Optimal Service Decomposition for Mobile Augmented Reality with Edge Cloud Support

Zhaohui Huang, Vasilis Friderikos

[a]*Department of Engineering, King's College London, London, WC2R 2LS, UK*

Abstract

Mobile augmented reality (MAR) applications are starting to attract significant attention due to the enhanced capabilities stemming from both the network and the end devices that propel their realization. However, despite the progress on the end user devices, MAR applications are inherently hugely demanding in terms of computational and memory requirements since they combine, inter alia, video streams, computer generated images, intense computer vision algorithms and geolocation. To this end, edge cloud computing is envisioned as a key technology for supporting such applications where part of the computationally demanding algorithms could be offloaded to suitably selected edge clouds. Within that context the inherent user mobility should be considered to allow an efficient service continuum between edge and the end-terminal. To this end, in this paper, an optimal edge cloud resource MAR service decomposition is presented that takes explicitly into account the AR service composition as well as the inherent user mobility to proactively allocate resources to satisfy the required strict latency and frame accuracy requirements of MAR applications. In addition to the optimal decision making using mathematical programming, and as a mean to provide real-time decision making two advanced heuristic techniques are proposed. A Simulated Annealing based mobility aware AR algorithm (SAMAR) is developed to enhance computing efficiency and a Long Short-Term Memory (LSTM) neural network which is trained offline with optimal solutions. Numerical investigations reveal that significant gains can be achieved by the proposed schemes compare to a number of baseline previously proposed techniques.

Keywords: 5G, Augmented reality, Mobility, Edge Cloud Computing, Resource Management, Services and Applications.

1. Introduction

The utilization of mobile augmented reality (MAR) applications is expected to further increase by the proliferation of enhanced mobile devices and enhanced capabilities offered by the current deployment of 5G networks. Enabled by powerful end devices and emerging network capabilities augmented reality applications have recently started to attract significant attention as they allow new type of applications since they can amalgamate the cyber with the real world by presenting artificial perceptual information generated by computers to augment the physical real world environment. Whilst it is already widely accepted in different industries (entertainment, advertisement and manufacturing) and on different devices (i.e. smart glasses) it is well expected that the next frontier of AR applications would be on mobile networks [1]. To enable lighter, highly mobile and more technical advanced AR technologies and devices,we are facing a series of challenges including platform diversity, computational efficiency and caching limitations [2][3]. According to [3], a significant volume of memory and CPU resources size is required when rendering 3D objects which naturally emphasizes the insufficiency of computing and caching resource when depending solely on the mobile terminal capabilities.

Mobile edge cloud (MEC) architectures can better support MAR applications because it is possible to anchor computational and memory intensive functions of the service at computing-efficient nodes at the edge of the network in close proximity to mobile devices [2][3][4]. Support of MAR applications under a nominal MEC architecture has been considered in [4][5], where architectural details of a tier-architecture is provided. Despite the gains that can be attained by offloading such complex tasks to edge clouds (ECs) attention should be placed on the required volume of data transmission that need to take place between end devices and the edge cloud. In addition, during congestion episodes

parts of the network might be overloaded resulting in communication delays between the end terminal and edge clouds which might affect the service quality [5]. Therefore, MAR system's performance in response time can be increased through enhancing transmission efficiency and optimizing the edge cloud servers allocation and utilization [3][4][5][6]. For example, during congestion episodes reducing the uploaded video frame size might ease the burden of data transmission and computation supported by the MEC. Hence, frame selection and target area/object selection could be used to reduce the amount of total uploaded data [4][7]. However, the analytic accuracy of frames deteriorates with a smaller uploaded size. To this end, it is necessary to achieve, inter alia, a balance between frame accuracy and service latency.

In the sequel, the different functions in which a typical MAR application can be decomposed are explained. It is worth pointing out from the outset that the MAR functions can serve as a network function virtualization (NFV) function chain. NFV service chaining is a well known network architecture paradigm and focuses on decomposed functions' placement, load balancing and availability [8][9]. It enables flexible and economical implementation of applications in 5G networks and beyond [10][9]. In this work, the decomposed MAR functions are categorized into two types: computational intensive functions that consumes mainly computing resources and storage intensive functions that requires significant amount of local cache memory. Firstly, the terminal is responsible for selecting the video frames that have the potential to be augmented with AR objects (ARO). Then, after a pre-processing those video frames are transmitted to an edge cloud for Object Detection and Feature Extraction. Subsequently, the original image of objects can be recognized from extracted features and according to auxiliary information from the uploaded frames (i.e. point of view), the Object Matching function will provide validation and selection of proper AROs in required gesture [11][12]. Different MAR system architectures enable different modules to execute these functions with other supporting ones in different layers (1) [5][6][11]. However, the general working process of the decomposed functions in a MAR service can be, in general, summarized into an overall decomposed flow of functions as presented in Figure (1).

It is worth pointing out that it is not efficient to deploy all the above mentioned functions at the mobile device due to their limited computational resources [13][14][15]. The above MAR functions can be further decomposed into computational and storage intensive

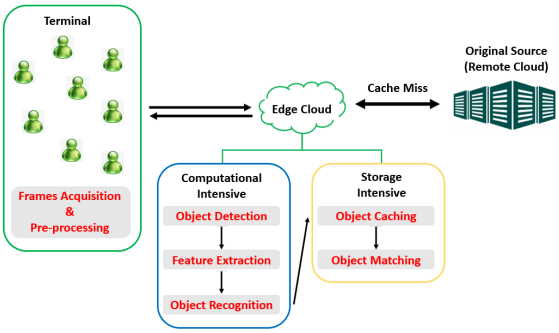

Figure 1: An example of EC-supported working flow of MAR applications including 2 types of MAR functions.

ones, in essence related to those that consume significant CPU resources with the database for ARO matching and those that consume significant cache memory resources. Such decomposition of the underlying MAR functions leads to a more flexible arrangement when some servers have sufficient computing resources with low availability on cache resources or vice versa. Optimization of these functions was the focus by some previous research to efficiently transmit AR tasks to the edge cloud but the mobility effect has not been explicitly taken into account as well as the above classification of the different AR functions [4][12] [16].

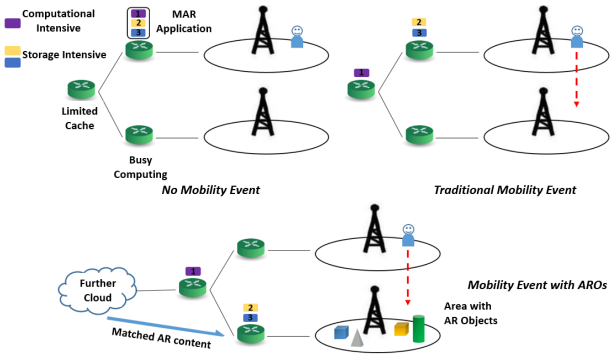

Figure 2: Illustrative toy examples on the effect of mobility on pro active resource allocation of MAR functions. Differences between applying a complete MAR application and its decomposed functions in a no mobility event, a nominal mobility event and a mobility event with AROs

As an illustration, several examples of an AR application applying service decomposition supported by ECs is presented by Figure (2). It reveals the mobility effect on the communication between the terminal and the target EC. According to this figure, there could be a detrimental effect in terms of the overall latency when changing the point of attachment during the lifetime of

a MAR session. The reason is that the new point of attachment is not an optimized choice in terms of the EC support for such a latency sensitive application. Compared to other applications, which might be tolerant to delay, the mobility effect in MAR services should be carefully considered. When facing both AROs and the user mobility, decomposition and proactive resource allocation can be combined to achieve an optimized decision making.

Motivated by the above aspects, the key novelty is that in this work we explicitly consider the user mobility and formulate a multi-objective optimization problem aiming to balance the overall delay, video frame quality and caching of augmented reality objects in the network faced by the MAR application that requires offloading task to an EC. To the best our our knowledge no previous research work considered explicitly the user mobility and service decomposition in order to optimize the service delivery and allocation of network resources. We extend hereafter our previous work [17] to construct the proposed optimal framework (via mathematical programming) including the development of a simulated annealing based scheme and a long short term memory (LSTM) neural network based scheme to provide competitive solutions, whilst being amenable to real-time decision making. Through a wide set of simulations, the performance of the above proposed schemes are compared with several baseline schemes stemming from closely related research. Advantages of the proposed schemes are discussed in analysis and evaluation section.

2. Related work

The proposed scheme in this paper provides, in essence, an EC-supported MAR system whereas MAR service decomposition and user mobility is explicitly taken into account whilst balancing augmented reality objects caching, video frame length and accuracy. In that frontier, there has been various recent research efforts to improve overall MAR systems. The work in [18] proposes a trade-off model between workload and service latency. It selects from candidate locations to place ECs and then focus on the users' allocation issue. While in this paper, our proposed scheme could find optimal solutions for both proactively caching and allocation decisions under pre-given network conditions. The research in [16] proposes a dynamic adaptive protocol over the edge for an MAR system. It also only focuses on the resource allocation and enables the AR reconfiguration by users to adapt to variations in wireless channels and computation workload. However, our proposed scheme reacts to changes through finding another suitable EC instead of changing configuration and causes potential interference to other requests. In a similar setting, the work in [7] considers a cloud-based architecture to improve the performance of MAR applications in development, deployment and maintenance. The above mentioned work considered the MAR system as a whole, however some other relevant works considered specific aspect or a particular functionality of a MAR application. The work in [19] improves object detection, while the work in [20] improves object recognition. On the other hand, Wu et al. focus on finding a better local cache management strategy [12]. Furthermore, Li et al. reduce the amount of initially loaded model data to minimize computational pressure [3]. Among existing works, mobility of AR applications is mentioned but its effect on service latency is not considered explicitly as is the case in this work.

