Service-Oriented MEC Applications Placement in a Federated Edge Cloud Architecture

Bouziane Brik[‡], Pantelis A. Frangoudis[§], and Adlen Ksentini[‡]
[‡]EURECOM, Sophia Antipolis, France
[§]Distributed Systems Group, TU Wien, Vienna, Austria
Email: [‡]name.surname@eurecom.fr, [§]pantelis.frangoudis@dsg.tuwien.ac.at

***Abstract*—Multi-access Edge Computing (MEC) is one of the key enablers in 5G, where the objective is to bring computation very close to the end users. MEC, as defined by ETSI, introduces several services that can be exposed to MEC applications regarding the mobile users, such as the Radio Network Information Service (RNIS) and the Location Service, which provide low-level information on mobile users (e.g., Channel Quality Indicator - CQI), allowing the development of context-aware edge applications. In this paper, we address the challenging question of where to deploy a set of MEC applications on a federated edge infrastructure so as to meet the applications' requirements in terms of computing resources and latency, while ensuring that the MEC platform services required by each application are available at the selected edge locations. We formulate this service placement problem as an Integer Linear Program, which aims at balancing the computing load between available Mobile Edge Platforms (MEP), while respecting application latency and MEP service availability constraints. This problem is shown to be NP-hard. To solve it computationally efficiently, we propose an algorithm based on the Tabu-Search (TS) meta-heuristic. Via simulation, we demonstrate the efficiency of our scheme in balancing computational load among available MEPs and its ability to optimize service placement.**

***Index Terms*—5G, Multi-access Edge Computing (MEC), MEC application placement, MEC Orchestrator, MEC service**

I. Introduction

5G is expected to support a huge number of new services related to different vertical industries [1]. These services are coming with different requirements in terms of network performance, such as access latency, bandwidth, communication reliability, the support for massive numbers of devices, etc. Among these new services, Ultra Reliable Low Latency Communications (URLLC) are the most challenging to support, as they need reliability with very low-latency access (around $1ms$ at the Radio Access Network - RAN) [2]. Many efforts have been put on leveraging the RAN in 5G to guarantee low latency, such as the reduction of Transmission Time Interval (TTI) to $0.5ms$ or less, and the usage of small packets for uRLLC services. However, this may not be not enough if the remote application (connected to the URLLC users) is deployed at remote centralized clouds. In this context, Multi-access Edge Computing (MEC) is considered as a complementary solution to sustain low latency for critical URLLC services [3]. In addition to providing an execution environment for running applications at the edge, MEC exposes services that provide information on end user and base station (eNodeB) context, such as the radio channel quality of users and their location in the network, allowing 3^{rd}-parties to build context-aware applications. MEC services are available by the MEC platform (MEP), which is connected to a set of eNodeBs and an Edge Virtualization Computing Infrastructure (EVCI). In a mobile network, one MEP/EVCI may cover a set of eNodeBs according to the operator's policy. Furthermore, a MEC application is run as a Virtual Machine (VM) or a container on top of the EVCI, similarly to a Virtual Network Function (VNF). The main difference is that a MEC application specifies in its descriptor (AppD) [4] some MEC-specific fields, such as the maximum tolerated latency, the set of required MEC platform services, traffic rules that allow to redirect the traffic to the MEC application, and the preferred deployment location.

This work has been partially supported by the European Union's H2020 5G-Transformer (grant no. 761536) and 5G!Drones (grant no. 857031) projects.

When deploying a MEC application at the edge, the MEC orchestrator (MEO) has to select the optimal MEP/EVCI components that satisfy a number of criteria, such as latency, capacity (CPU, storage, etc.) and the availability of specific MEC services.

However, some MEP/EVCIs may not be able to host a MEC application, as: (i) the EVCI computation capacity is limited compared to the centralized cloud; (ii) the latency to access the user data plane from the EVCI may not be respecting the stringent application constraint; (iii) the MEP does not offer one of the MEC services required by the MEC application. Therefore, the challenging question that is raised is where to execute MEC applications so as to better meet their application requirements in terms of computing resources, latency and MEC service availability, at the same time serving as many requests as possible.

