Deployment and Migration of Virtualized Services with Joint Optimization of Backhaul Bandwidth and Load Balancing in Mobile Edge-Cloud Environments

Tarik Chanyour*[1], Mohammed Ouçamah Cherkaoui Malki[2]
FSDM, LPAIS
Sidi Mohamed Ben Abdellah University
P.O. Box 1796, Atlas-Fez, Morocco.

***Abstract*—Mobile edge-cloud computing environments appear as a novel computing paradigm to offer effective processing and storage solutions for delay sensitive applications. Besides, the container based virtualization technology becomes solicited due to its natural lightweight and portability as well as its small migration overhead that leads to seamless service migration and load balancing. However, with the mobility property, the users' demands in terms of the backhaul bandwidth is a critical parameter that influences the delay constraints of the running applications. Accordingly, a Binary Integer Programming (BIP) optimization problem is formulated. It minimizes the users' perceived backhaul delays and enhances the load-balancing degree in order to offer more chance to accept new requests along the network. Also, by introducing bandwidth constraints, the available user backhaul bandwidth after the placement are enhanced. Then, the adopted methodology to design two heuristic algorithms based on Ant Colony System (ACS) and Simulated Annealing (SA) is presented. The proposed schemes are compared using different metrics,and the benefits of the ACS-based solution compared to the SA-based as well as a genetic algorithm (GA) based solutions are demonstrated. Indeed, the normalized cost and the total backhaul costs are given by more optimal values using the ACS algorithm compared to the other solutions.**

Keywords*—Mobile edge-cloud computing; delay-sensitive services; container migration; container deployment; backhaul bandwidth; load balancing

I. Introduction

Mobile Edge Computing (MEC) is an emerging distributed computing paradigm that can deliver timely services to mobile users [1], [2]. They generally use resource-limited smart mobile devices (SMD) that allow them to run indispensable smart applications related to social networking, learning, businesses and entertainment. To reinforce privacy, reduce latency, preserve bandwidth and offer location-awareness, MEC enables computation and storage at the edge of the network using a set of edge nodes (EN). These nodes are resource-rich network cells or edge servers (ES) that are deployed in close proximity of the end-users and offer virtualized services to allow offloading of the mobile applications' workloads [3]. The use of these applications leads to appear new constraints related to mobility, limited energy, limited computational capacity and short latency.

The MEC model uses the virtualization techniques to master the resource allocation operations for Virtual Services (VSs) [4]. These VSs are often placed, migrated or replicated over the ENs according to the users' locations and resources availability while considering constraints such as QoS, load balancing and energy. Besides, the new container-based lightweight virtualization solution is intended to decrease the communication network overhead and enhance continuity and quality of services. Though, especially with the user's mobility intrinsic property that is mostly frequent and unpredictable and the limited coverage of nodes, a guaranteed QoS for the deployed virtualized services is the most critical issue [5]. Indeed, when the user moves far from the edge server that deploys the corresponding virtualized service, the service response time becomes significant and can hamper the smooth running of the service. Therefore, a service migration [6] process in this case becomes important to make the service more interactive and guarantee its continuity. But, due to the high cost of this process regarding its time and the consumption of the available network bandwidth and other resources, the migration decision is very critical. Actually, with the non-negligible migration overhead, frequent migration according to the user's movement cannot be tolerated in all network conditions, whereas limited migration leads to the accumulation of communication delays which may degrade the QoS.

Service migration has sprung up recently as a leading problem in MEC networks. It involves complex procedures to dynamically move running services from one edge node to another. It becomes solicited in different edge management procedures, such as service failures handling, load balancing, mobile workloads offloading handling, etc. Also, to guarantee service-level agreements (SLA) or seamless services, it has to meet many constraints related to the available network and computing resources, the latencies' order of magnitude as well as the users' mobility [7]. The migration decisions are taken while optimizing a general cost or profit function that is evaluated in a long-term or short-term scenarios. Its formulation uses many metrics such as the migration duration, service downtime duration, network resources consumption, etc. However, a precise evaluation of these metrics remains a major problem for a good modelling of this problem. On the one hand, because of the great diversity and the strong dynamicity of the parameters as well as the mobility of the users. On the other hand, because of the limitation of the resources involved in the migration which accentuates the constraints and limits the number of possible solutions.

