Field of View Aware Proactive Caching for Mobile Augmented Reality Applications

Zhaohui Huang and Vasilis Friderikos
Center of Telecommunication Research, King's College London, London, U.K.
E-mail: {zhaohui.huang, vasilis.friderikos} @kcl.ac.uk

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10000639

***Abstract*—Mobile Augmented Reality (MAR) applications require significant computational and storage resources at the end devices or at edge clouds (EC) to, inter alia, support for the Augmented Reality Objects (AROs) that are amalgamated to the physical world. In this work, a MAR service is considered under the lenses of microservices where MAR service components can be decomposed and anchored at different locations ranging from the end device to different ECs to optimize the overall service and network efficiency. The novel content-aware aspect of the proposed solution allows for proactive caching of high probability 2D field of views (FoVs) of the AROs to be stored instead of caching the significantly larger and complex 3D original AROs. To this end, a joint optimization scheme (Optim) considering mobility and the trade-off between delay, storage capacity and FoV allocation is proposed. A nominal Long Short Term Memory (LSTM) deep neural network is further explored to provide efficient pro-active decision making in real-time. More specifically, the LSTM deep neural network is trained with optimal solutions derived from a mathematical programming formulation in an offline manner. A set of numerical investigations reveal that optimal decisions manage to outperform recently proposed schemes by 24.4% to 67.5% in delay under different weights, whilst the LSTM deep neural network is effective in providing competitive solutions as well as being amenable for real-time decision making.**

***Index Terms*—5G, Augmented Reality, Field of View, Mobility, Long Short Term Memory, Wireless networks**

I. Introduction

DIFFERENT from traditional applications, Mobile Augmented reality (MAR) enhances virtual experiences through elevating capabilities of mobile devices and hence brings about significant allowing for amalgamating Augmented Reality Objects (AROs) with the physical world. Such applications have demanding requirements in computing and caching resources, especially when rendering 3-dimensional (3D) AROs [1]. To this end, Edge clouds (ECs) support where computational expensive processing and storage can be offloaded is vital in efficient deployment and operation of mobile augmented reality services. However, compared to the classical remote cloud, these edge nodes are typically equipped with low-power computational resources with limited computational capability. Therefore, efficient utilization of the edge cloud resources is of paramount importance.

The work in [2], outlines how a MAR application can be decomposed into a series of granular micro-services together with an optimization framework to provide optimal pro-active mobility-aware decision making in terms of EC assignment. As illustrated in Fig. (1), the captured video of a MAR application is preprocessed at the terminal and then frames with AR objects are transmitted for detection, extraction and recognition. The local cache is searched to find if there is a match. Finally, the matched results are transmitted back for presentation. According to their features, these MAR functions are categorized into two types: computational intensive ones that require significant CPU resources and storage intensive ones that require significant cache resources. Clearly, they have a predefined order when working as a service chain. When there is no mobility, it could be possible to deploy the MAR application at a single server. However, it is clear that, without decomposition, the complete MAR application uses significant levels of CPU and cache memory resources. On the other hand, decomposition allows a more flexible allocation but the proactive caching is still not necessary here because there is no target objects distributed in the area. When bringing in both AR objects and the user mobility, an isolated module could be designed to provide predictions for proactive resource allocation according to target AROs and the user's possible future destination [3]. Thus, the given prediction not only contains where to set functions and send requests, but also indicates what AR content should be allocated to the target server.

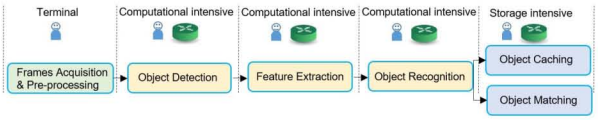
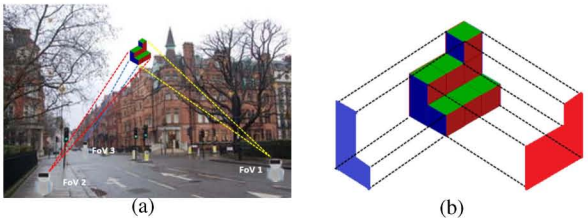


Fig. 1. The flow of the different nominal mobile AR functions, their characteristics and the potential location where those functions can run (terminal and/or at the EC).