In this paper, our focus is mainly on the deployment of MEC applications in line with the ETSI MEC model [5]. We propose a new deployment algorithm that decides the placement of a MEC application among the available MEP/EVCIs based on the requirements extracted from each MEC application's descriptor, i.e., AppD, ensuring that the proposed solution is fully ETSI MEC-compliant. Our algorithm aims to balance the computing load between the available MEP/EVCIs, while ensuring the constraints posed by

MEC. It relies essentially on an Integer Linear Programming formulation and heuristic solutions. Our contributions are summarized as follows:

- We study a MEC applications placement problem in line with the MEC ETSI architecture. The problem is formulated as an Integer Linear Program (ILP) whose objective is to balance the computing load among MEP/EVCIs.
- We show the problem to be NP-hard and propose a new Tabu Search-based (TS) algorithm to find near-optimal solutions. This is important when the number of MEC applications and MEP/EVCIs is large and exact solutions are computationally expensive. We also resolve our ILP using Python's PuLP optimization package [6].
- We demonstrate the performance of the proposed algorithm via simulation, showing our scheme to succeed in balancing the load among the existing MEP/EVCIs while ensuring MEC applications' requirements.

The remainder of this paper is organized as follows. § II gives a literature review about MEC application placement. § III presents an overview on the MEC architecture as defined by ETSI. § IV describes our application placement scheme. We discuss the obtained experimental results in § V and conclude the paper in § VI.

II. Related Work

Only few works have addressed the problem of the placement of MEC applications at the Edge Cloud. Indeed, most of the existing works have proposed algorithms in the context of mixed edge and central cloud such as [7] [8], [9]. Usually, VNFs are placed either in physical machines within central cloud infrastructures only [10], or in MEC servers [11], [12]. Other works [13], [14] investigate the placement in the context of a hybrid federated cloud, using a combination of different cloud infrastructures, without however involving the edge. In this context, the authors in [9] also propose a VNF placement solution that captures the trade-off between cost efficiency and Quality of Experience (QoE) in a multi-cloud environment. However, all these works do not consider MEC placement in a full federated edge architecture where the availability of MEC services is as important as latency to allow context-aware MEC applications. Moreover, these solutions are not fully compliant with the ETSI MEC architecture.

III. MEC architecture

Since its creation in 2013, the ETSI ISG MEC group has been working on the development of standardization activities around MEC. The first released document of the group covers the reference architecture [5], which aims to specify the different necessary components. It introduces three main entities:

- The MEC host or EVCI, which provides the virtualization environment to run MEC applications, while interacting with mobile network entities via the MEC platform (MEP) to provide MEC services and data offload to MEC applications. Two EVCIs can communicate together aiming at managing user mobility via the migration of MEC applications among MEC hosts.
- MEP acts as an interface between the mobile network and the MEC applications. It has an interface (Mp1) with MEC applications, so that the latter can expose and consume MEC services, and another interface (Mp2) to interact with the mobile network. The latter is used to obtain statistics from the RAN on UEs and eNBs, e.g., in order to provide the Radio Network Information Service (RNIS) and the Location Service, and to appropriately steer user-plane traffic to MEC applications.
- MEC applications that run on top of a virtualized platform.

Another concept introduced by ETSI MEC is the MEC service, which is either a service provided natively by the MEC platform, such as the RNIS and traffic control, or a service provided by a MEC application, e.g., video transcoding. MEC services provided by third-party MEC applications should be registered with the MEP and made available over the Mp1 reference point. Once registered, a service may be discovered and consumed by other MEC applications. Regarding the management plane, ETSI MEC introduced the Mobile Edge Orchestrator (MEO), which is in charge of the life-cycle of MEC applications (instantiation, orchestration and management), and acts as the interface between the MEC host and the Operations/Business Support System (OSS/BSS). The MEO is also in charge of the placement of MEC applications on the appropriate MEP/EVCI by executing a placement algorithm, which relies on the latency and region information to decide if a MEC application has to be placed at a MEP/EVCI in a specific region. In this paper, we assume that the MEO has a global picture on the availability of virtual resources that can satisfy the latency requirements of a MEC application in a specific region or location, as well as the availability of a MEC service on the MEP covering that region/location.