II. Related Works

With the high-mobility characteristic in the context of Vehicular Edge Network, the authors of [8] considered a delay-based cost function involving wireless transmission, backhaul and computing delays. They examined the problem of joint service migration and mobility optimization with minimum migration cost and travel time. To solve the problem, a multi-agent deep reinforcement learning algorithm was proposed. In [9], the authors use migration frequency and migration time as the migration cost and suggest a QoS aware solution to enhance the handover operations by exchanging additional information in order to perform service migration.

With the user mobility awareness assumption and QoS concerns for efficient service migration in MEC networks, many relevant works target service migration optimization. The work in [10] uses the follow-me edge concept to derive a service performance optimization problem constrained to a long-term cost budget to decide the service migration. The decision metrics include the Computing and Communication delays plus the migration cost. The long-term optimization problem is decomposed using Lyapunov optimization then approximated based on Markov approximation to derive a near-optimal solution with fast convergence rate. Moreover, based on this last concept to guarantee high availability and prop ultra-low latency, in [11] the authors studied four container-based migration strategies. They considered both predefined and unknown path scenarios. The work in [12] considered a cost function with a combination of three metrics: the topology cost that depends on the network structure and routing mechanism, the user-perceived delay and the risk of location privacy leakage. They modelled the migration procedure as a Markov Decision Process (MDP) problem, and propose a modified policy iteration algorithm to find the optimal decision. Also, a distance-based MDP was proposed in [13] to optimize the trade-off between the user-experienced delay and migration cost while considering the distance separating the user and the service locations. The work in [14] considered a dynamic task migration problem with delays, tasks' deadlines and user mobility consideration. The objective function to maximize was the number of tasks with guaranteed deadlines.

However, the mobility information is usually unavailable in real world due to privacy and inaccuracy issues. With this consideration, several recent works tackled the optimization of service or container migration from various perspectives. The work in [15] studied container migration in edge networks using a joint load balancing and migration cost minimization model. The migration cost encompasses two main metrics: network transmission delay and container migration downtime. They designed a migration solution based on a modified Ant Colony System algorithm. In [16] a live migration framework of container-based offloading services is presented. The basic optimization idea consist in sharing common storage layers across the edge hosts. Also in [17], the authors addressed the high network consumption problem while migrating virtual machines within cloud-edge fusion computing. They proposed heuristic algorithms to balance migration and communication costs.

The rest of this work is organized as follows. The system's model is describe in Section III. The obtained optimization problem is presented in Section IV, and its resolution's approaches are summarized in Section V. Evaluation and results are presented in section VI. Finally, Section VII concludes the paper.

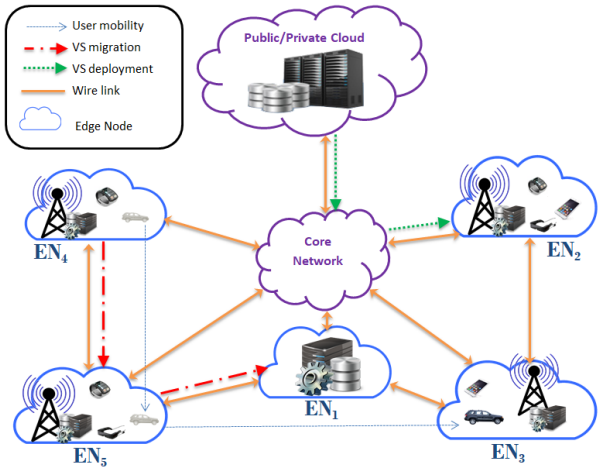

Fig. 1. Mobile Edge-Cloud System Architecture.