(a) (b)

Fig. 2. Possible FoVs from different viewpoints on street (a) and Example FoVs of the target ARO (b)

As illustrated in Fig. (2), the different field of views (FoVs) can be achieved when viewing the target ARO from different positions and two such typical 2D projections of the target 3D ARO are presented as examples. Comparing with caching 3D AROs, which are usually in the order of tens of Mbytes in terms of size [4], caching their 2D counterparts from given FoVs can be another choice. These 2D FoVs are only several hundred KBytes in size and can also be easily reconstructed or conversed back (if needed) into 3D objects through algorithms like correlation-based, tracking-based and deep extraction [5] [6] [7]. In addition only few FoV can be deemed as of interest. For example the work in [8], collected and analyzed 295500 viewpoint samples from 153 volunteers. Subsequently, they splitted a 360° range into several groups; for example, in 90% of time, the X, Y and Z angles are within $-15.31° \sim 15.06°$, $-34.73° \sim 35.14°$, and $-8.61° \sim 8.62°$ [8]. This means the user prefer to observe the target in this angle which makes such FoV a popular one. In [8] [9] and [10], the probabilistic model are developed and learn from the historical data of FoVs. According to results in [8], when considering applying 2D figures as an alternative, it is reported that up to 80% bandwidth can be saved through caching only figures of popular FoVs. In addition, 3D AROs are more complex and the matching function requires a pose calculation stage for them. Thus, 3D AROs suffer from a higher computation complexity and require more computing resources. Although proactive caching 2D FoVs with high probability cannot always ensure the proper pose of the target ARO, it owns an obvious advantage in massively reducing caching and computing resources requirements. Clearly, more stored FoVs per ARO leads to a higher probability of a cache hit during matching but consumes more cache space and more processing delay. The differences between nominal caching and pro-active caching for MAR applications in this paper is further revealed by Fig. (3). Therefore, the joint optimization problem in this paper subjects to strict edge resources constraints and seeks the balance between service latency and allocation of FoVs in the mobility event. Constructing and solving a complex

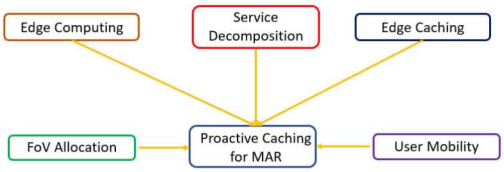


Fig. 3. Pro-active Caching for MAR

mixed integer linear problem is quite time-consuming and cannot respond to network changes in time. Therefore, the previous scheme should be improved to take into account both high-quality solutions and computational efficiency. Leveraging machine learning (ML) techniques on the network, such as neural networks or reinforcement learning, are considered to have the potential to solve resource allocation problems in different types of networks [11] [12] [13]. Thus, a well-known ML technique, long short term memory (LSTM), is applied in this paper to learn from optimal solutions and provide reliable predictions. LSTM is widely accepted as an enhanced recurrent neural network due to its ability in memorizing significant information and forgetting unnecessary inferences to overcome potential gradient vanishing and gradient exploding problems [14] [15]. As pointed out by our previous work [2], the execution sequence of two main types of functions is pre-defined during the decomposition of MAR services. Therefore, the correct order of the path could be figured out whenever the assignment is provided. An extra feasibility check stage is added to ensure the predictions can still satisfy constraints and fit the original network environment.

As discussed earlier, the obvious advantage in saving edge resources when caching 2D FoVs motivates our research in its optimization problem. The mathematical programming formulation consists of the service latency based on [2], the size and allocation of pre-cached 2D FoVs; in that case, an edge cloud resource allocation joint optimization framework is proposed to proactively allocate resources and to satisfy related requirements of MAR applications. The user mobility is considered explicitly and the overall delay is minimized as a multi-objective optimization problem. Although the optimal solution is desirable however such a framework cannot be used to provide real-time decision making since solving a mixed integer mathematical program suffers from the curse of dimensionality which means that requires significant amount of time to provide a solution. Therefore, in this paper, an LSTM-based approach is further explored where the optimal solutions are used to train the deep neural network in an offline manner.