IV. MEC Applications Placement Scheme

A. MEC Applications Placement Formulation

We assume that each region is covered by several MEPs and EVCIs. Each MEP is associated with only one EVCI. Each region may correspond to a set of Tracking Areas (TA), as defined by 3GPP. Each TA includes several eNBs. Fig. 1 depicts the envisioned architecture.

We assume that E MEC services exist based on the ETSI MEC specifications. These include the Radio Network Information Service (RNIS), traffic redirection, DNS, etc. We note by $Ms_i = \{s_1, s_2, \ldots, s_n\}$ the set of MEC services supported by MEP_i, Avg_delay_i the average delay of a MEP_i to access the user data plane (i.e., eNodeB), and C_i the capacity in terms of CPU of the EVCI associated to the MEP_i. We assume that MEO receives a request to deploy a set of MEC applications from the OSS/BSS in a specific region. The MEC application is described by the

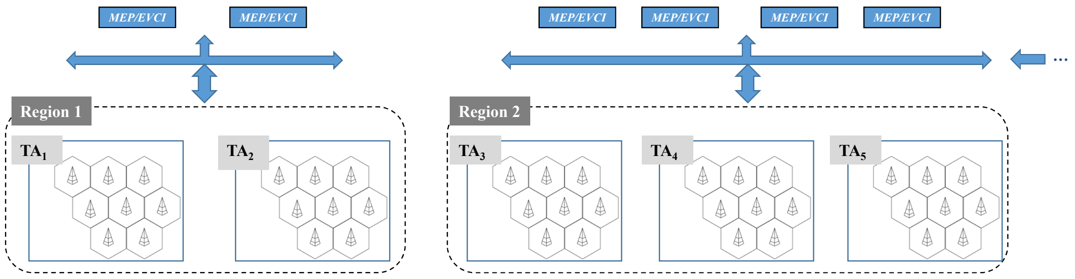

Fig. 1: Envisioned system architecture.

AppD. We note l, r and $\{s_1, s_2, \ldots, s_n\}$ as the maximum tolerated latency by the MEC application, the requested resources in terms of CPU, and the requested MEC services, respectively. All this information is derived from the AppD, which is communicated by the OSS/BSS when requesting the onboarding and instantiation of a MEC application. Indeed, the AppD includes fields that indicate these variables, such as `appServiceRequired`, `appLatency`, and `virtualComputeDescriptor`, while location information can be encoded in the `userDefinedData` field or specified in the `selectedMEHostInfo` field of the instantiation request. The proposed model aims to select the MEP/EVCI, where to deploy each MEC application. To do so, we propose the following formulation:

$$\begin{aligned}
&Min \sum_{i=1}^{i=N-1} \sum_{k=i+1}^{k=N} (| \sum_{j=1}^{j=M} X(i,j) * r_j - \\
&\sum_{j=1}^{j=M} X(k,j) * r_j |) \\
&\sum_{i=1}^{i=N} X(i,j) = 1, \ \forall j = 1, \ldots, M \\
&\sum_{i=1}^{i=N} \sum_{j=1}^{j=M} X(i,j) * r_j \leq C_i \\
&\forall i \in \{1, N\}, j \in \{1, M\}, k \in \{1, E\}, X(i,j) * S(j,k) = \\
&Ms_i(k) \\
&\forall i \in \{1, N\}, j \in \{1, M\}, X(i,j) * Avg\_delay(i) \leq l(j) \\
&\forall i \in \{1, N\}, j \in \{1, M\}, X(i,j) \in \{0,1\}
\end{aligned} \tag{1}$$