In a closely related work, Liu et al. focus on the relationship between latency and frame resolution in MAR systems and edge clouds [4]. They provide an edge network orchestrator for EC-based MAR systems that considers the trade-off between service latency and analytic accuracy [4]. This is similar to the proposed optimization trade-off balancing model for EC-based MAR systems, however there are some significant differences. The most important one is the inherent user mobility. In [4], user's mobility is not taken into account which means that the model considers only the path between user's initial location to assigned edge cloud. However, in this work, mobility is explicitly taken into account which relates to the path between candidate user locations and different edge clouds. Furthermore, another key difference is that in the so-called FACT scheme proposed in [4], the MAR service is regarded as a whole while in this work it is decomposed into two types of functions, i.e., computational and memory intensive. This allows us to have a more flexible assignment, compared to other previous monolithic solutions, according to cache size and CPU frequency. In addition, the work in [4] considers a convex type of function between frame length and computational complexity which is only affected by the frame length. However, this work takes into account differences between EC CPU frequencies, i.e., considers heterogeneous edge clouds. Finally, the authors in [4] considered three main constraints: minimum accuracy requirement, once service per request and range of the decision variable. However, in this paper a more realistic scenario is considered where additional constraints such the available capacity of the edge clouds, the cache size and the cache miss penalty are explicitly taken into account. In [21], au-

thors also consider the cache hit/miss at ECs similarly to this work. However, the emphasis is into the caching and power consumption of the mobile AR devices and the associated mobile cache management. Hence, a different trade-off model has been considered with focus on balancing between energy consumption of the MAR device and latency in terms of caching size. In addition, service decomposition for the MAR application is applied in [22]. Their proposed framework also achieves efficient computing and offloading through dynamically reconfiguration of video and server. They also take processing and transmission delay into account similarly to this work. However, when considering the quality, they consider the user expected latency and user perceived video quality. Thus, we consider the FACT algorithm in [4] as a typical example from these related works to compare our proposed framework. A more detailed comparison in results between the previously proposed FACT algorithm and the proposed schemes in this paper is presented in numerical investigation section.

3. System Model

3.1. Preliminaries

An undirected graph $\mathcal{G} = \{\mathbb{V}, \mathbb{L}\}$ is used to represent the mobile network, where $\mathbb{V}$ expresses the set of vertices (i.e., network nodes) and $\mathbb{L}$ denotes the set of links. The available ECs in the network are expressed with a location set which is defined as $\mathbb{M} = \{1, 2, ..., M\} \subseteq \mathbb{V}$. Requests $r \in \mathbb{R}$ are assumed to be created by a set of MAR devices and their initial connected access router is $f(r)$. Due to mobility events, they can connect to different adjacent access routers k belonging to set $\mathbb{K} = \{1, 2, ..., K\} \subseteq \mathbb{V}$. Based on available historical data of a mobile network operator, it can be estimated with high accuracy a probability matrix P_{ij} which denotes the total moving probability of a mobility event between access routers $i, j \in \mathbb{K}$ [23].

As already eluded above, two main AR sets of functionalities are required to run (i.e., offloading) on the ECs. These are categorized into ARO caching related functions that require storage space and intense image processing functions requiring significant CPU processing power [11][12]. Hence, these two functionalities are expressed with η and ϱ for CPU processing and caching respectively. We denote the set of available AROs as $\mathbf{N}$={1,2,...,N}. A subset of those AROs, denoted as L_r, are randomly allocated to each request $r \in \mathbb{R}$. Thus, $l \in \mathbf{L_r} = \{1, 2, ..., L_r\} \subseteq \mathbf{N}$ denotes each ARO l that is required in the request r and its size is denoted as O_l^r. $F_{\eta r}$ and $F_{\varrho r}$ are used to describe the file size for each function of the request r. $F_{\eta r}$ represents the size of frames sent for object detection while $F_{\varrho r}$ represents the size of recognized objects used for matching. Denoting the frame length in its set as $s \in \mathbb{S}$, $F_{\eta r}$ is measured by σs^2; we note that the product of frame length s^2 and σ express the number of bits to represent a single pixel [4]. A nominal frame rate is assumed of 15 frames/second and the service could happen at every other frame ($\tilde{1}$33.2ms interval) [24][25][26]. Thus, the service delay of the aforementioned work flow within the above time interval could be regarded as acceptable.

Hereafter, we look into the scenario where a user requests a certain set of AROs and functions η and ϱ run in a pre-defined order at the same or different ECs. Due to the consideration of mobility events, user's potential points of attachment can influence these functions' hosted ECs. If a cache hit happens for all required AROs, the matched ones will then be sent back to the MAR device for displaying [11][12]. Otherwise, an extra latency D for ARO cache miss will be triggered since the AROs will have to be fetched from a remote core network server that host the augmented reality objects [12].

3.2. Mathematical Programming Formulation

According to the overall MAR system model and aforementioned preliminaries, the required decision variables are introduced paving the way for the proposed mathematical programming formulation. To this end, the decision variables are defined as follows,

$$x_{rj} = \begin{cases} 1, \text{ if computational function for request } r \\ \quad \text{located at node } j, \\ 0, \text{ otherwise.} \end{cases} \tag{1}$$

$$y_{rj} = \begin{cases} 1, \text{ if caching function for request } r \text{ located at} \\ \quad \text{node } j, \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

$$h_{rlj} = \begin{cases} 1, \text{ if ARO } l \text{ of request } r \text{ cached at} \\ \quad \text{node } j, \\ 0, \text{ otherwise.} \end{cases} \tag{3}$$

$$a_{rs} = \begin{cases} 1, \text{ if request } r \text{ use frame length } s, \\ 0, \text{ otherwise.} \end{cases} \tag{4}$$

The following decision variables are also defined to capture the case of cache miss or hit,

$$z_{rj} = \begin{cases} 1, \text{ if there is a cache hit for AROs of request } r \\ \quad \text{at node } j, \\ 0, \text{ otherwise.} \end{cases} \tag{5}$$

We describe the relationship between the decision variables h_{rlj} and z_{rj} as follows,

$$z_{rj} = \begin{cases} 1, \text{ if } \sum_{l \in N} h_{rlj} \geqslant L_r, \\ 0, \text{ otherwise.} \end{cases} \tag{6}$$

As mentioned above, it is necessary to ensure that if $\sum_{l=1}^{N} h_{rlj} \geqslant L_r$ then $z_{rj} = 1$ in the mathematical programming formulation. An equivalent either-or constraint can be applied to replace this conditional constraint. More specifically, it can be written as follows,

$$\left(\sum_{l \in N} h_{rlj} < L_r \right) \text{ or } (z_{rj} = 1) \tag{7}$$

A small tolerance value ϵ can be specified to change the first expression to inequality. Thus, the either-or logical constraint can be written as,

$$\left(\sum_{l \in N} h_{rlj} + \epsilon \leq L_r \right) \text{ or } (z_{rj} = 1) \tag{8}$$

A new decision variable q_j is now introduced with a large arbitrary number U to rewrite the above either-or constraint into the following expressions that need to be added in the mathematical programming formulation,

$$\begin{aligned} \sum_{l \in N} h_{rlj} + \epsilon \leq L_r + U(1 - q_j) \\ z_{rj} = 1 - q_j \end{aligned} \tag{9}$$

The overall cost measured in terms of delay can be split into wireless delay, wired network delay and computational delay [4]. The actual transmission time is mainly driven by the uploaded frame size and can be written as follows,

$$\frac{\sigma s^2 a_{rs}}{\widetilde{R}_r} \tag{10}$$

An average achievable data rate is assumed to be allocated to a user across the whole session lifetime and we are not posing any constraints and/or requirements on the instantaneous data rate. Hence, without loss of generality, an average rate of $\widetilde{R}_r$ at different connected cells is allocated to each request r based on a defined Service Level Agreement (SLA) [27].

The delay between ECs is noted by C_{ij}, which encapsulate the least communication delay (shortest path routing cost) between locations i and j [23]. Propagation delay is also captured together with other types of delay during this period such as for example nominal queuing delay.

Two types of computing delay at ECs are captured by parameters V_{rj} and W_{rj} and can be estimated as follows [28],

$$V_{rj} = \sum_{s \in S} \frac{\omega F_{\eta rs}}{f_V^j} a_{sr}, \; W_{rj} = \frac{\omega F_{\varrho r}}{f_V^j} \tag{11}$$

where

$$\sum_{s \in S} a_{sr} = 1 \tag{12}$$

In the formula above, the number of CPU cycles to process 1 byte input is represented by ω [28][29][30][31]. The VM computational speed f_V^j is assigned for request r at location j and is assumed, without loss of generality, to be allocated equally [31]. Given the condition that each VM takes up a fixed portion of a CPU core, the product of core frequency and allocated percentage can express its computational speed [32].