II. System Model

A set $\mathbb{M} = \{1, 2, ..., M\}$ is defined to represent the available ECs in the network. Multiple mobile devices are assumed to create MAR service requests $r \in R$. With η and ϱ we define computational intensive and storage intensive MAR functionalities respectively [2]. The allowed destinations for a user are limited to adjacent access routers. Thus, The probability of a user moving from the initial location to an adjacent server can be defined as $p_{rij} \in [0, 1]$, where adjacent servers $\{i, j\} \subset \mathbb{M}$. The location of executing one functionality for a request can be denoted as $L_{\eta r} \in \mathbb{M}$ or $L_{\varrho r} \in \mathbb{M}$. Also, in a typical MAR service, each request requires the execution of two functionalities in a pre-defined order [2]. Thus, the assignment of one request can be arranged according to the route of its flow. The route matrix can be denoted as $RT_r = [s_r, L_{\eta r}, L_{\varrho r}, d_r]$. This matrix can be provided by the optimal scheme and consist a route assignment set $\mathbb{RT}$ [2]. In LSTM training, requests' initial locations and their destinations can be fed as inputs and decisions of where to anchor MAR functionalities can be treated as the output. Thus, we further split the route matrix into $X_r = [s_r, d_r] \in \mathbb{X}$ and $Y_r = [L_{\eta r}, L_{\varrho r}] \in \mathbb{Y}$. The major part of the route set $\mathbb{RT}$ is used as input and fed into the LSTM-based model for training. The rest of this set remains as a testing

set, Therefore, we denote with $\mathbb{XTrain}, \mathbb{XTest} \subset \mathbb{X}$ and $\mathbb{YTrain}, \mathbb{YTest} \subset \mathbb{Y}$ to differentiate training and testing sets. The output is the predictions made by the trained model and hence denoted as $\mathbb{YPred}$.

At first, a joint optimization scheme considering the service delay, the total size and allocation of pre-cached FoVs is designed to track the MAR requests in the network. Denote $p \in \mathbb{P}$ as a 2D FoV to observe an 3D ARO. Then the decision variable h_{rlj} in [2] can be rewritten as h^p_{rlj} that decides whether to pre-cache the FoV p of the ARO l required by the request r at the location j.

$$h^p_{rlj} = \begin{cases} 1, \text{ if the FoV } p \text{ of an ARO } l \text{ required} \\ \quad \text{by the request } r \text{ is cached at node } j, \\ 0, \text{ otherwise.} \end{cases} \tag{1}$$

With extra constraints,

$$\sum_{j\in M}\sum_{l\in N}\sum_{p\in P} h^p_{rlj} \geqslant 1, \ \forall r \in R \tag{2}$$

$$\sum_{r\in R}\sum_{l\in N}\sum_{p\in P} h^p_{rlj} \leqslant 1, \ \forall j \in M \tag{3}$$

(2) means at least one FoV of an ARO is required by each request. (3) means even the same FoV is required by different requests at the same location, it only needs to be pre-cached once. Then the decision variable for cache hit/miss can be written as,

$$z_{rj} = \begin{cases} 1, \text{ if} \sum_{l\in N}\sum_{p\in P} h^p_{rlj} \geqslant L_r, \\ 0, \text{ otherwise.} \end{cases} \tag{4}$$

The constraints (19b) and (19d) in [2] that reveal the cache limitation and the cache hit/miss relation can be updated as,

$$\sum_{r\in \mathbf{R}}\sum_{l\in \mathbf{L_r}}\sum_{p\in \mathbf{P}} h^p_{rlj} l^p_{rj} \leq \Theta_j, \forall j \in \mathbf{M} \tag{5}$$

$$\sum_{l\in \mathbf{N}}\sum_{p\in \mathbf{P}} h^p_{rlj} + \epsilon \leq L_r + U(1-q_j) \ \forall j \in \mathbf{M}, r \in \mathbf{R} \tag{6}$$

where ϵ is a small positive real number [2]. More pre-cached FoVs lead to an extra burden for processing the matching function. Hence, denote the size of a FoV p of an ARO l at location j as l^p_{rj}. The processing delay of the matching function can be written as,

$$W_{rj} = \frac{\omega(F_{\varrho r} + \sum_{l\in N}\sum_{p\in P} h^p_{rlj} l^p_{rj})}{f^j_V} \tag{7}$$