$X(i,j)$ is a binary decision matrix, where $X(i,j) = 1$ if application j is hosted at the EVCI i; otherwise $X(i,j)$=0. The vector $l(j)$ represents the maximum latency supported by the MEC applications. The matrix S corresponds to the MEC services required by the MEC applications. $S(i,k)$=1 means that application i requires MEC service k.The objective function aims to balance the load among the MEP/EVCIs covering a geographical area. The first constraint ensures that a MEC application is placed in one and only one MEP/EVCI. The second constraint guarantees that the capacity in terms of CPU of an EVCI is not exceeded. The third constraint aims to ensure that a MEC application is hosted at a MEP/EVCI that provides the required MEC services. The fourth constraint ensures that a MEC application is placed at a MEP/EVCI, which does not violate its latency constraint. The last constraint ensures that the variable $X(i,j)$ is binary. In fact, the MEC application placement problem (1) is NP-hard where the proof can be done by reduction from the partition problem, which is known to be NP-complete.
To remove absolute value in the objective function, least absolute deviation is employed; it adds two new constraints to the problem, which now becomes:

$$\begin{aligned}
&Min \sum_{i}^{i=N-1} \sum_{k=i+1}^{k=N} T_j(i,k) \\
&\sum_{i=1}^{i=N} X(i,j) = 1, \ \forall j = 1, \ldots, M \\
&\sum_{i=1}^{i=N} \sum_{j=1}^{j=M} X(i,j) * r_j \leq C_i \\
&\forall i \in \{1, N\}, j \in \{1, M\}, k \in \{1, E\}, X(i,j) \times S(j,k) = \\
&Ms_i(k) \\
&\forall i \in \{1, N\}, j \in \{1, M\}, X(i,j) * Avg\_delay(i) \leq l(j) \\
&\forall i \in \{1, N\}, j \in \{1, M\}, X(i,j) \in 0, 1 \\
&\forall i \in \{1, N-1\}, \forall k \in \{i+1, N\}, j > i, \\
&T_j(i,k) \geq \sum_{j=1}^{j=M} X(i,j) * r_j - \sum_{j=1}^{j=M} X(k,j) * r_j \\
&\forall i \in \{1, N-1\}, \forall k \in \{i+1, N\}, j > i, \\
&T_j(i,k) \geq -(\sum_{j=1}^{j=M} X(i,j) * r_j - \sum_{j=1}^{j=M} X(k,j) * r_j)
\end{aligned}$$

B. *Tabu-Search for solving our problem*

The complexity of the problem can make its solution prohibitively expensive computationally for large problem instances. In such cases, the use of meta-heuristics to find a sub-optimal solution, such as Tabu search [15] [16], is primordial. Before we proceed further, we give a general view of the Tabu Search method.

1) Overview of Tabu Search: Tabu Search (TS) is a meta-heuristic method that finds good solutions to large combinatorial problems, in many practical scenarios. The basic idea of TS is the use of short-term memories to avoid cycles and hence returning to already visited solutions "Tabus". These memory structures compose the Tabu List (TL) which contains a set of recently visited locations. Thus, a local search algorithm is used to move from a solution to another in the neighborhood solution space, until a termination condition. Usually, termination condition is a fixed number of algorithm iterations or a threshold value.

The TS algorithm starts with an initialization step to generate an initial potential solution X_{init}. We note that the farther this solution is from the optimal solution, the greater is the overall execution time.

2) TS-based MEC application placement: In this section, we describe how we use TS to optimize the placement of MEC applications. We first present the main elements of TS:

- For a MEC application, and as a first step, we filter the available MEP/EVCIs such that we delete the ones that do not provide the required services by the MEC application.
- A potential solution X is a $(N \times M)$ assignment matrix that ensures all the constraints in our formulation
$$X = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & x_{ij} & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$$
- To move from a solution to another, we consider swapping the assignment of two MEC applications to two EVCIs, selected randomly. Thus, a move $m(N, M)$ is a matrix where its elements are equal to zero except for the elements corresponding to the positions of the new and old assignment, which are set to one.
- A neighborhood of a solution X, or a new solution X', can be reached through a simple **XOR** operation: $X' = X \oplus m$.
- We consider the objective function value of each solution as its *attribute*. We then update the TL by adding the attribute of the best achieved solution X_{best}.

In fact, adding the attribute of the best-achieved solution, among all explored solutions, to the TL will enable to avoid returning to already visited solutions. Hence, we reduce both the time of computation and the required memory for TL.
We use a simple "greedy-based" algorithm, to generate an initial potential solution ensuring all our problem constraints. To this end, and for each application, we select randomly an EVCI verifying both latency and computing constraints. We repeat this procedure until we assign all MEC applications.