An EC at location j is assumed to have an allowable cache capacity Θ_j and an allowable computing capacity Δ_j measured by the number of server's remaining available VMs [31].

Finally, notations including key variables and brief explanations are summarized in table (1).

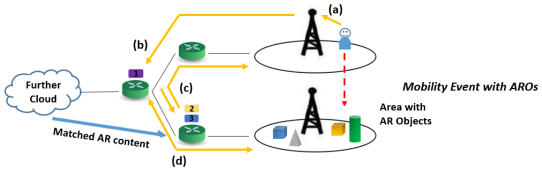

Figure 3: A toy example for Equation (13a)

As shown in the formulation below, the overall service latency is captured including transmission, computation and delay caused by cache retrieval and processing. The Equation (13a) consists of 4 main parts in order, which are shown in Fig. (3). The first term captures the wireless transmission delay from the user's initial location to the access point as shown by (a) in the figure. While the second term shown in the Figure by (b) captures the link delay to the first target edge cloud server and the processing delay of computational intensive functions at that location. The third term shown by (c) captures the link delay between two edge clouds, the processing delay of the matching function and the incurred delay of returning to the initial access point. Finally, the fourth term shown by (d) captures the link delay from potential destinations. The repeated process like wireless transmission or executing functions are not counted again for destinations. Note that both the so-called before and after mobility events are included in

the objective function and it captures the overall service latency. To this end, the mathematical optimization problem can be formulated as follows,

$$min \sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S}}\frac{F_{\eta rs}a_{rs}}{\widetilde{R}_r}+\sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}(C_{f(r)i}+V_{ri})x_{ri}+ \sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}\sum_{j\in\mathbf{M}}\Big(C_{ij}x_{ri}+W_{rj}+C_{jf(r)}+q_jD\Big)y_{rj}+ \sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}\sum_{k\in\mathbf{K}}P_{f(r)k}C_{ik}(x_{ri}+y_{ri}) \tag{13a}$$

s.t. (12)

$$\sum_{r\in\mathbf{R}}\sum_{l\in\mathbf{L_r}}h_{rlj}O_l\le\Theta_j,\forall j\in\mathbf{M} \tag{13b}$$

$$\sum_{r\in\mathbf{R}}x_{rj}+y_{rj}\le\Delta_j,\forall j\in\mathbf{M} \tag{13c}$$

$$\sum_{l\in\mathbf{N}}h_{rlj}+\epsilon\le L_r+U(1-q_j)\ \forall j\in\mathbf{M},r\in\mathbf{R} \tag{13d}$$

$$z_{rj}=1-q_j,\ \forall j\in\mathbf{M},r\in\mathbf{R} \tag{13e}$$

$$\sum_{j\in\mathbf{M}}x_{rj}=1,\forall r\in\mathbf{R} \tag{13f}$$

$$\sum_{j\in\mathbf{M}}y_{rj}=1,\forall r\in\mathbf{R} \tag{13g}$$

$$a_{rs},x_{rj},y_{rj},h_{rlj},z_{rj},q_j\in\{0,1\},\ \forall r\in\mathbf{R},j\in\mathbf{M},l\in\mathbf{L},s\in\mathbf{S} \tag{13h}$$

The different costs added to delay are captured by the objective function (13a), which embed the routing to the ECs and computational delays for the two functions of AR video processing and ARO caching. Cache capacity limit is captured by constraint (13b), while the limit on the number of utilized VMs is captured by constraint (13c). As explained earlier, we initially have the case that if AROs required by a user are all pre-cached, then it is a cache hit. This is equivalent to an either-or Constraint (7). Through adding a tolerance value to be Constraint (8) and then bringing in the decision variable q_j, the both cases could be merged into an inequality expression, Constraint (13d), which describes the cache hit/miss requirement. Constraint (13e) help bind decision variables z_{rj} and q_j. Finally, constraints (13f) and (13g) capture the initialization requirement for MAR functions which means they should run only once at a specific EC location for each request.

Note that the product of decision variables make the previous defined mathematical program (13a) non-linear. However, it is possible to linearize the problem by introducing the following auxiliary decision variables,

$$\sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}\sum_{j\in\mathbf{M}}\xi_{rij}=\sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}\sum_{j\in\mathbf{M}}x_{ri}y_{rj} \tag{14}$$

$$\sum_{r\in\mathbf{R}}\sum_{j\in\mathbf{M}}\psi_{rj}=\sum_{r\in\mathbf{R}}\sum_{j\in\mathbf{M}}q_jy_{rj} \tag{15}$$

$$\sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S}}\sum_{j\in\mathbf{M}}\phi_{rsj}=\sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S}}\sum_{j\in\mathbf{M}}a_{rs}x_{rj} \tag{16}$$

In addition, the binding constraints added for these new decision variables are as follows,

$$\xi_{rij}\le x_{ri},\ \forall r\in\mathbf{R},i,j\in\mathbf{M} \tag{17}$$

$$\xi_{rij}\le y_{rj},\ \forall r\in\mathbf{R},i,j\in\mathbf{M} \tag{18}$$

$$\xi_{rij}\ge x_{ri}+y_{rj}-1,\ \forall r\in\mathbf{R},i,j\in\mathbf{M} \tag{19}$$

$$\psi_{rj}\le q_j,\ \forall r\in\mathbf{R},j\in\mathbf{M} \tag{20}$$

$$\psi_{rj}\le y_{rj},\ \forall r\in\mathbf{R},j\in\mathbf{M} \tag{21}$$

$$\psi_{rj}\ge q_j+y_{rj}-1,\ \forall r\in\mathbf{R},j\in\mathbf{M} \tag{22}$$

$$\phi_{rsj}\le a_{rs},\ \forall r\in\mathbf{R},s\in\mathbf{S},j\in\mathbf{M} \tag{23}$$

$$\phi_{rsj}\le x_{rj},\ \forall r\in\mathbf{R},s\in\mathbf{S},j\in\mathbf{M} \tag{24}$$

$$\phi_{rsi}\ge a_{rs}+x_{rj}-1,\ \forall r\in\mathbf{R},s\in\mathbf{S},j\in\mathbf{M} \tag{25}$$

Finally, the previous defined expression of delay can be rewritten into its linearized version as follows,

$$min \sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S}}\frac{F_{\eta rs}a_{rs}}{\widetilde{R}_r}+\sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}(C_{f(r)i}x_{ri}+\frac{F_{\eta rs}\phi_{rsi}}{f_V^i})+ \sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}\sum_{j\in\mathbf{M}}\Big(C_{ij}\xi_{rij}+(W_{rj}+C_{jf(r)})y_{rj}+\psi_{rj}D\Big)+ \sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}\sum_{k\in\mathbf{K}}P_{f(r)k}C_{ik}(x_{ri}+y_{ri}) \tag{26a}$$

s.t. (12)

$$\sum_{r\in\mathbf{R}}\sum_{l\in\mathbf{L_r}}h_{rlj}O_l\le\Theta_j,\forall j\in\mathbf{M} \tag{26b}$$

$$\sum_{r\in\mathbf{R}}(x_{rj}+y_{rj})\le\Delta_j,\forall j\in\mathbf{M} \tag{26c}$$

$$\sum_{l\in\mathbf{N}}h_{rlj}+\epsilon\le L_r+U(1-q_j)\ \forall j\in\mathbf{M},r\in\mathbf{R} \tag{26d}$$

$$z_{rj}=1-q_j,\ \forall j\in\mathbf{M},r\in\mathbf{R} \tag{26e}$$

$$\sum_{j\in\mathbf{M}}x_{rj}=1,\forall r\in\mathbf{R} \tag{26f}$$

Table 1: Notation

Parameter	Description
$\mathbf{N}$	Set of all AROs
$\mathbf{M}$	Set of Edge Clouds
$\mathbf{R}$	Set of User Requests
Θ_j	Cache capacity at EC j
$\mathbf{S}$	Set of possible frame lengths
$f(s)$	get accuracy from frame length
$g(r)$	Initial access router for request r
Δ_j	Number of available VMs at EC j
P_{ij}	Mobility probability from EC i to j
D	Fixed delay cost caused by a cache miss
η, ϱ	Two types of AR functions (CPU, Cache)
C_{ij}	Communication delay between nodes i and j
$O_l, \mathbf{L_r}$	Size of object l, and set of objects in request r
σ	the number of bits required to represent 1 pixel
a_{rs}	0/1 var.: if the frame legnth s is selected for r
x_{rj}, y_{rj}	0/1 var.: if function η or ϱ for r is set at EC j
ω, f_V^j	Computational load and CPU availability at EC j
V_{rj}, W_{rj}	Processing delay for request r for function η, ϱ at EC j
$F_{\eta r}, F_{\varrho r}$	Input video frame size for functions η, ϱ in request r
h_{rlj}	0/1 var.: if object l appearing in r is cached at EC j
z_{rj}	0/1 var.: if there is a cache hit of all ARO for r at EC j

$$\sum_{j \in \mathbf{M}} y_{rj} = 1, \forall r \in \mathbf{R} \tag{26g}$$

$$(21) - (29) \tag{26h}$$

$$a_{rs}, x_{rj}, y_{rj}, h_{rlj}, z_{rj}, q_j \in \{0, 1\}, \forall r \in \mathbf{R}, j \in \mathbf{M}, l \in \mathbf{L}, s \in \mathbf{S} \tag{26i}$$

$$\psi_{rj}, \xi_{rij}, \phi_{rsj} \in \{0, 1\}, \forall r \in \mathbf{R}, i, j \in \mathbf{M}, s \in \mathbf{S} \tag{26j}$$

The relation between uploaded frame size and its accuracy can be given below by an approximation function in [4],

$$f(s) = 1 - 1.578e^{-6.5 \times 10^{-3} s} \tag{27}$$

Note that the RMSE (root mean square error) of function $f(s)$ is less than 0.03 [4]. Thus, the product $f(s)a_{rs}$ can transfer frame lengths into discrete values with high accuracy.