Where ω is the computation load and $F_{\varrho r}$ is the extracted features defined in [2]. Noticing that when processing the matching function at the location j ($W_{rj} y_{rj}$), the production of decision variables $h^p_{rlj} y_{rj}$ appears inside and can be handled through bringing in a new decision variable α^p_{rlj} with following constraints,

$$\alpha^p_{rlj} = h^p_{rlj} y_{rj} \tag{8a}$$

$$s.t. \alpha^p_{rlj} \leqslant h^p_{rlj} \tag{8b}$$

$$\alpha^p_{rlj} \leqslant y_{rj} \tag{8c}$$

$$\alpha^p_{rlj} \geqslant h^p_{rlj} + y_{rj} - 1 \tag{8d}$$

The service latency L maintains the same as (19a) in [2] and L_{max} here means the maximum possible latency. Thus, we have,

$$\frac{L}{L_{max}} \in [0, 1] \tag{9}$$

The total size of pre-cached 2D figures can be written as,

$$S = \sum_{r\in R}\sum_{j\in M}\sum_{l\in N}\sum_{p\in P} h^p_{rlj} l^p_{rj} \tag{10}$$

S_{max} here means the maximum allowable size of pre-cached FoVs. The allocation of FoVs of an ARO at a location is $\sum_{r\in R}\sum_{p\in P} h^p_{rlj}$. Denote $t^p_{lj} \in \mathbb{T}_{\mathbb{lj}}$ as the probability of viewing a certain FoV at the location. Then the summary reveals the quality of FoV allocation and can be written as,

$$A = \sum_{j\in M}\sum_{l\in N}\sum_{r\in R}\sum_{p\in P} h^p_{rlj} t^p_{lj} \tag{11}$$

A_{max} here means the sum of probabilities of all available FoVs of target AROs. As mentioned earlier, if the cache space is limited, pre-caching several FoVs with higher probabilities from multiple AROs is better in achieving a better quality than storing the same number of FoVs but from only several AROs. In addition, $\mathbb{T}_{\mathbb{lj}}$ is defined as a descending probability vector and t^p_{lj} inside can be achieved through,

$$t^p_{lj} = \begin{cases} 1, \text{ if } p = 1, \\ \text{rand } (t^{p-1}_{lj}), \text{ if } p > 1. \end{cases} \tag{12}$$

Where the function rand(x) means generating a random number within $[0, x]$. Then to limit the sum of probabilities of an ARO's different FoVs as 1, the normalization should be done and each element t^p_{lj} in the vector can be updated by $\frac{t^p_{lj}}{\sum_{p\in P} t^p_{lj}}$.

Therefore, the formula capturing both size and allocation can be written as,

$$\frac{1}{2}\left(\frac{S}{S_{max}} + \frac{A}{A_{max}}\right) \in [0, 1] \tag{13}$$

Denote the weight parameter as $\mu \in [0, 1]$, the joint optimization problem can eventually be written as,

$$min \ \mu\frac{L}{L_{max}} - \frac{1-\mu}{2}\left(\frac{S}{S_{max}} + \frac{A}{A_{max}}\right) \tag{14a}$$

$$\text{s.t. } z_{rj} = 1 - q_j, \ \forall j \in \mathbf{M}, r \in \mathbf{R} \tag{14b}$$

$$\sum_{r\in \mathbf{R}} (x_{rj} + y_{rj}) \leq \Delta_j, \forall j \in \mathbf{M} \tag{14c}$$