Once an initial solution is reached, we apply the TS-based algorithm in order to explore the solution space, as illustrated in **Algorithm 2**.

Algorithm 1 Tabu Search-based to optimize the MEC applications placement

```
Require: An initial solution $X_{init}$ (Algorithm 1), number of EVCIs $N$, number of applications $M$, offered_CPU($N$), required_CPU ($M$), offered_Latency ($N$), required_Latency ($M$), and $Max_{iter}$.
Ensure: An optimal (sub-optimal) assignment matrix $X_{best}$.
 1: $X_{best} \leftarrow X_{init}$
 2: $Attribute_{best} \leftarrow Attribute_{init}$
 3: while $iter \leq Max_{iter}$ do
 4:     Move procedure: generate a neighbor, $X'$, of the current solution $X_{best}$ by applying a move $m \in Neighbors(X_{best})$.
 5:     Attribute procedure: compute the attribute of the new solution $X'$ (Objective function value)
 6:     if ($Attribute_{best} > Attribute_{X'}$) then
 7:         $X_{best} \leftarrow X'$
 8:         $Attribute_{best} \leftarrow Attribute_{X'}$
 9:         Update the TL: add the $Attribute_{X'}$ to the TL.
10:     end if
11:     $iter \leftarrow iter + 1$
12: end while
13: return $X_{best}$
```

V. Simulation and results

In this section, we present results of experiments we performed to evaluate our placement scheme.

A. Simulation Setup

First, we solved the ILP using Python via the linear programming optimization package PuLP [6]. We also used Python to implement our TS-based algorithm. We varied the number of MEC applications as well as the EVCIs in order to evaluate our placement algorithm in terms of both: (i) the average latency offered by the selected EVCI, and (ii) the average number of offered computing resources (CPUs). We note that we generated randomly the C_i, r, Avg_delay_i, and l values.

Moreover, to validate the performance of our scheme, we compared it with another scheme that does not aim to balance the load among the MEPs, but, instead, to maximize the offered computing resources by the selected EVCI. Therefore, we define the objective function of this second scheme as follows:$Max \sum_{i=1}^{i=N} \sum_{j=1}^{j=M} (X(i,j) \times C_i)$ We note that this scheme shares the same constraints with our proposed solutions.

B. Performance evaluation of our Integer Linear Program

TABLE I shows the percentage allocation of MEC applications to six available EVCIs, when comparing our scheme to the no-load-balancing scheme. As we can see, our scheme succeeds in balancing the load among the existing EVCIs while ensuring our formulation constraints. For instance, when the number of applications is 9, our scheme hosts the applications at the EVCI 1, 2, 3, 5, 6 with a percentage

TABLE I: MEC Applications Load Balancing among six MEP/EVCIs.

		1	**2**	**3**	**4**	**5**	**6**
Our Scheme	**3**	33.33%				33.33%	33.33%
	6		16.66%	16.66%		33.33%	33.33%
	9	11.11%	11.11%	22.22%		22.22%	33.33%
	12	8.33%	16.66%	33.33%		25%	16.66%
No Load Balancing	**3**	33.33%	66.66%				
	6	66.66%				33.33%	
	9	55.55%			44.44%		
	12	91.66%	8.33%				