Finally, the frame selection, EC allocation and route decision problem in the MAR system can be transferred into an optimization problem,

$$\begin{aligned} min\ F &= \mu \frac{L}{L_{max}} - (1 - \mu) \sum_{r \in R} \sum_{s \in S} \frac{f(s)a_{rs}}{f(s)_{max}} \\ &= \mu \frac{L}{L_{max}} - \frac{(1 - \mu)}{|R|} \sum_{r \in R} \sum_{s \in S} f(s)a_{rs} \end{aligned} \tag{28a}$$

$$\text{s.t. } (12), (26b) - (26j) \tag{28b}$$

Where $\mu \in [0, 1]$ is the weight parameter, L is the objective function shown in (26a) and $f(s)_{max}$ is assumed to be 1. The above described integer linear optimization problem due to the combinatorial nature can not be solved efficiently for large network instances. State of art branch and bound techniques can help finding the optimal solution for low to medium size network instances. Hence, metaheuristic algorithms are needed to provide competitive solutions whilst being amenable for real-time implementation.

3.3. A Simulated Annealing Based Heuristic

Although the proposed scheme can provide an optimal decision making, suffers from the curse of dimensionality due to the combinatorial nature of the mixed integer linear program (MILP). To this end, a Simulated Annealing framework is developed that evolves from a basic greedy solution and avoids sinking into local optima (please refer to [33][34] and references therein regarding the specifics of the simulated annealing framework). In this paper, a Simulated Annealing based algorithm firstly starts with a greedy scheme that selects an EC which is close in distance (Euclidean) to the user and then continuously switching to neighbor solutions

to identify ones that increase the quality of the objective function. The typical cooling process of Simulated Annealing allows a quick decision making stage and accepts worse case sometimes in a smaller and smaller range during iterations [33][34]. The pseudo-code of the proposed SA-based mobility aware AR algorithm (SAMAR) is shown in (1). Temperature T can be calculated from $T = 0.93^k \frac{l}{200}$, where l is the frame length and k is the current number of iterations ($k \in [1, 500]$). Normalizing the gap between neighbor solutions over the temperature through $\exp^{-\frac{f(k)-f(k-1)}{T}}$ into (0,1), we get the probability on accepting the solution $f(k)$ even if it's worse ($f(k) \geq f(k-1)$). Noticing that the temperature goes down with the number of iterations, the probability becomes smaller as well. Thus, the SAMAR schemes also accepts the worse solution in a decreasing probability. From (1), the first loop generates a basic greedy solution based on the distance and vCPU frequency. Then multiple iterations are applied to explore its neighbor solutions to find a solution of increased quality. The neighbor solution means that compared with the original solution, only one request's assignment or frame length choice is changed randomly to another feasible one. SAMAR accepts a better solution and an inferior one within a probability controlled by the temperature parameter.

3.4. A Long Short Term Memory (LSTM) Neural Network Based Scheme

To increase the efficiency and reliability in responding to the changes in the network conditions, a LSTM neural network based scheme is developed in this section. It is a well known and used machine learning technique that avoids gradient vanishing and gradient exploding through keeping the important information and key features in the long term while forgetting unnecessary inferences [35][36]. Hence, in this paper, it is utilized to learn from optimal solutions and provide high quality decision making during the inference phase in an efficient manner. Regarding the service requests the possible destinations in terms of selecting edge cloud support are limited to the adjacent servers compared to the current location of a user and is denoted as $d_r \in \mathbb{K}$ for each request. According to previous notations, routing paths per request are pre-defined and denoted as $RT_r = [f(r), x_{ri}, y_{rj}, d_r]$. The set of this matrix $\mathbb{RT}$ is provided by the optimal scheme described in the previous section (denoted as Optim in the sequel). During training, initial locations and destinations of requests can be fed as input and decisions of where to anchor MAR functionalities can be provided

Algorithm 1 SAMAR

Input: requests **R**, EC list **ECL**, weight μ, distances between servers **C**, frame length set **L**, destinations set **K**

Output: server assignment **A**, chosen frame length CFL

for $r \in$ **R do**
  ECL ← sort servers by its distance to starting point $s(r)$ and all possible destinations **K**;
  FL ← select n closest servers in **ECL** and sort again according to vcpu frequency;
  A ← select $\{i, j\}$ from top servers in **FL**;
  if free VM at $i == 0$ **then**
    ECL ← remove i-th server;
  end if
  if free VM at $j == 0$ **and** $i \neq j$ **then**
    ECL ← remove j-th server;
  end if
  for $l \in$ **L do**
    $f(l)$ ← value of formula (28a) with assignment **A** and frame length l;
  end for
  $f(0), CFL \leftarrow$ minimum $f(l)$ and corresponding l;
end for
for $k = 1 \rightarrow BOUND$ **do**
  $\{\mathbf{A}', CFL'\} \leftarrow$ swift to neighbor solution
  $f(k)$ ← value of formula (28a) with assignment $\mathbf{A}'$ and CFL';
  $p \leftarrow$ a random number from 0 to 1;
  if $f(k) < f(k-1)$ **then**
    $\{\mathbf{A}, CFL\} \leftarrow \{\mathbf{A}', CFL'\}$
  else
    if $p < \exp^{-\frac{f(k)-f(k-1)}{T}}$ **then**
      $\{\mathbf{A}, CFL\} \leftarrow \{\mathbf{A}', CFL'\}$
    end if
  end if
end for
return $\{\mathbf{A}, CFL\}$

as the output. Thus, the route matrix can be separated into $X_r = [s_r, d_r] \in \mathbb{X}$ and $Y_r = [x_{ri}, y_{rj}] \in \mathbb{Y}$. Most routes inside consist a training set while the rest of them are applied for testing. Thus, $\mathbb{XTrain}, \mathbb{XTest} \subset \mathbb{X}$ and $\mathbb{YTrain}, \mathbb{YTest} \subset \mathbb{Y}$ are denoted to differentiate training and testing sets. The predictions made by this scheme are denoted as $\mathbb{Pred}$. Firstly, the Optim scheme is executed to provide the optimal decision making for anchoring MAR functionalities to different ECs. The process from obtaining optimal solutions to providing predictions is presented in Fig. (4). $\mathbb{YTrain}$ is adapted to the categorical type to enable the scheme to complete the classification work. Selecting any two different access routers from the set $\mathbb{K}$ give K^2 different types of results. Hence, each assignment is allocated a unique index $Ind_r \subset \{1, ..., M^2\}$ to differentiate its type. During training, the scheme will aim to classify and fathom out the relation between input sets. Then, the predictions are provided and transferred back for Feasibility Check. This stage will check whether the predictions can still fit the original network settings and satisfy all constraints. If an overloaded EC is selected, the request will be sent to a core cloud deeper in the network and as a result this allocation will trigger an extra penalty. The predictions are finally compared with the optimal solutions to evaluate the quality of the LSTM-based scheme.

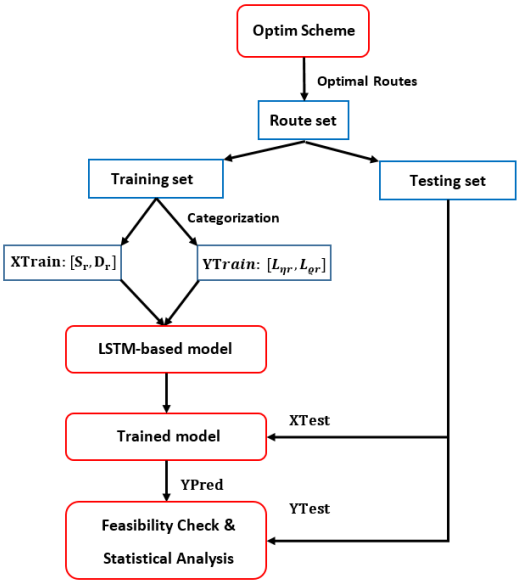

Figure 4: Working process

Fig. (5) shows the architecture of the LSTM-based scheme. The LSTM layer follows the nominal design without modifying its state changing and gate controlling formula. Therefore, the potential of nominal LSTM network for service decomposition is explored. A dropout layer follows the LSTM layer to avoid overfitting by setting a random number of elements to 0 ($rand(size(X)) < P$, X is a layer input and P is a self-defined probability) and scaling other elements by $\frac{1}{1-P}$ [37][38]. Two sets of such layers are applied to improve the overall performance. Then, a fully connected layer multiplies a weight matrix and adds a bias vector to the result in order to combine features and learn the underlying pattern [39][40]. Since the logistic sigmoid function is a smooth curve in (0,1) and good at classify sets without very large differences,it is applied as the output activation function in the softmax layer [41]. Finally, the classification layer calculates the entropy loss at the current time step t, denoted as y_t. This process is repeated until the training phase is concluded and then the resulting LSTM neural network can be used for real-time prediction at the inference stage.