$$\sum_{j\in \mathbf{M}} x_{rj} = 1, \forall r \in \mathbf{R} \tag{14d}$$

$$\sum_{j\in \mathbf{M}} y_{rj} = 1, \forall r \in \mathbf{R} \tag{14e}$$

$$\xi_{rij} \leq x_{ri},\ \forall r \in \mathbf{R}, i, j \in \mathbf{M} \quad (14f)$$
$$\xi_{rij} \leq y_{rj},\ \forall r \in \mathbf{R}, i, j \in \mathbf{M} \quad (14g)$$
$$\xi_{rij} \geq x_{ri} + y_{rj} - 1,\ \forall r \in \mathbf{R}, i, j \in \mathbf{M} \quad (14h)$$
$$\psi_{rj} \leq z_{rj},\ \forall r \in \mathbf{R}, j \in \mathbf{M} \quad (14i)$$
$$\psi_{rj} \leq y_{rj},\ \forall r \in \mathbf{R}, j \in \mathbf{M} \quad (14j)$$
$$\psi_{rj} \geq z_{rj} + y_{rj} - 1,\ \forall r \in \mathbf{R}, j \in \mathbf{M} \quad (14k)$$
$$x_{rj}, y_{rj}, h^{p}_{rlj}, z_{rj}, q_j, \alpha^{p}_{rlj}, \psi_{rj}, \xi_{rij} \in \{0, 1\},$$
$$\forall r \in \mathbf{R}, j \in \mathbf{M}, l \in \mathbf{L}, p \in \mathbf{P} \quad (14l)$$
(2), (3), (5), (6)
(8b), (8c), (8d)

The constraint (14b) together with updated constraints (5) and (6) reveal the relation between pre-caching decisions and the cache miss/hit for each request [2]. The constraint (14c) is the cache limitation while (14d) and(14e) forces the once execution of each function of a request at a single server as explained in [2]. The constraints (8b), (8c), (8d) and (14f) to (14k) are brought in to solve products of decision variables for linearization.

The optimal assignments generated by the aforementioned ILP model are grouped and reshaped into a route matrix and then separated for training and testing. Since execution locations for functionalities can only be selected from the set $\mathbb{M}$, all possible different results can be calculated as selecting any two from M elements with order, which is M^2. For simplicity, each assignment can be given an index to show which type it belongs to. The LSTM-based model will then classify and create a mapping between the two input sets. After training, the model will provide its predictions according to the input training set. The predictions are checked for feasibility in terms if constraints such as cache and EC capacity limit are satisfied. If for an EC capacity is reached, remaining predicted requests will be allocated to an available neighbor EC as a backup choice. However, if both ECs are occupied, then the request is sent to a cloud deeper in the network and, as a result, an extra penalty is triggered. In order to shed light into the quality of the decision making, the assignments are compared with the optimal assignments to evaluate the quality of the LSTM-based scheme. The architecture of the LSTM-based model follows the nominal design and its state changing and gate controlling follows the original formula. The aim is to explore the potential of using a nominal LSTM network for service decomposition.

III. Numerical Investigations

Hereafter, the effectiveness of the proposed Optim and LSTM-based schemes are investigated via a wide-set of simulations and compared with previously proposed schemes.

We assume a typical tree-like network topology [2], with 20 ECs in total with 4 to 8 ECs being activated and up to 40 requests are sent by MAR devices. The remaining available resources allocated for MAR support of an EC is assumed to be CPUs with 4 to 8 cores and 16GB memory [2]. Each request requires a single free unit for each service function (for example a VM) and each 2D FoV of the target ARO is set from 0.1MB to 2MB to cover different use cases [16] [5] [6] [7]. Each ARO is assumed to have up to 4 different FoVs. There are 14 available VMs in each EC, with equal splitting of CPU resources [2]. Furthermore, available remaining storage for each EC is assumed to range between 100 to 500 Mbytes for supporting MAR services. As eluded previously, users move between adjacent access routers from the initial location. Optimal solutions are calculated based on the proposed Optim scheme and a set of solutions are used to feed the LSTM neural network. From that set, 90% is fed into the model as a training set while the rest 10% is used for testing. In the LSTM layer, the initial learning rate is set to 0.005 and the maximum number of epochs is 160. The probability in the dropout layer is set to 5% to avoid over-fitting.