III. User's Backhaul and Load Balancing Aware Migration and Deployment of Containers (UBL-MDC)

In this section, the need to optimise a multiple criteria decision-making problem in the proposed edge-cloud architecture is shown. Then, the involved cost functions as well as the final overall objective function to optimize is formulated.

A. System Model

In this paper, the service deployment and migration problem from the perspective of an edge-cloud service provider is studied. As shown in Fig. 1, an edge-cloud network that uses a public/private cloud (PC) and a set of Edge Nodes (ENs) within a 2-D geographical local area is considered. Each EN is equipped with an Edge Server (ES) that can be hosted in a Base Station (BS) that offers access to the wireless communication network for all SMDs in its coverage, or simply deployed to offer to the network more processing and storage capabilities. In this last case, the server is called independent edge server and denoted (IS). For ease of use, an EN or its ES are indifferently used, while the edge-cloud server i is denoted s_i. A given ES within a BS serves the SMDs within the coverage area of the BS or other remote ones, whereas an independent ES serves only remote SMDs. The PC is supposed to have unlimited capacity, whereas all ESs are supposed heterogeneous with limited resources. Also, each ES can provide a set of independent virtualized services (VS) using the container-based lightweight virtualization technology where each running service uses a container instance and serves one SMD only. The set of all available edge-cloud servers is denoted $\mathcal{S} = \{s_1, s_2, ..., s_{\sigma_s}\}$ where σ_s is the number of servers. For ease of use, the set of all involved containers is denoted $\mathcal{C} = \{c_1, c_2, ..., c_{\sigma_c}\}$ where σ_c is the number of containers.

1) UBL-MDC Variables: To model the involved operations in the studied system, the decision variables are presented:

The migration binary decision variable of container i from its edge server s_i^c to EN j is denoted $\alpha_{i,j}$ where $\alpha_{i,j} = 1$ refers to the decision to migrate c_i from s_i^c to j, otherwise, $\alpha_{i,j} = 0$.

$$\alpha_{i,j} \in \{0;1\} \qquad ;i \in \mathcal{C}; j \in \mathcal{S} \tag{1}$$

Additionally, when migrating container i from its edge server s_i^c to j the decision variable to select the migration path among the possible paths set $\mathcal{P}_{s_i^c,j}$ is the binary variable $\beta_{i,j}^k$ where $\beta_{i,j}^k = 1$ refers to the decision to use the k-th path in $\mathcal{P}_{s_i^c,j}$ to migrate i from edge server s_i^c to j, otherwise $\beta_{i,j}^k = 0$.

$$\beta_{i,j}^k \in \{0;1\} \qquad ;i \in \mathcal{C}; j \in \mathcal{S}; k \in \mathcal{P}_{s_i^c,j} \tag{2}$$

2) Paths and delay: The SMDs get access to the ESs via wireless channels, while the nearby ENs are connected to each other in wired manner using high speed Ethernet cables or optical fibers. The MEC network topology is given by the set of nodes $\mathcal{S}$ and the set of links relying them. The set of links is denoted $\mathcal{L}$ which can be defined as $\mathcal{L} = \{\mathcal{L}_{j,j'} | ; j \in \mathcal{S}; j' \in \mathcal{S} \setminus \{j\}\}$ where $\mathcal{L}_{j,j'}$ is one hop link between ES s_j and $s_{j'}$. Also, $\mathcal{P}_{s_i^c,j}$ is used to denote the set of feasible [1] paths connecting ESs s_i^c and j that can serve to migrate container c_i located in ES s_i^c to server s_j. Without loss of generality, we assume that the set $\mathcal{P}_{s_i^c,j}$ is precalculated and given while deciding the containers migration. Then, $\mathcal{P}$ is used to denote the set of all sufficient paths connecting all pairs of distinct nodes (j, j') defined as:

$$\mathcal{P} = \{\mathcal{P}_{j,j'} \ / \ j \in \mathcal{S}; j' \in \mathcal{S} \setminus \{j\}\} \tag{3}$$