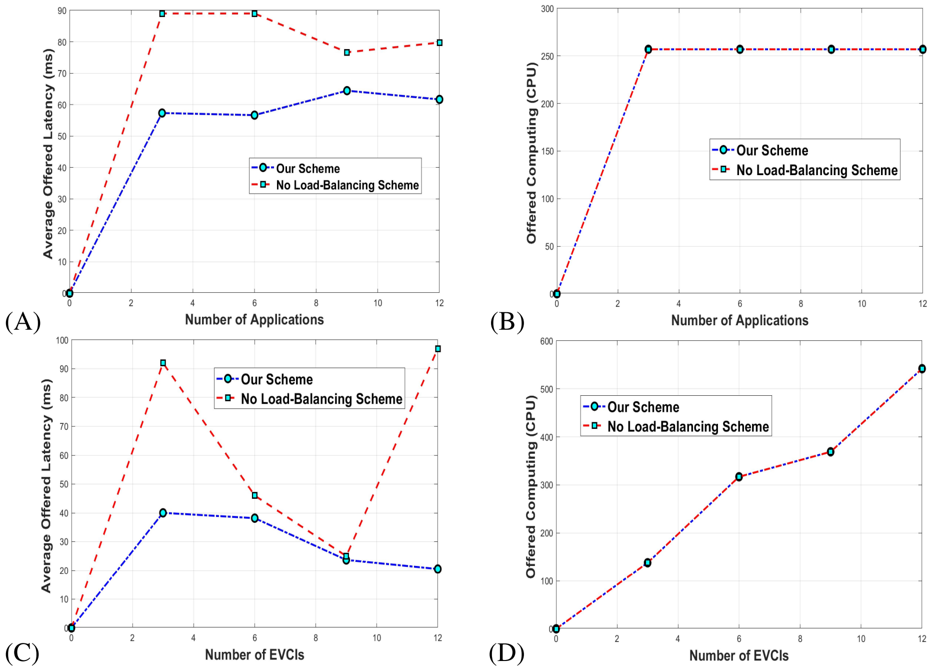

Fig. 2: Performance evaluation of our scheme in terms of our placement scheme. (A) The average offered latency while varying the number of applications. (B) The offered computing resources while varying the number of applications. (C) The average offered latency while varying the number of EVCIs. (D)The offered Computing resources while varying the number of EVCIs.

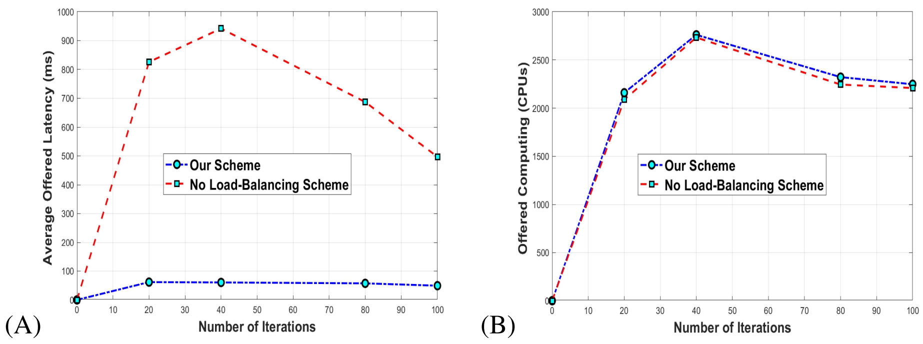

Fig. 3: Performance evaluation of our TS-based scheme in terms of the average offered latency (A) and computing resources (B), while varying the number of algorithm iterations and fixing the number of EVCIs and MEC applications to 50 and 100, respectively.

of 11.11%, 11.11%, 22.22%, 22.22%, 33.33%, respectively. We note that it did not assign any application to the EVCI 4, as the latter cannot ensure the problem constraints (latency and/or computing resources). However, for all cases, we observe that the no-load-balancing scheme hosts the applications at a maximum of two EVCIs. It is clear that this scheme aims only at maximizing the number of offered computing resources (CPUs), which results in not balancing the applications load between the EVCIs.

Fig. 2-(A) depicts the average latency offered by the selected EVCI when we increase the number of applications. Our scheme outperforms the other scheme by minimizing the offered latency whatever the number of applications. In fact, balancing the load among the MEPs enables also to improve the latency, as it will be possible to make more in parallel connections, which improves the connection delay to access to the user data plane (i.e., eNodeB). However, both schemes generate the same performance in terms of offered computing resources, when varying both the number of applications (cf. Fig. 2-(B)) and EVCIs (cf. Fig. 2-(D)). This is mainly because the no-load-balancing scheme also aims to maximize the available computing resources. In addition, we observe that both schemes provide a stable number of computing resources when we keep the number of EVCIs fixed (equal to 6, cf. Fig. 2-(B)), while it increases as we increase the number of EVCIs (cf. Fig. 2-(D)). This is a reasonable result as the offered computing resources depends strongly on the number of available EVCIs and not the applications.