The proposed optimization framework (Optim) provides the optimal solutions by solving the underlying MILP problems [2]. The Mobility Oblivious Allocation scheme (MOA) finds optimal solutions but without taking the user's mobility effect into account [2]. The random selection scheme (RandS) selects the target servers randomly whilst the closest-first scheme (CFS) tend to choose the nearest available EC to the user's initial location [17]. The utilization scheme (UTIL) takes the least loaded neighbor EC as a backup if the nearest one excesses an availability threshold status (for example, more than 80% of total VMs are occupied) [18]. All these baseline schemes are FoV-oblivious which means storing all FoVs without considering their probability. As shown by Fig. (4), the weighted fraction of stored FoVs' probability per ARO drops as expected when the weight μ increases. Table (I) shows the weighted fraction of stored FoVs' size per ARO under different weights. The value of such factor drops from 0.81 (over 3 FoVs per ARO) to 0.28 (slightly over 1 FoV per ARO) and hence the Optim scheme can save much storage resources as expected.

TABLE I
Weighted Fraction of Stored FoVs' Size per ARO under Different Weights (μ)

Weight	**0**	**0.2**	**0.5**	**0.8**	**1**
$\frac{S}{S_{max}}$	0.81	0.62	0.46	0.34	0.28

According to Fig. (5), the proposed Optim scheme has the least delay whilst depicting a desired insensitivity as the weight increases. The gap between the Optim scheme ($\mu = 0$) and other baseline schemes increase with the number of requests and hence the Optim scheme owns an obvious advantage in a more congested network. The greedy schemes share a common tendency to execute decomposed MAR services on a few ECs. The flexibility of decomposition is reduced and hence these greedy schemes can suffer a penalty cost because of the narrow space and limited resources. However, the gains in delay between Optim schemes with different weights are becoming narrow. Compared to the CFS scheme, when $\mu = 0$,

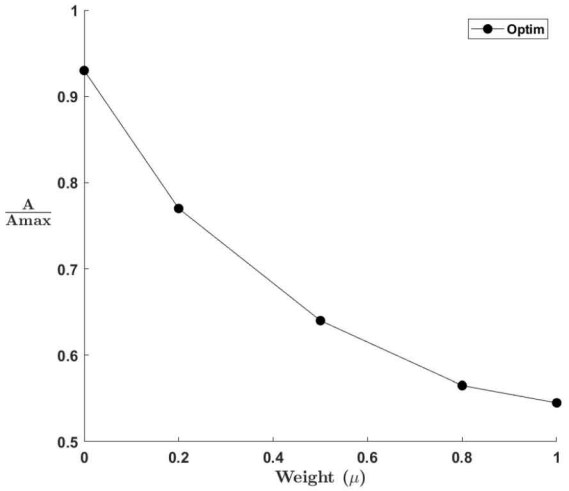


Fig. 4. Weighted Fraction of Stored FoVs' Probability per ARO under Different Weights (μ)

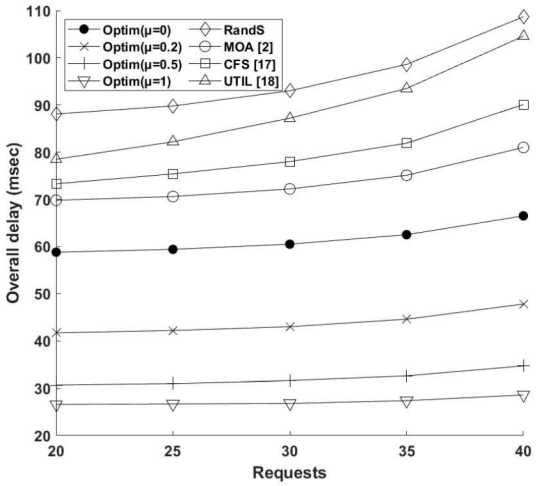


Fig. 5. Average service delay of different schemes under different numbers of requests (6 ECs and Capacity is 14)

the Optim scheme has a similar weighted fraction of stored FoVs' probability per ARO but can save around 24.4% in terms of delay. When $\mu = 0.2$, the Optim scheme reduce A/A_{max} by 16% to achieve around 47% less delay than the CFS scheme. When $\mu = 0.5$, A/A_{max} reduced by 29% but for 62% less delay. In an extreme condition that $\mu = 1$, the Optim scheme saves 67.5% delay but has a 39% loss in the weighted fraction of stored FoVs' probability per ARO. Hence, a balance between delay and amount of FoV cache should be carefully considered. This is also supported by the narrowing gap between weighted fraction of stored FoVs' size per ARO according to Table (I). According to Table (II), the LSTM-based scheme can follow the Optim scheme closely and performs better when focusing on delay. According to Table (III), when there is no mobility and the network is not in a congested state, different schemes perform similarly. The Optim scheme shares the same performance as the MOA scheme and is also close to the CFS scheme. The LSTM-based scheme is only 1.8ms worse than the Optim scheme. Compared with other baseline schemes, its performance is still acceptable in this case.

