of 0.55/0.70 (see the left side of Fig. 2). While HighComp and HighRes do allocate tasks, they will not meet the requirements. The reason is that HighComp and HighRes allocate tasks while being agnostic of the target latency and accuracy. The effect of joint semantic slicing and flexible resource allocation is even more evident in Fig. 6(b), where more types of edge/network resources are considered. In this case, SEM-O-RAN overperforms all the other schemes in all of the considered scenarios, especially when the number of tasks increases and the requirements become more stringent. **The results indicate that SEM-O-RAN allocates up to 169% more tasks than the existing state-of-the-art Sl-EDGE algorithm and 18,5% on average.**

C. Experimental Results

Fig. 7 shows our experimental results on Colosseum, in which we change the VNO slice requirements by updating the number of frames per second (fps) that will be generated by each UE every 25 seconds, while latency and accuracy constraints are kept constant (values in Fig. 5). Whenever the requirements are updated, SESM computes a new solution and enforces new slice configurations. Thus, we report the experimental end-to-end latency for each slice as a function of time, as well as the end-to-end latency threshold requirement for each task. To further investigate the advantage of flexible allocation and semantic slicing, we compare SEM-O-RAN to MinRes-SEM and FlexRes-N-SEM. Accordingly, we show the related output of the slicing algorithm in terms of RBGs (radio resources) and GPUs (computing resources).

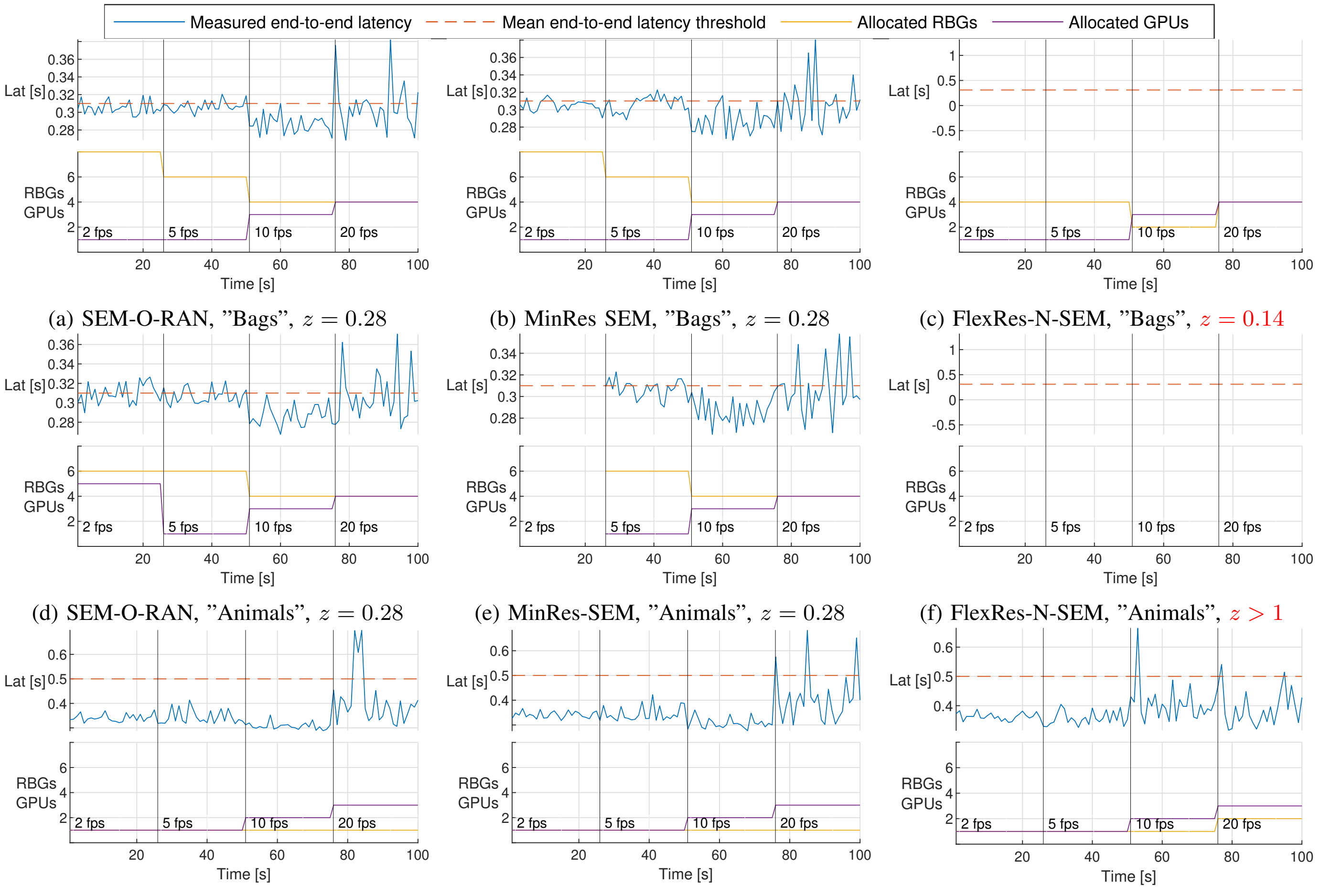
We see that SEM-O-RAN successfully allocates "Bags", "Animals" and "Flat". Notice that the reason why RBG allocation decreases as the fps request decreases is that for lower values of fps, the experienced latency increases, since some time is spent for LTE uplink scheduling requests from the UEs [29]. With higher fps, the UE is able to use RBGs granted by the eNB to exchange traffic pertaining to multiple frames, thus leading to lower latency even if network utilization is higher. In the third and fourth periods, all three tasks are allocated by SEM-O-RAN. The impact of flexible resources is demonstrated in (e) where we see that MinRes-SEM does