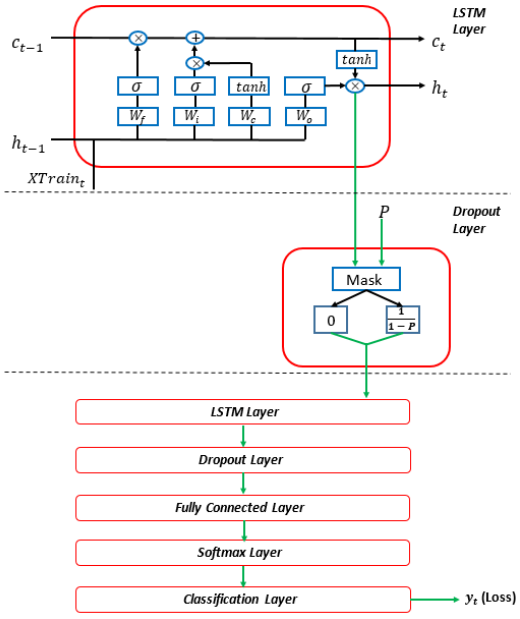

Figure 5: Layered structure of LSTM-based model (at time step t)

3.5. Architectural View: Implementation Aspects

We expect that the proposed set of algorithms to be embedded into an edge computing 5G and beyond platform. Native support of edge computing is being supported even from Release 17 within 3GPP.

According to Rel-17 3Gpp, the edge data network provides the service to application clients and consists of edge application servers, edge enabler servers and edge configuration servers. The proposed schemes

could run by a given module, known as Edge Application Server Discovery Function (EASDF), which acts as a Domain Name System (DNS) resolver and complements the DNS queries with user location and related information. At the same time, the enhancements in the User Equipment route-selection policy (URSP) could better support the distributed anchoring of applications and the relocation of the server. Thus, the EASDF module with the URSP policy enables the utilization of advanced allocation schemes in an EC supported network. The implementation of approaches further requires the dynamic availability (e.g. deployment changes and user mobility), network capability exposure (e.g.Network Exposure Function) and the support for seamless service continuity.

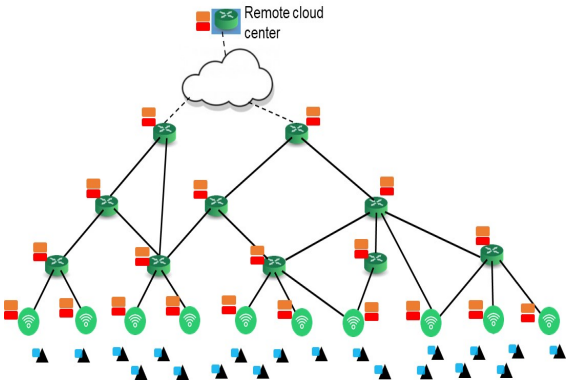

Figure 6: Typical tree-like designed network topology

4. Numerical investigations

4.1. Parameterization

In this section, the proposed schemes are investigated and compared vis-á-vis with baseline schemes to assess their performances through a wide-set of numerical investigations.

Throughout the simulations, the wireless transmission rate from MAR device to the access point under 5G environment is set in the range of 100 to 120 Mbps. Each EC is assumed to have a physical CPU with 8 cores and 16GB memory [23][11]. The CPU frequency is generated randomly for each EC ranging from 2 to 4 GHz [11][32]. Each VM in an EC is assumed to take up the CPU resources equally and hence virtual CPU frequencies between these VMs can also be equally split [32]. In each EC, up to 14 VMs are assumed to be available. In addition, the computation load ω at each EC is assumed as 10 cycles/bit [42]. The communication latency between the terminal and target ECs can be achieved through adding link delays in the shortest path. The average latency for each hop is assumed to be 2ms, including propagation delay and queuing delay [43][44]. When a cache miss takes place, the corresponding penalty is triggered and set as 25ms. A tree-like topology is created as presented in Figure (6) and is supported by dense ECs (20 in total). Because MAR applications are not necessary to run on every EC in the network, only a random subset of ECs are assumed to be active in simulations.

The majority of AR applications tend to apply square shaped frames or vision markers due to its advantage in providing multiple co-planar corresponding points [45]. To balance the usability, efficiency, accuracy and reliability, frame lengths like VGA (640×480 pixels) and Stefan (352×288 pixels) are accepted as suitable ones, such as for example in ARCore, ARToolKit and MixedRealityToolKit [46][45][47]. Although some AR applications could handle high resolution images, a small-sized region of markers (from $200 \times 200 to 514 \times 414$ in VGA sized frames) are uploaded for processing when focusing on its marker detection system such as for example in ARToolKit (ATK) or in Siemens Corporate Research (SCR) [45]. Hence, in this paper, the average length of uploaded video frames (or region of markers) to the EC is chosen from the following set: $\{200 \times 200, 400 \times 400, 600 \times 600, 800 \times 800\}$ (pixels). These video frame lengths can be translated to approximately 0.12MB to 2MB in terms of data size [4]. After extracting features from frames, each compact feature representation (using fisher vectors for example) are around 25KB ($[100, 300]$KB in total) [48]. Each ARO is $(0, 50]$MB [11] and we assume that follows a normal distribution. Since we simulate in a small scale network, a nominal LSTM network, including a single LSTM layer is utilized. The acceptable validation accuracy is set above 85% and if the trained network only fits the training set without satisfying this requirement for other testing sets (over-fitting), the network has to be retrained with fewer neurons and a larger dropout rate. In the simulation, we start from 100 neurons and reduce by 5 until finding a suitable LSTM network with 80 neurons. 90% of 300 route vectors with corresponding optimal decisions are used for training, while the rest 10% are used for validation. The initial learning rate is set to 0.005, the maximum number of epochs is 160 and the dropout probability is 5% to avoid over-fitting. These key parameters used in the numerical investigations are summarized in Table (2).

The weight parameter (μ) as defined in (28a) belongs to $[0, 1]$ while the assumed β value for the FACT scheme

Table 2: Simulation parameters

Parameter	Value
Number of available ECs	6
Number of requests	$[20, 40]$
AR object size	$(0, 50]$ Mbyte
Total moving probability	$[0, 1]$
Frame length	$\{200^2, 400^2, 600^2, 800^2\}$ pixels
Compact feature representations	$[100, 300]$ KByte
CPU Memory Capacity	16 GByte
CPU frequency	2 – 4 GHz
CPU cores	8
CPU portion per VM	0.125 – 0.25
Computation load	10 cycles/bit
Wireless transmission rate	$[100, 120]$ Mbps
Average hop's latency	2 ms
Cache miss penalty	25 ms
Number of user route vectors / optimal decisions	300
Initial LSTM learning rate	0.005
Maximum number of epochs	160
LSTM Dropout probability	5%

is a positive number (the assumed range is $[0, 100]$ in [4]). When implementing the FACT scheme for comparison, detailed settings of simulation in [4] are adjusted according to the system settings in this work. Besides, cache limit and cache miss penalty are not considered in FACT [4]. During implementation, the penalty is triggered whenever FACT's solution excesses the limit. In order to compare the two schemes fairly, after deciding a value for μ, we calculate the corresponding accuracy in the proposed scheme and then experiment with different β values for the FACT scheme to find a suitable one so that both schemes share the same level of accuracy. The relation between the weighting parameters μ and β when sharing a common level of accuracy is presented by Fig. (7). It shows a very different tendency between the FACT scheme and the proposed optimal framework based on mathematical programming. For example, when our proposed scheme sets the weight μ as 0.5 which regards accuracy and latency as equal, the weight β of the FACT scheme is only 20 to achieve the same level of frame accuracy. As pointed out by authors in [4], such a small value indicates a decreased importance on the frame accuracy. Hence, by considering also mobility, service decomposition and an increased set of constraints, the proposed scheme is able to provide better decision making and hence increase the performance of the application. The comparison of service delay with the same level of frame accuracy are discussed in detail in the next section.