Each path p_k in $\mathcal{P}_{s_i^c,j}$ is an ordered set of distinct links of length $|p_k|$ such that $p_k = (\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_{|p_k|})$. Here, the source node of link $\mathcal{L}_1$ is s_i^c and the target node of the last link $\mathcal{L}_{|p_{j,j'}^k|}$ is j. For the remaining links, the source node of link $\mathcal{L}_\ell$ is the target node of link $\mathcal{L}_{\ell-1}$ and the target node of link $\mathcal{L}_\ell$ is the source node of link $\mathcal{L}_{\ell+1}$. Fig. 2 shows a network topology example given by a set of five ESs $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5\}$ where $\sigma_s = 5$ and six wire links denoted : $\mathcal{L} = \{\mathcal{L}_{s_1,s_2}, \mathcal{L}_{s_1,s_3}, \mathcal{L}_{s_2,s_3}, \mathcal{L}_{s_1,s_4}, \mathcal{L}_{s_1,s_5}, \mathcal{L}_{s_4,s_5}\}$. Also, the dotted links show a migration path instance p1 with its ordered links set given by $p_1 = (\mathcal{L}_{s_2,s_1}, \mathcal{L}_{s_1,s_4})$. Then, the set of possible paths connecting s_2 and s_4 is given by the following set $\mathcal{P}_{2,4} = \{p_1, p_2, p_3, p_4\}$ where: $p_1 = (\mathcal{L}_{s_2,s_1}, \mathcal{L}_{s_1,s_4})$, $p_2 = (\mathcal{L}_{s_2,s_1}, \mathcal{L}_{s_1,s_5}, \mathcal{L}_{s_5,s_4})$, $p_3 = (\mathcal{L}_{s_2,s_3}, \mathcal{L}_{s_3,s_1}, \mathcal{L}_{s_1,s_4})$ and $p_4 = (\mathcal{L}_{s_2,s_3}, \mathcal{L}_{s_3,s_1}, \mathcal{L}_{s_1,s_5}, \mathcal{L}_{s_5,s_4})$.
In the proposed model each link $\ell \in \mathcal{L}$ is characterized by its total available bandwidth $b(\ell)$. In addition, given the path $p_k \in \mathcal{P}_{s_i^c,j}$ and the set $\mathcal{L}$ of all σ_l links, the binary array $\delta_{i,j}^k$ of length σ_l indicating membership of all links to p_k is defined. Accordingly, the binary indicators $\delta_{i,j}^{k,\ell}$ of each link $\ell \in \mathcal{L}$ can be computed using the paths in $\mathcal{P}_{s_i^c,j}$ such that $\delta_{i,j}^{k,\ell}$ takes 1 if link ℓ in $\mathcal{L}$ is crossed in path $p_k \in \mathcal{P}_{s_i^c,j}$, otherwise it takes 0.

[1] we assume that a restriction set of paths is sufficient to obtain the optimal solution without the need to consider all possible paths

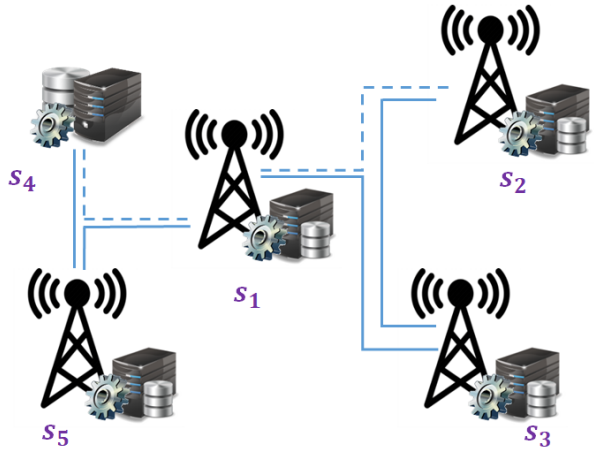

Fig. 2. Inter-Server Routing Paths Example.