Fig. 2-(C) shows the average offered latency while varying the number of EVCIs and fixing the number of applications to 6. As for Fig. 2-(A), our scheme improves latency when compared to the other scheme, as the latter focuses more on the computing resources. Moreover, the generated latency by our scheme decreases as we increase the number of EVCIs. It is clear that providing more EVCIs and balancing the load between them will enable to improve the application latency through the parallel connections, as we mentioned before.

C. Performance evaluation of our Tabu Search-based scheme

Fig. 3-(A) and (B) illustrate the average offered latency and computing resources of the selected solution using our TS-based algorithm. We clearly remark that our scheme minimizes the applications latency as compared to the no-load-balancing scheme. In addition, we see that the provided computing resources of our scheme's solution are a little bit more than that of the other scheme. These results demonstrate the efficiency of our Tabu Search algorithm to find an optimal (sub-optimal) solution that improves both the application latency and offered computing resources.

Therefore, our scheme achieves a better and stable performance compared to the no-load-balancing scheme. This shows not only the utility of balancing the applications load among EVCIs to provide the required computing resources and to achieve a good application latency, but also the efficiency of our TS-based algorithm to select and find a high-quality solution.

VI. Conclusion and Future Work

In this paper, we devised a new MEC application placement scheme in a full federated edge architecture following the ETSI model. Our scheme is based on both a linear program for a low number of applications as well as EVCIs, and on the meta-heuristic Tabu-Search when the number of applications and EVCIs is increased. It allows the MEO to decide which MEP/EVCI has to host the instance of a MEC application by considering ETSI MEC related constraints: computing resources, latency and MEC service availability. Experimental results showed the efficiency of our scheme in balancing MEC applications load among available MEP/EVCIs, as well as the ability of our TS-based algorithm to find sub-optimal, or optimal, solutions. As a future work, we are aiming to evaluate more thoroughly the performance of our TS-based scheme by comparing it with other meta-heuristics and considering other performance indicators.

References

[1] A. Gupta and R. K. Jha, "A survey of 5g network: Architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, 2015.

[2] A. Ksentini *et al.*, "Providing low latency guarantees for slicing-ready 5G systems via two-level MAC scheduling," *IEEE Network*, 2018, in press.

[3] T. Taleb *et al.*, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[4] *Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application lifecycle, rules and requirements management*, ETSI Group Specification MEC 010, 2017.

[5] *Mobile Edge Computing (MEC); Framework and Reference Architecture*, ETSI Group Specification MEC 003, 2016.

[6] S. Mitchell, S. M. Consulting, and I. Dunning, "Pulp: A linear programming toolkit for python," 2011.

[7] L. Yala *et al.*, "Latency and availability driven vnf placement in a mec-nfv environment," in *Proc. IEEE GLOBECOM*, 2018.

[8] F. B. Jemaa, G. Pujolle, and M. Pariente, "QoS-aware VNF placement optimization in edge-central carrier cloud architecture," in *Proc. IEEE GLOBECOM*, 2016.

[9] I. Benkacem *et al.*, "Optimal VNFs placement in CDN slicing over multi-cloud environment," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 616–627, 2018.

[10] F. Bari *et al.*, "Orchestrating virtualized network functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 725–739, 2016.

[11] S. Wang *et al.*, "Dynamic service migration in mobile edge-clouds," in *Proc. IFIP Networking*, 2015.

[12] K. Katsalis *et al.*, "SLA-driven VM scheduling in Mobile Edge Computing," in *Proc.* 9^{th} *IEEE International Conference on Cloud Computing (CLOUD)*, 2016, pp. 750–757.

[13] Z. Wen *et al.*, "Cost effective, reliable and secure workflow deployment over federated clouds," *IEEE Transactions on Services Computing*, vol. 10, no. 6, pp. 929–941, 2017.

[14] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads," in *Proc. IEEE* 3^{rd} *International Conference on Cloud Computing (CLOUD)*, 2010, pp. 228–235.

[15] H. Kamal, M. Coupechoux, and P. Godlewski, "A tabu search dsa algorithm for reward maximization in cellular networks," in *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, Oct 2010, pp. 40–45.

[16] R. B. Messaoud and Y. Ghamri-Doudane, "Qoi and energy-aware mobile sensing scheme: A tabu-search approach," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, Sept 2015, pp. 1–6.