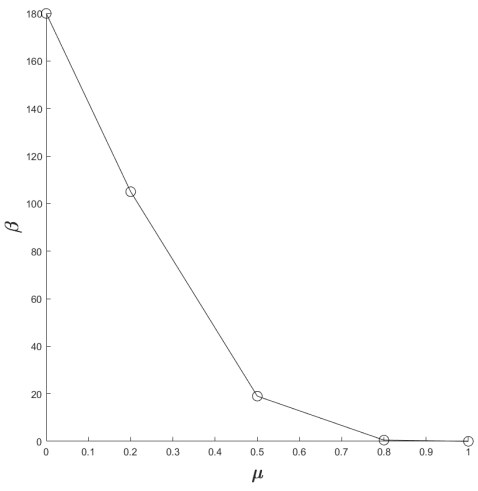

Figure 7: Relation between weight parameters μ and β

Three other schemes are also designed for comparison reasons. These are the random selection scheme (RandS), the closest-first scheme (CFS) [49] and the utilization based scheme (UTIL) [50]. The RandS scheme randomly selects an available EC and picks a free VM from the selected EC. The CFS scheme focuses on the nearest EC to the user and accepts an available neighbor

EC as a backup choice [49]. Similarly, the UTIL scheme selects the closest EC at first but will turn to neighbor least loaded ECs if the closest server is highly utilized (for example over 80% of the total VMs are occupied) [50]. Note that the penalty still works for the above schemes when their first and backup choices are all not feasible in the network. Overall delay and after mobility event delay are defined here for a more in-depth analysis of different schemes and a detailed comparison. The overall delay includes the aggregated latency consumed in communication and is in essence the result of formula (28a). However, the so called "after mobility event delay" captures the communication process and acquired delay between the assigned EC and the potential destinations. The results reported below are average values stemming from 50 Monte Carlo simulations.

4.2. Simulation Results

As eluded in the related work section, mobility awareness in edge cloud support is one of the main difference between the proposed schemes and other schemes. The LSTM scheme, which is trained by the Optim scheme, is able to capture efficiently the effect of mobility. Table (3) evaluates the impact of the total moving probability on the performance of schemes' delay (the results presented are for 30 Requests, the value of the weight μ is equal to 0.5 and we assume 14 units for the EC capacity). Observe from rows 1-6 in table (3) that the delay reduction achieved by the Optim scheme is increasingly more significant compared to the other schemes when the total moving probability increases. Note that, as expected, all schemes share a similar level of overall delay when there is enough capacity and no mobility. As compared with the FACT scheme, the Optim scheme reduces delay up to 8% and maintain the similar frame accuracy (β = 20) when the total moving probability increases. From row 7-12 in table (3), CFS, UTIL and FACT schemes faced increased delays which is significant evident in the after mobility event because they always try to deploy the service at an adjacent EC to the end user instead of considering potential destinations due to mobility effects. Therefore, the Optim scheme owns an obvious advantage compared to the other schemes in a high moving probability scenario, especially in the so-called after mobility event.

Another aspect is the available EC capacity, i.e., the number of unused VMs in an activated EC. Although the FACT scheme does not explicitly utilize the notion of VMs as is the case of the proposed schemes and the Optim framework in this paper, it still has computing capacity measured by TFLOPS [4]. Thus, by translating this computing capacity to EC capacity, results of the FACT scheme can be compared with others'. The range of EC capacity is 10 to 14 and the corresponding computing capacity in the FACT scheme is 0.63 to 0.88 TFLOPS. As shown in Figure (8) and (10), when EC capacity increases, the average delay for all schemes decreases because an increased available capacity usually leads to a better potential solution and a lower probability of overload and cache miss. Notice that the UTIL scheme is more sensitive to capacity than others and hence has the fastest dropping rate. The RandS scheme is worse in overall delay than CFS, FACT and UTIL schemes, but is better in the after mobility event. The reason is that the RandS scheme treats all active servers equally while other three schemes tend to select the closest server to the user's initial location. From this figure and Fig. (9), the LSTM-based scheme is slightly better than the SAMAR scheme. It is interesting to show that the gap between the Optim scheme and the proposed LSTM-scheme is also quite narrow. Different values of parameters will not interfere its learning process and affect its learning effect. Therefore, the gap between the LSTM-based scheme and the Optim scheme will maintain narrow at different network conditions, which reveals the reliability and stability of this proposed scheme. Clearly, the advantages of the proposed three schemes become more evident in a congested network. Most greedy schemes tend to allocate MAR services to a few ECs. They neglect the flexibility of decomposition and hence will suffer from a serious penalty because of the limited memory space and computing resources in the case of network congestion episodes.

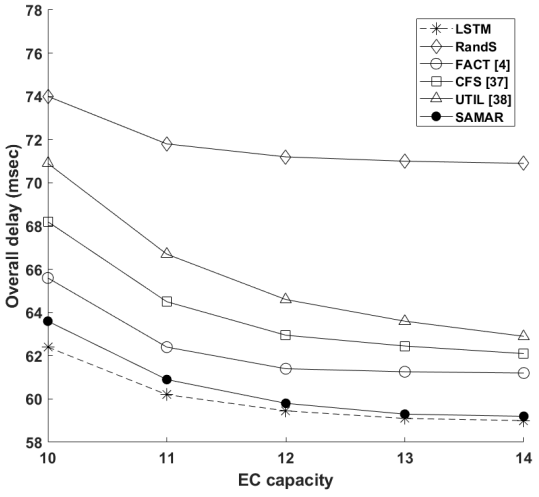

Figure 8: Overall delay for schemes under different EC capacities (P=1, μ = 0.5 and 30 requests)

Table 3: Combined delay table for different moving probabilities(msec)

P	Optim	FACT	CFS	RandS	UTIL	SAMAR	LSTM
0	47.2	47.2	47.6	56.1	50.4	47.4	48.5
0.2	49.4	50.3	50.8	59.1	53.1	49.8	50.9
0.4	51.5	53.5	54.0	62.3	55.7	52.1	52.7
0.6	53.6	56.7	57.3	65.4	58.5	54.5	55.2
0.8	55.8	59.9	60.5	68.6	61.3	57.0	57.2
1	58.2	63.2	63.8	71.8	64.1	59.7	59.9
Overall delay for schemes under different moving probabilities							
0	0	0	0	0	0	0	0
0.2	10.6	12.0	12.2	11.2	12.4	10.8	10.8
0.4	21.3	24.1	24.6	22.7	24.7	21.6	21.7
0.6	32.0	36.1	36.9	33.9	37.0	32.4	32.6
0.8	42.5	48.1	49.2	45.4	49.4	43.1	43.4
1	53.2	60.2	61.5	56.7	61.8	53.9	54.2
After mobility event delay for schemes under different moving probabilities							

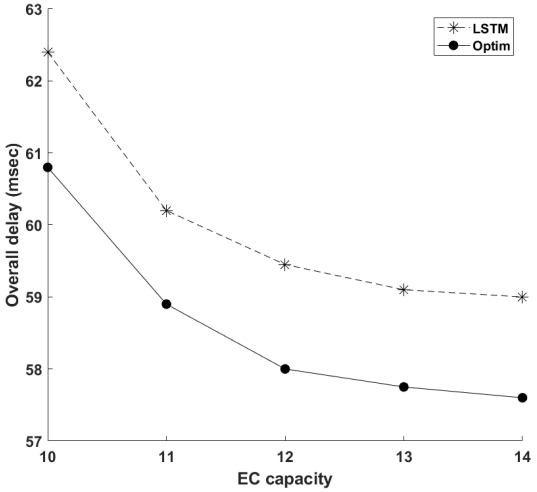

Figure 9: Difference in performance between LSTM and Optim

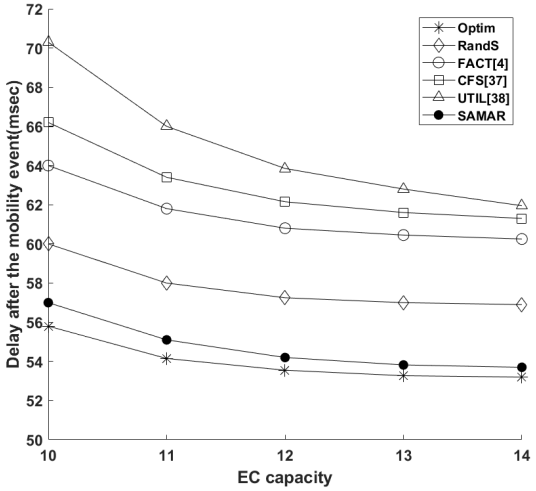

Figure 10: After mobility event delay for schemes under different EC capacities (P=1, μ = 0.5 and 30 requests)

We are also looking into the effects of increased number of requests stemming from mobile augmented reality users which increase network utilization and congestion in the network. As expected, both types of delay has an increasing trend for all schemes as the number of requests increase but the Optim scheme seems to better maintain lower overall latency with a slower ascending trend. These findings are illustrated in Figure (11) and Figure (12). Observe that the CFS and FACT schemes witness an increased delay and this is because their EC selection methods create several "hot spots" areas and these "hot spots" lead inevitably to higher delays as the number of requests increases. Also, the UTIL scheme by redirecting requests to adjacent ECs when an EC is operating at more than 80% utilization make this scheme also this scheme more vulnerable to network congestion episodes due to increased number of requests. However, it performs better under a low utilization level in a mobility event due to its balanced workload. Finally, the proposed SAMAR scheme approaches the Optim scheme and is less sensitive to increased congestion compared to the other baseline schemes.