Thus, each available path $p_k \in \mathcal{P}_{j,j'}$ with the hop count $\mathcal{H}_{j,j'}^k$ offers an allocatable bandwidth $\mathcal{B}_{j,j'}^k$ (see Ref [15]) for multi hop data transmission. They are respectively expressed as:

$$\mathcal{B}_{j,j'}^k = \begin{cases} \infty & ; j = j' \\ \min\limits_{\ell \in p_k} \{b(\ell)\} & ; j \neq j' \end{cases} \quad ;j \in \mathcal{S}; j' \in \mathcal{S}; p_k \in \mathcal{P}_{j,j'} \tag{4}$$

$$\mathcal{H}_{j,j'}^k = \begin{cases} 0 & ; j = j' \\ |p_k| - 1 & ; j \neq j' \end{cases} \quad ;j \in \mathcal{S}; j' \in \mathcal{S}; p_k \in \mathcal{P}_{j,j'} \tag{5}$$

Thus, when container c_i is transferred to node j, the backhaul bandwidth between its target node s_i^t and node j is given by:

$$\mathcal{B}_{i,j} = \begin{cases} \infty & ; \quad s_i^t = j \\ \max\limits_{k \in \mathcal{P}_{s_i^t,j}} \left\{\mathcal{B}_{s_i^t,j}^k\right\} & ; \quad s_i^c \neq j \end{cases} \qquad ;i \in \mathcal{C}; j \in \mathcal{S} \tag{6}$$

which gives:

$$\mathcal{B}_i = \sum_{j \in \mathcal{S}} \alpha_{i,j} \mathcal{B}_{i,j} \qquad ;i \in \mathcal{C} \tag{7}$$

3) Containers: Hereafter and for ease of notation, the following variables i, j, k, ℓ, m are reserved to use for containers, servers, paths, links and resources respectively. Also, from now on, each container c_i is characterized by the following operating parameters : $\Omega_i \triangleq \langle s_i^c, s_i^t, \mathcal{R}_i^{dem}, B_i^{ser}, x_i \rangle$ and the set of all operating parameters is denoted $\Omega = \{\Omega_i\}_{i \in \mathcal{C}}$. Here, s_i^c refers to the current server hosting container c_i and s_i^t refers to the actual node hosting the communication access point connecting the user of the service associated with container c_i. This node is considered as the best candidate target node for deployment or migration so that the best transfer paths for c_i are in $\mathcal{P}_{s_i^c,s_i^t}$. Actually, if a path is a feasible solution, container c_i will be migrated to the direct proximity of the user with no communication overhead. Also, $\mathcal{R}_i^{dem} = \{r_{i,1}^d, r_{i,2}^d, ..., r_{i,\sigma_r}^d\}$ represents the resources demand set of c_i and are given in the number of standardized virtual resource units. Here, σ_r is the number of resource types and $r_{i,m}^d$ represents the required quantity in terms of resource r_m demanded by container i. Furthermore, B_i^{ser} concerns the minimum allowable data rate in terms of available bandwidth between c_i and its associated user after the migration or deployment procedures.The binary

x_i indicates whether c_i is requested for a migration ($x_i = 1$) or for a new deployment procedure ($x_i = 0$).

4) Edge servers resources: Each ES provides a set of resources among multiple types including CPU, GPU, memory, storage, etc. Here, the set of possible σ_r resources is denoted $\mathcal{R} = \{r_1, r_2, ..., r_{\sigma_r}\}$. Accordingly, every server j is characterized by its capacity set in terms of resources which is denoted $\mathcal{R}_j^{cap} = \{r_{j,1}^c, r_{j,2}^c, ..., r_{j,\sigma_r}^c\}$. Here, $r_{j,m}^c$ represents the maximum available quantity in terms of resource r_m that s_j can furnish.
Within the server j which runs a set of containers using the allocated resources, the deployment and migration will result in hosting new containers and freeing others conforming to the placement decisions. Thus, the utilization of resource r_m on ES j after the migration process is calculated as follows:

$$r_{j,m}^u(\alpha) = \sum_{i \in \mathcal{C}} \alpha_{i,j} r_{i,m}^d \qquad ; j \in \mathcal{S}; m \in \mathcal{R}. \tag{8}$$

5) Container deployment: Deploying a service in this work refers to the transfer of unstarted components of the container (program codes, libraries, databases, etc.) from the storing node to a MEC server in order to make them available to serve a user. The provider's containers are stored in its PC or in a specific known EN depending on the requested service. Thus, all new incoming service requests from the users trigger service deployment from the hosting nodes to the ENs. Here, the same notation s_i^c is adopted to refer the hosting node of the requested container c_i.