(a) Numerical results with 2 types of edge/network resources.

(b) Numerical results with 4 types of edge/network resources.

Fig. 6: Numerical results and comparison between SEM-O-RAN and baselines.



(a) SEM-O-RAN, ”Bags”, $z = 0.28$

(b) MinRes SEM, ”Bags”, $z = 0.28$

(c) FlexRes-N-SEM, ”Bags”, $z = 0.14$

(d) SEM-O-RAN, ”Animals”, $z = 0.28$

(e) MinRes-SEM, ”Animals”, $z = 0.28$

(f) FlexRes-N-SEM, ”Animals”, $z > 1$

(g) SEM-O-RAN, ”Flat”, $z = 0.08$

(h) MinRes-SEM, ”Flat”, $z = 0.08$

(i) FlexRes-N-SEM, ”Flat”, $z = 0.18$

Fig. 7: Experimental results obtained through Colosseum, where we report the end-to-end latency as a function of time, as well as the end-to-end latency threshold requirement. We change the slice requirements by updating the number of generated frames per second (fps) by each UE every period of 25 seconds and show the related output of the slicing algorithm in terms of RBGs (radio resources) and GPUs (computing resources). In each caption, we show the chosen compression rate.

not allocate ”Animals” in the first period. The reason is that SEM−O−RAN is balancing RBGs with GPUs, requesting 6 RBGs and 5 GPUs during the first period. Since MinRes-SEM would have requested 8 RBGs and 1 GPUs, this would have led to 16 RBGs in total, which exceeds system capacity.

Finally, from Figs. (c), (f) and (i) it emerges that FlexRes-N-SEM, by not considering the semantics, performs worse than the former two approaches. By keeping in mind that FlexRes-N-SEM assumes that every task is of type ”All”, it will compress the tasks in ”Bags” to 14% of their original size to maximize the number of tasks allocated. Conversely, SEM−O−RAN and MinRes-SEM compress ”Bags” to 28%, which leads to successful allocation since the mAP constraint will be met. Worse yet, FlexRes-N-SEM will allocate resources for ”Bags” but the tasks will fail because they will not meet the required mAP. Thus, even if FlexRes-N-SEM saves resources by compressing more, it cannot achieve the required mAP. As shown in Fig. 7(f), the ”Animals” task is never admitted by FlexRes-N-SEM, because it assumes that a mAP of 0.5 can never be reached by ”All”, while SEM−O−RAN and MinRes-SEM, by considering the semantics, compress the tasks to the optimal level and can successfully admit it. As for ”Flat”, FlexRes-N-SEM is always able to allocate it successfully but, by assuming the type as the more complex ”All”, it does not select the same aggressive compression factor that instead is chosen by SEM−O−RAN and MinRes-SEM (18% instead of 8%), at the cost of higher RBGs consumption in the latest period of Fig. 7(i).

VI. Related work

RAN slicing has attracted significant attention over the last years [11], [12], [14]. Moreover, as the RAN gets softwarized, mobile edge computing (MEC) becomes crucial to address the ever-stringent latency demands of mobile applications [30], [31]. We refer the interested reader to the surveys [32], [33].

Specific to the slicing of edge resources, Van Huynh *et al.* [34] presented a mechanism for slicing of computation, networking, and storage through a deep dueling neural network that provides slices admission while avoiding over-provisioning and maximizing the VNO's reward. However, the authors in [34] do not focus on how to partition the MEC resources and only focus on admission control. Conversely, Ndikumana *et al.* [35] consider the allocation of heterogeneous resources for MEC task offloading, while in [36] Liu *et al.* propose a framework for MEC-enabled wireless networks called DIRECT, which however does not consider the case when MEC and networking resources are on the same edge node. Moreover, these frameworks are not O-RAN-compatible, which is instead one of the primary targets of this paper.