TABLE II
DELAY (MS) FOR LSTM AND OPTIM UNDER DIFFERENT WEIGHTS

Weight	0	0.2	0.5	0.8	1
LSTM	64.5	45.4	34.1	29.3	27.1
Optim	60.3	42.8	32.5	28.1	26.2

TABLE III
RMSE VALUES AND DELAY IN NO MOBILITY EVENT ($\mu = 1$, 6 ECS, 30 REQUESTS AND CAPACITY IS 14)

Scheme	Optim	LSTM	MOA	CFS	RandS	UTIL
Delay (ms)	23.5	25.3	23.5	24.2	28.1	25.6
RMSE	-	1.5	7.0	2.6	2.1	4.4

When comparing predictions and optimal solutions, we also have $rmse < 1.6$ (root mean square error), $\delta < 14\%$ (relative error) and $r^2 > 0.83$ (determination coefficient) in most cases, which indicate the predictions can be deemed as reliable and highly competitive to the optimal solutions. Small values of $rmse$ and δ indicates that the proposed scheme generates a very similar version to the Optim's solution. $rmse$ values compared to the Optim scheme are also shown by Table (III). We note that the indexes have a location information, i.e., access routers which are topologically close have also adjacent indices. In addition, the user mobility is limited to adjacent EC within each period which is a valid assumption for active MAR sessions. Thus, high similarity to the optimal solutions usually indicates that the corresponding scheme has the ability to provide high-quality solutions as well. The value of r^2 is close to 1 and hence the proposed model shows a high quality fitting ability in this problem. Clearly, the quality will deteriorate when the output solution lead to an infeasible allocation. In that case, requests will have to be served by a cloud server which will entail some penalty. Experimentation showed that this was rarely the case during the training set but tends to appear in the testing set. The validation accuracy increases with iterations and finally maintains stable at around 94.3%.

Although obtaining optimal solutions and training the LSTM model based on the Optim scheme can be time-consuming, the inference stage itself is highly efficient. Also, the time required to train the network dominated by the time to provide the optimal solutions as shown in Fig. (6) (averaged by 50 executions). The average processing time of the inference stage is compared with other algorithms in Table (IV) (executed once). Considering the fact that optimal solutions obtaining and network training stage only execute offline, the inference stage has a more dominant effect. Other greedy schemes need to check and handle different constraints (e.g. capacity and cache

size) repeatedly in iterations and lead to a longer processing time. In the Optim scheme, a complex MILP problem is constructed and solved to find optimal solutions and hence is the most time-consuming one. Thus, the inference stage of the proposed scheme is the most efficient one among all schemes. This is of a significant importance for network operation since it achieves better performance and takes much less time to provide the decision making than other algorithms.

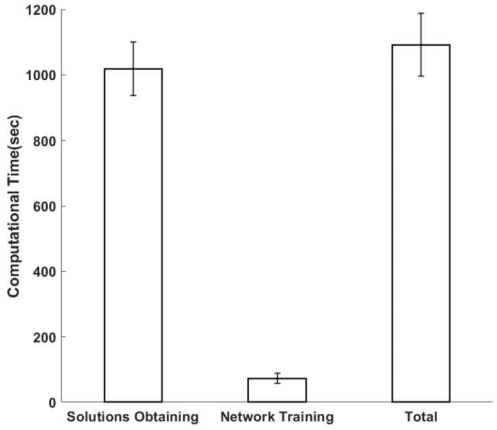


Fig. 6. Average execution time of the training phase (6 ECs, 30 Requests and Capacity is 14)

TABLE IV
AVERAGE 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
LSTM	0.427	0.065
Optim [2]	201.506	21.965