Table (4) shows the variation of delay under different values of weight μ (30 Requests, P=1 and 14 units of EC

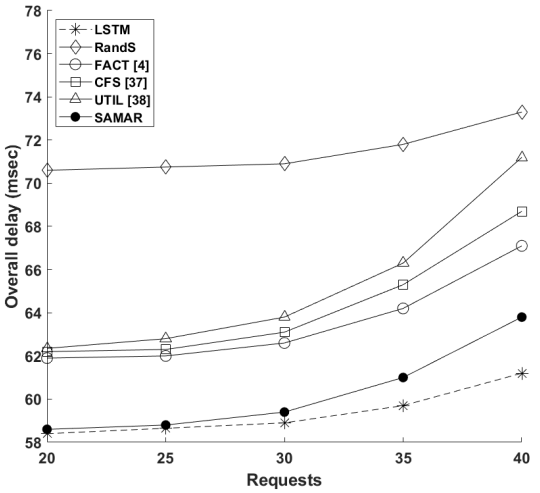

Figure 11: Overall delay for schemes under different number of requests (P=1, μ = 0.5 and 14 units of EC capacity)

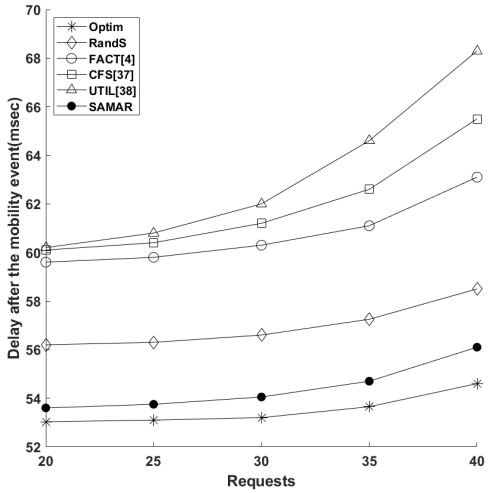

Figure 12: After mobility event delay for schemes under different number of requests (P=1, μ = 0.5 and 14 units of EC capacity)

capacity). The Optim scheme is around 10.2% less than the FACT scheme in overall delay and around 14.0% less in after mobility event delay. According to Table (4), the SAMAR scheme achieves a similar level of delay to the Optim scheme. Compared with the SAMAR scheme, the LSTM-based scheme is more stable and only a bit inferior to the SAMAR scheme with a large weight parameter.

In simulations, when comparing predictions (LSTM-based) and optimal solutions (Optim), we get $rmse < 1$ (root mean square error), $\delta < 11\%$ (relative error) and $r^2 > 0.88$ (determination coefficient) in most of the cases. This is a clear indication that the predictions can be deemed as reliable and highly competitive to the optimal solutions. According to Formula (29) and (30), small values of *rmse* and δ indicate that the LSTM-based scheme generates a decision policy that closely resembles the optimal solution. The comparison of *rmse* values of all schemes is shown by Table (5). The indexes have a location information, i.e., access routers which are topologically close have also adjacent indices. The user mobility is limited to adjacent servers within each period for active MAR sessions. Thus, the high similarity to the optimal solutions indicates the high-quality solutions of the LSTM-based scheme. In addition, the value of r^2 also reveals that the LSTM-based scheme has a strong fitting ability in the MAR service decomposition problem. It is rarely the case in experimentation when the request is served by a remote cloud server, hence triggering a penalty. The variation of training and validation accuracy is shown in Fig. (13). The LSTM-based scheme gradually increases its decision making quality and closely follows the training set. The training and validation accuracy increase with iterations and eventually converge at 93.6% and 88% respectively.

$$rmse = \sqrt{\frac{\sum_{i=1}^{n} (YTest_i - YPred_i)^2}{n}} \tag{29}$$

$$\delta = \frac{\sum_{i=1}^{n} \left| \frac{YTest_i}{YPred_i} - 1 \right|}{n} \times 100\% \tag{30}$$

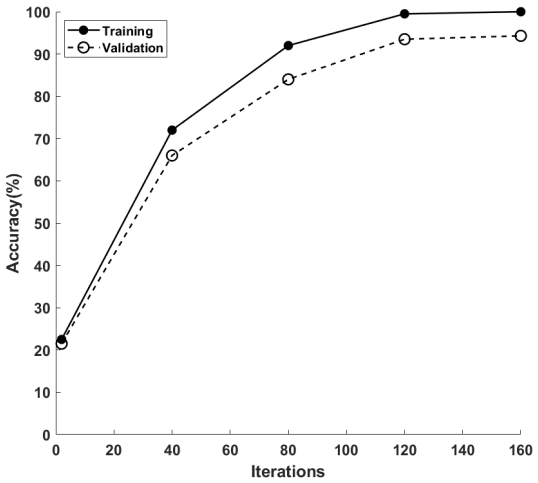

Figure 13: Training and Validation Accuracy with increasing iterations

Compared with the Optim scheme, the SAMAR scheme and the inference stage of the LSTM-based

Table 4: Combined delay table for different weights(msec)

μ	Optim	FACT	CFS	RandS	UTIL	SAMAR	LSTM
0	163.7	177.8	179.3	184.2	180.8	167.1	165.4
0.2	104.9	114.2	115.3	122.6	116.1	108.0	106.5
0.5	58.2	63.1	64.3	71.7	64.1	59.6	59.6
0.8	29.5	33.9	34.9	41.7	35.5	30.7	30.8
1	29.4	33.9	34.8	41.6	35.3	30.5	30.7
Overall delay for schemes under different weights							
0	158.9	174.7	176.7	169.4	177.1	161.5	160.8
0.2	100.3	111.0	112.7	107.8	113.3	102.6	102.9
0.5	53.2	60.2	61.3	56.7	61.9	53.8	54.7
0.8	24.7	30.9	32.3	26.8	33.2	25.3	25.9
1	24.7	30.8	32.2	26.8	33.1	25.2	25.9
After mobility event delay for schemes under different weights							

Table 5: RMSE Values of All Schemes

Algorithm	LSTM	RandS	CFS	FACT	UTIL	SAMAR
RMSE	0.9	6.8	1.9	1.5	3.4	1.3

scheme are efficient in terms of complexity. The computational complexity of the SAMAR scheme is $O(n^2)$. According to (6) (executed once), the SAMAR scheme has a slight increased complexity compared to RandS, CFS and UTIL. Although relaxing an ILP into an LP problem in general greatly reduces the computational time for FACT[4], constructing and solving linear programming optimizing problems still consume more time than other schemes. The Optim scheme solves a complex MILP problem and hence is the most time consuming one. As shown by Fig. (14) and (6), the LSTM-based scheme suffers from a time consuming process for obtaining the solutions for network training, but its inference stage itself is efficient compared with other algorithms. Note that its solution obtaining and network training stage only needs to execute once offline, its efficient inference stage is more dominant in the whole process. However, it is also possible to train the LSTM neural network with advanced metaheuristic algorithms (like SAMAR for example) and provide real-time decision making during the inference phase.

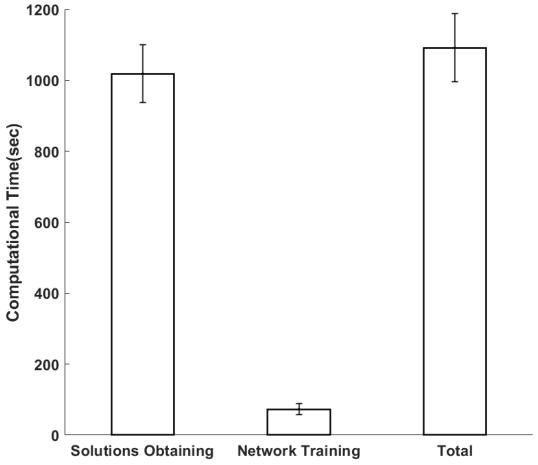

Figure 14: Average execution time of the training phase (30 Requests and 14 units of Capacity)

5. Conclusions

Mobile augmented reality applications are still in a rather embryonic state but they currently attracting significant attention from both academia and industry stemming from the increased capabilities offered from the network side as well as the terminals. The ability of augmented reality services to overlay digital content over the physical space and surroundings results in an immersive metaverse type of environment which open up the possibility for a plethora of potential use cases and applications. As with respect to mobile wireless networks, mobility has been proved to be an important factor in service latency of MAR systems. Previous research mainly focused on various techniques of

Table 6: Processing time of algorithms

Algorithm	Average Processing time(sec)	STD
RandS	1.076	0.151
CFS	1.083	0.156
UTIL	1.091	0.155
SAMAR	1.497	0.163
FACT	2.132	0.447
LSTM	1.727	0.205
Optim	201.506	21.965

*tested in a PC with intel i7, 6500U, 2 cores

offloading processing without considering the decomposition of MAR functions and the effect of mobility in an explicit manner. To this end, this paper proposes an optimization framework that considers a rich set of augmented reality specific constraints, provides service decomposition and allows a trade off between service latency and frame accuracy to improve the communication efficiency. Based on this scheme, a simulated annealing based mobility aware AR (SAMAR) algorithm and a long short term memory based scheme (LSTM) neural network for predicting edge clouds for anchoring MAR functions are designed for achieving high-quality solutions whilst being amenable for real-time implementation. Both proposed schemes deliver competitive performance and focusing on the inference stage, the long short term memory based scheme (LSTM) shows enhanced performance and shorter execution time.