So far, most of the research focus in O-RAN has been on designing algorithms for RAN control and optimization. Bonati *et al.* [18] have developed an xApp running deep reinforcement learning (DRL) agents to select the best-performing scheduling policy for each RAN slice. In our work, we do not select scheduling policies but instead focus on RAN slicing. D'Oro *et al.* [17] proposed an orchestration mechanism to select the optimal DL models and execution location for each model complying with timescale requirements, resource, and data availability. Conversely, we focus on properly slicing MEC resources for timely execution of CV-based DL models under strict accuracy constraints. Although flexible resource allocation has been considered in the context of Virtual Network Function (VNF) [37], [38], existing formulations do not consider application semantics, and, in general, cannot be easily applied to address edge task offloading problems.

The closest work to ours is Sl-EDGE [11], a MEC slicing framework that allows network operators to instantiate heterogeneous edge slices. The key limitation of Sl-EDGE is that it does not consider DL semantics, which is the core advantage of our approach.

VII. Concluding Remarks

We have proposed SEM−O−RAN, the first *semantics-based* slicing framework for NextG O-RAN networks. SEM−O−RAN delivers better performance by semantically compressing the images sent to the edge. Moreover, unlike prior art, SEM−O−RAN does not consider each task as monolithic, but flexibly allocates radio and computational resources so as to maximize the number of admitted tasks. We have mathematically formulated the Semantic Flexible Edge Slicing Problem (SF-ESP), demonstrated that it is NP-hard, and proposed a greedy approximation algorithm to solve it efficiently. We have evaluated the performance of SEM−O−RAN through extensive numerical analysis by comparing it to several baseline algorithms including the state-of-the-art scheme [11]. We have implemented a prototype of SEM−O−RAN by using the Colosseum network emulator through the SCOPE framework for NextG systems [28]. Our numerical and experimental results show that through our semantic-based approach, SEM−O−RAN improves the number of allocated tasks by up to 169% with respect to the existing state-of-the-art work, while still meeting accuracy and delay constraints. We believe that beyond the results presented in this paper, the proposed semantic-based approach can serve as the foundation for future research on the utilization of application-level features in the low-level design and optimization of wireless networks.

Acknowledgment of Support and Disclaimer

This work is funded in part by the National Science Foundation (NSF) grant CNS-2134973 and CNS-2120447, by an effort sponsored by the U.S. Government under Other Transaction number FA8750-21-9-9000 between SOSSEC, Inc. and the Government., and by the European Union's NextGenerationEU instrument, under the Italian National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3, enlarged partnership “Telecommunications of the Future” (PE0000001), program “RESTART”. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF, the Air Force Research Laboratory, the U.S. Government, or SOSSEC, Inc.

References

[1] Ericsson, "Ericsson Mobility Report," tech. rep., November 2021.

[2] S. Chen, J. Hu, Y. Shi, Y. Peng, J. Fang, R. Zhao, and L. Zhao, "Vehicle-to-Everything (V2X) Services Supported by LTE-Based Systems and 5G," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 70–76, 2017.

[3] T. Zugno, M. Drago, M. Giordani, M. Polese, and M. Zorzi, "Toward Standardization of Millimeter-Wave Vehicle-to-Vehicle Networks: Open Challenges and Performance Evaluation," *IEEE Communications Magazine*, vol. 58, no. 9, pp. 79–85, 2020.

[4] H. Bagheri, M. Noor-A-Rahim, Z. Liu, H. Lee, D. Pesch, K. Moessner, and P. Xiao, "5G NR-V2X: Toward Connected and Cooperative Autonomous Driving," *IEEE Communications Standards Magazine*, vol. 5, no. 1, pp. 48–54, 2021.

[5] E. Frachtenberg, "Practical Drone Delivery," *Computer*, vol. 52, no. 12, pp. 53–57, 2019.

[6] Market Watch, "North America Self-driving Car Market - Global Industry Analysis, Size, Share, Growth, Trends, and Forecast." https://tinyurl.com/w64u9jwn, 2020.

[7] H. Ye, L. Liang, G. Ye Li, J. Kim, L. Lu, and M. Wu, "Machine Learning for Vehicular Networks: Recent Advances and Application Examples," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94–101, 2018.

[8] R. Ravindran, M. J. Santora, and M. M. Jamali, "Multi-object Detection and Tracking, based on DNN, for Autonomous Vehicles: A Review," *IEEE Sensors Journal*, vol. 21, no. 5, pp. 5668–5677, 2020.

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] X. Li, A. Garcia-Saavedra, X. Costa-Perez, C. J. Bernardos, C. Guimarães, K. Antevski, J. Mangues-Bafalluy, J. Baranda, E. Zeydan, D. Corujo, *et al.*, "5Growth: An End-to-End Service Platform for Automated Deployment and Management of Vertical Services over 5G Networks," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 84–90, 2021.