*tested by PC, intel i7, 6500U, 2 cores

IV. CONCLUSIONS

Mobile Augmented Reality (MAR) applications are sensitive to the user mobility and service delay. Service decomposition together with edge clouds allow for a more flexible resource allocation to better tackle the mobility event with AR objects for MAR applications. Enabling content-aware caching by storing high probability 2D FoVs instead of 3D AROs could significantly ease the requirements for storage and computing resources with limited effects on overall quality. To this end, this work proposes a joint optimization framework considering the balance between service delay, field-of-view aware proactive caching and FoV allocation. Furthermore, an LSTM-based deep neural network is explored to provide real time decision making. The LSTM network is trained using optimal solutions offline, and during inference can efficiently provide high-quality decision making in real-time. This is validated by a series of simulations showing that the proposed scheme outperforms previously proposed solutions. Future avenues of research will articulate on the use of advanced meta-heuristic algorithms to provide near-optimal decision making for training the deep neural network in large network instances where optimal decision making is not efficient.

REFERENCES

[1] L. Li, X. Qiao, Q. Lu, P. Ren, and R. Lin, "Rendering optimization for mobile web 3d based on animation data separation and on-demand loading," *IEEE Access*, vol. 8, pp. 88 474–88 486, 2020.

[2] Z. Huang and 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.

[3] D. Alencar, C. Both, R. Antunes, H. Oliveira, E. Cerqueira, and D. Rosário, "Dynamic microservice allocation for virtual reality distribution with qoe support," *IEEE Transactions on Network and Service Management*, 2021.

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

[5] K. Patoommakesorn, F. Vignat, and F. Villeneuve, "The 3d edge reconstruction from 2d image by using correlation based algorithm," in *2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)*. IEEE, 2019, pp. 372–376.

[6] I. Tsubaki, A. Shimeno, T. Tsukuba, T. Suenaga, and M. Shioi, "2d to 3d conversion based on tracking both vanishing point and objects," in *The 1st IEEE Global Conference on Consumer Electronics 2012*. IEEE, 2012, pp. 110–114.

[7] X. Huang, L. Wang, J. Huang, D. Li, and M. Zhang, "A depth extraction method based on motion and geometry for 2d to 3d conversion," in *2009 Third International Symposium on Intelligent Information Technology Application*, vol. 3. IEEE, 2009, pp. 294–298.

[8] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 1161–1170.

[9] N. Kan, J. Zou, K. Tang, C. Li, N. Liu, and H. Xiong, "Deep reinforcement learning-based rate adaptation for adaptive 360-degree video streaming," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 4030–4034.

[10] L. Zhao, Y. Cui, Z. Liu, Y. Zhang, and S. Yang, "Adaptive streaming of 360 videos with perfect, imperfect, and unknown fov viewing probabilities in wireless networks," *IEEE Transactions on Image Processing*, 2021.

[11] R. Bianchini, M. Fontoura, E. Cortez, A. Bonde, A. Muzio, A.-M. Constantin, T. Moscibroda, G. Magalhaes, G. Bablani, and M. Russinovich, "Toward ml-centric cloud platforms," *Communications of the ACM*, vol. 63, no. 2, pp. 50–59, 2020.

[12] S. B. Nath, S. Chattopadhyay, R. Karmakar, S. K. Addya, S. Chakraborty, and S. K. Ghosh, "Ptc: Pick-test-choose to place containerized microservices in iot," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[13] N. Jalodia, S. Henna, and A. Davy, "Deep reinforcement learning for topology-aware vnf resource prediction in nfv environments," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2019, pp. 1–5.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] Y. Zuo, Y. Wu, G. Min, and L. Cui, "Learning-based network path planning for traffic engineering," *Future Generation Computer Systems*, vol. 92, pp. 59–67, 2019.

[16] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *IEEE INFOCOM 2018*, 2018, pp. 756–764.

[17] K. Toczé and S. Nadjm-Tehrani, "Orch: Distributed orchestration framework using mobile edge devices," in *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*, 2019, pp. 1–10.

[18] C. Sonmez, A. Ozgovde, and C. Ersoy, "Fuzzy workload orchestration for edge computing," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 769–782, 2019.