As illustrated by a wide set of numerical investigations, the proposed set of solutions can efficiently decompose the augmented reality service to available edge cloud resources by considering mobility patterns of users and hence increase the quality of decision making compared to other previously proposed and baseline schemes.

References

[1] D. Chatzopoulos, C. Bermejo, Z. Huang, P. Hui, Mobile augmented reality survey: From where we are to where we go, Ieee Access 5 (2017) 6917–6950.

[2] X. Qiao, P. Ren, S. Dustdar, L. Liu, H. Ma, J. Chen, Web ar: A promising future for mobile augmented reality—state of the art, challenges, and insights, Proceedings of the IEEE 107 (4) (2019) 651–666.

[3] L. Li, X. Qiao, Q. Lu, P. Ren, R. Lin, Rendering optimization for mobile web 3d based on animation data separation and on-demand loading, IEEE Access 8 (2020) 88474–88486.

[4] Q. Liu, S. Huang, J. Opadere, T. Han, An edge network orchestrator for mobile augmented reality, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 756–764.

[5] X. Qiao, P. Ren, G. Nan, L. Liu, S. Dustdar, J. Chen, Mobile web augmented reality in 5g and beyond: Challenges, opportunities, and future directions, China Communications 16 (9) (2019) 141–154.

[6] S. Rattanarungrot, M. White, Development of service oriented mobile ar applications for museum learning activities, in: 2016 22nd International Conference on Virtual System & Multimedia (VSMM), IEEE, 2016, pp. 1–8.

[7] Z. Huang, W. Li, P. Hui, C. Peylo, Cloudridar: A cloud-based architecture for mobile augmented reality, in: Proceedings of the 2014 workshop on Mobile augmented reality and robotic technology-based systems, 2014, pp. 29–34.

[8] J. G. Herrera, J. F. Botero, Resource allocation in nfv: A comprehensive survey, IEEE Transactions on Network and Service Management 13 (3) (2016) 518–532.

[9] K. Kaur, V. Mangat, K. Kumar, A comprehensive survey of service function chain provisioning approaches in sdn and nfv architecture, Computer Science Review 38 (2020) 100298.

[10] G. Sun, G. Zhu, D. Liao, H. Yu, X. Du, M. Guizani, Cost-efficient service function chain orchestration for low-latency applications in nfv networks, IEEE Systems Journal 13 (4) (2018) 3877–3888.

[11] W. Zhang, B. Han, P. Hui, V. Gopalakrishnan, E. Zavesky, F. Qian, Cars: Collaborative augmented reality for socialization, in: Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications, 2018, pp. 25–30.

[12] C.-C. Wu, L.-P. Tung, C.-Y. Lin, B.-S. P. Lin, Y.-C. Tseng, On local cache management strategies for mobile augmented reality, in: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, IEEE, 2014, pp. 1–3.

[13] C. Bachhuber, A. S. Martinez, R. Pries, S. Eger, E. Steinbach, Edge cloud-based augmented reality, in: 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP), IEEE, 2019, pp. 1–6.

[14] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in neural information processing systems, 2015, pp. 91–99.

[15] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[16] Q. Liu, T. Han, Dare: Dynamic adaptive mobile augmented reality with edge computing, in: 2018 IEEE 26th International Conference on Network Protocols (ICNP), IEEE, 2018, pp. 1–11.

[17] Z. Huang, V. Friderikos, Proactive edge cloud optimization for mobile augmented reality applications, in: 2021 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2021, pp. 1–6.

[18] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, C.-H. Hsu, User allocation-aware edge cloud placement in mobile edge computing, Software: Practice and Experience 50 (5) (2020) 489–502.

[19] L. Liu, H. Li, M. Gruteser, Edge assisted real-time object detection for mobile augmented reality, in: The 25th Annual International Conference on Mobile Computing and Networking, 2019, pp. 1–16.

[20] P. Jain, J. Manweiler, R. Roy Choudhury, Overlay: Practical mobile augmented reality, in: Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services, 2015, pp. 331–344.

[21] Y.-J. Seo, J. Lee, J. Hwang, D. Niyato, H.-S. Park, J. K. Choi, A novel joint mobile cache and power management scheme

for energy-efficient mobile augmented reality service in mobile edge computing, IEEE Wireless Communications Letters 10 (5) (2021) 1061–1065.
[22] J. Ren, L. Gao, X. Wang, M. Ma, G. Qiu, H. Wang, J. Zheng, Z. Wang, Adaptive computation offloading for mobile augmented reality, Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 5 (4) (2021) 1–30.
[23] G. Zheng, A. Tsiopoulos, V. Friderikos, Optimal vnf chains management for proactive caching, IEEE Transactions on Wireless Communications 17 (10) (2018) 6735–6748.
[24] V. Cozzolino, L. Tonetto, N. Mohan, A. Y. Ding, J. Ott, Nimbus: Towards latency-energy efficient task offloading for ar services, IEEE Transactions on Cloud Computing (2022).
[25] G. Niu, Q. Chen, Learning an video frame-based face detection system for security fields, Journal of Visual Communication and Image Representation 55 (2018) 457–463.
[26] A. T. Naman, R. Xu, D. Taubman, Inter-frame prediction using motion hints, in: 2013 IEEE International Conference on Image Processing, IEEE, 2013, pp. 1792–1796.
[27] P. Patel, A. H. Ranabahu, A. P. Sheth, Service level agreement in cloud computing (2009).
[28] C. Sun, J. Zhou, J. Liuliang, J. Zhang, X. Zhang, W. Wang, Computation offloading with virtual resources management in mobile edge networks, in: 2018 IEEE 87th Vehicular Technology Conference (VTC Spring), IEEE, 2018, pp. 1–5.
[29] Y. Sun, Z. Chen, M. Tao, H. Liu, Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff, IEEE Transactions on Communications 67 (11) (2019) 7573–7586.
[30] T. Dang, M. Peng, Joint radio communication, caching, and computing design for mobile virtual reality delivery in fog radio access networks, IEEE Journal on Selected Areas in Communications 37 (7) (2019) 1594–1607.
[31] M. Liu, Y. Liu, Price-based distributed offloading for mobile-edge computing with computation capacity constraints, IEEE Wireless Communications Letters 7 (3) (2017) 420–423.
[32] Z. Liu, Y. Xiang, X. Qu, Towards optimal cpu frequency and different workload for multi-objective vm allocation, in: 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), IEEE, 2015, pp. 367–372.
[33] C. Wu, M. Wang, B. Ma, K. Chen, Heterogeneous networks topology optimization based on simulated annealing algorithm, in: 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Vol. 1, IEEE, 2020, pp. 2074–2078.
[34] F. Fausto, A. Reyna-Orta, E. Cuevas, Á. G. Andrade, M. Perez-Cisneros, From ants to whales: metaheuristics for all tastes, Artificial Intelligence Review 53 (1) (2020) 753–810.
[35] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.
[36] Y. Zuo, Y. Wu, G. Min, L. Cui, Learning-based network path planning for traffic engineering, Future Generation Computer Systems 92 (2019) 59–67.
[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The journal of machine learning research 15 (1) (2014) 1929–1958.
[38] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM 60 (6) (2017) 84–90.
[39] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proc. 13th international conference on artificial intelligence and statistics, 2010, pp. 249–256.
[40] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proc. IEEE intern. conference on computer vision, 2015.
[41] C. M. Bishop, Pattern recognition and machine learning, Springer, 2006.
[42] X. Yang, Z. Chen, K. Li, Y. Sun, N. Liu, W. Xie, Y. Zhao, Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff, IEEE Access 6 (2018) 16665–16677.
[43] H. Saarnisaari, A. O. Laiyemo, C. H. de Lima, Random access process analysis of 5g new radio based satellite links, in: 2019 IEEE 2nd 5G World Forum (5GWF), IEEE, 2019, pp. 654–658.
[44] G. Barb, M. Otesteanu, On the influence of delay spread in tdl and cdl channel models for downlink 5g mimo systems, in: 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE, 2019, pp. 0958–0962.
[45] X. Zhang, S. Fronz, N. Navab, Visual marker detection and decoding in ar systems: A comparative study, in: Proceedings. International Symposium on Mixed and Augmented Reality, IEEE, 2002, pp. 97–106.
[46] M. Speicher, B. D. Hall, A. Yu, B. Zhang, H. Zhang, J. Nebeling, M. Nebeling, Xd-ar: challenges and opportunities in cross-device augmented reality application development, Proceedings of the ACM on Human-Computer Interaction 2 (EICS) (2018) 1–24.
[47] H. Shu, L.-P. Chau, Frame size selection in video downsizing transcoding application, in: 2005 IEEE International Symposium on Circuits and Systems, IEEE, 2005, pp. 896–899.
[48] F. Perronnin, Y. Liu, J. Sánchez, H. Poirier, Large-scale image retrieval with compressed fisher vectors, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 3384–3391.
[49] K. Toczé, S. Nadjm-Tehrani, Orch: Distributed orchestration framework using mobile edge devices, in: 2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC), IEEE, 2019, pp. 1–10.
[50] C. Sonmez, A. Ozgovde, C. Ersoy, Fuzzy workload orchestration for edge computing, IEEE Transactions on Network and Service Management 16 (2) (2019) 769–782.