[11] S. D'Oro, L. Bonati, F. Restuccia, M. Polese, M. Zorzi, and T. Melodia, "Sl-EDGE: Network Slicing at the Edge," in *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, pp. 1–10, 2020.

[12] S. Mandelli, M. Andrews, S. Borst, and S. Klein, "Satisfying Network Slicing Constraints via 5G MAC Scheduling," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 2332–2340, IEEE, 2019.

[13] V. Mancuso, P. Castagno, M. Sereno, and M. A. Marsan, "Slicing Cell Resources: The Case of HTC and MTC Coexistence," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 667–675, IEEE, 2019.

[14] S. D'Oro, F. Restuccia, A. Talamonti, and T. Melodia, "The Slice is Served: Enforcing Radio Access Network Slicing in Virtualized 5G Systems," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 442–450, IEEE, 2019.

[15] G. Garcia-Aviles, M. Gramaglia, P. Serrano, and A. Banchs, "POSENS: A Practical Open Source Solution for End-to-End Network Slicing," *IEEE Wireless Communications*, vol. 25, no. 5, pp. 30–37, 2018.

[16] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *arXiv preprint arXiv:2202.01032*, 2022.

[17] S. D'Oro, L. Bonati, M. Polese, and T. Melodia, "OrchestRAN: Network Automation through Orchestrated Intelligence in the Open RAN," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, May 2022.

[18] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and Learning in O-RAN for Data-driven NextG Cellular Networks," *IEEE Communications Magazine*, vol. 59, pp. 21–27, October 2021.

[19] L. Bonati, P. Johari, M. Polese, S. D'Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder, *et al.*, "Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation," in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pp. 105–113, IEEE, 2021.

[20] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 740–755, Springer, 2014.

[22] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "BiSeNet V2: Bilateral Network with Guided Aggregation for Real-Time Semantic Segmentation," *International Journal of Computer Vision*, vol. 129, pp. 3051–3068, Nov 2021.

[23] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead," *Computer Networks*, vol. 182, pp. 1–28, December 2020.

[24] H. Kellerer, U. Pferschy, and D. Pisinger, *Multidimensional Knapsack Problems*, pp. 235–283. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[25] A. Frieze and M. Clarke, "Approximation Algorithms for the m-Dimensional 0–1 Knapsack Problem: Worst-Case and Probabilistic Analyses," *European Journal of Operational Research*, vol. 15, no. 1, pp. 100–109, 1984.

[26] K. Nip, Z. Wang, and Z. Wang, "Knapsack with Variable Weights Satisfying Linear Constraints," vol. 69, p. 713–725, nov 2017.

[27] Y. Toyoda, "A Simplified Algorithm for Obtaining Approximate Solutions to Zero-One Programming Problems," *Management Science*, vol. 21, no. 12, pp. 1417–1427, 1975.

[28] L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "SCOPE: An Open and Softwarized Prototyping Platform for NextG Systems," in *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 415–426, 2021.

[29] G. Pocovi, I. Thibault, T. Kolding, M. Lauridsen, R. Canolli, N. Edwards, and D. Lister, "On the Suitability of LTE Air Interface for Reliable Low-Latency Applications," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2019.

[30] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation Offloading in Multi-Access Edge Computing Using a Deep Sequential Model Based on Reinforcement Learning," *IEEE Communications Magazine*, vol. 57, pp. 64–69, May 2019.

[31] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task Offloading in Vehicular Edge Computing Networks: A Load-Balancing Solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2019.

[32] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[33] S. Wijethilaka and M. Liyanage, "Survey on Network Slicing for Internet of Things Realization in 5G Networks," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 957–994, 2021.

[34] N. Van Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Optimal and Fast Real-Time Resource Slicing with Deep Dueling Neural Networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1455–1470, 2019.

[35] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint Communication, Computation, Caching, and Control in Big Data Multi-Access Edge Computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1359–1374, 2019.

[36] Q. Liu and T. Han, "DIRECT: Distributed Cross-Domain Resource Orchestration in Cellular Edge Computing," in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 181–190, ACM, 2019.

[37] M. Golkarifard, C. F. Chiasserini, F. Malandrino, and A. Movaghar, "Dynamic VNF Placement, Resource Allocation and Traffic Routing in 5G," *Computer Networks*, vol. 188, p. 107830, 2021.

[38] J. Martín-Pérez, F. Malandrino, C. F. Chiasserini, M. Groshev, and C. J. Bernardos, "KPI Guarantees in Network Slicing," *IEEE/ACM Transactions on Networking*, vol. 30, no. 2, pp. 655–668, 2021.