6) Container migration: Migrating a container tries to achieve load balancing of ESs and increase the number of services that meet the execution latency constraints if necessary. this process involves the transfer of all runtime memory states as well as the related storage data that should be synchronized in the target ES. Furthermore, migration traffic routing in MEC networks not only helps to significantly reduce services downtime and interruption by selecting the best routing paths, but it protect the network from route failure. Indeed, if some links are in use or completely fail, alternate paths can be selected to redirect and salvage the data flows. Accordingly, a container migration decision has to found the expedited path to route the migration flows while avoiding the network congested links.

The important notations used are summarized in Table I.

TABLE I. MAIN NOTATIONS

Notation	Definition
$\mathcal{C}$	The set of containers
$\mathcal{S}$	The set of edge-cloud servers
$\mathcal{L}$	The set of links
σ_c, σ_s, σ_r	The the total number of containers, servers, resources
$\mathcal{P}_{j,j'}$	The set set of sufficient paths connecting servers s_j and $s_{j'}$
$\mathcal{B}_{j,j'}^k$	The bandwidth of path $p_k \in \mathcal{P}_{j,j'}$
$\mathcal{H}_{j,j'}^k$	The hop count of path $p_k \in \mathcal{P}_{j,j'}$
$\mathcal{B}_i$	The backhaul bandwidth associated with container c_i
Ω_i	The operating parameters of container c_i
$r_{i,m}^d$	The c_i demand in terms of resource r_m
$r_{j,m}^u$	The s_j resource usage in terms of resource r_m

B. The Cost Models

As already alluded above, the containers' deployment or migration has to be decided while optimizing a cost model as it is the most suitable way to favour one possible migration solution over another. Thus, in the present section the costs that are involved to formulate the objective function of the optimization problem are presented. Table II shows some important notations used to express these costs.

TABLE II. IMPORTANT COST NOTATIONS

Notation	Definition
$Cost_{i,j}^{back}$	The container c_i backhaul cost when transferred to s_j
$Cost^{back}$	The overall user backhaul cost
$Cost_j^{proc}$	The processing load cost related to server s_j
$Cost^{proc}$	The overall processing load cost
$Cost^{netw}$	The overall network load cost
$Cost(\alpha, \beta)$	The cost or objective function
ϕ_0, $\phi_{i,j}$	The pheromone initial and current values
Δ_ϕ^l, Δ_ϕ^g	The local and global pheromone evaporation rates
ε_1, ε_2	The pheromone and heuristic information parameters
$temp_0$	The initial temperature value

1) User backhaul cost: After placing container c_i at node j, the backhaul delay of its user depends on the characteristics of the path connecting node j and its communication access node s_i^t. In fact, the ideal situation is achieved if $j = s_i^t$. Accordingly, to favour such migration, this cost is introduced in order to bring the containers as close as possible to their end users. Generally, the smaller is this cost the more efficient the placement is. To assess this cost, the available bandwidth between nodes j and s_i^t as well as the hop count between them are used. Accordingly, the following weighted sum is adopted:

$$Cost_{i,j}^{back} = \begin{cases} 0 & ; s_i^t = j \\ \min\limits_{k \in \mathcal{P}_{s_i^t,j}} \left\{ \Delta_r \dfrac{\min\limits_{k' \in \mathcal{P}_{s_i^t,j}} \mathcal{B}_{s_i^t,j}^{k'}}{\mathcal{B}_{s_i^t,j}^k} + \Delta_h \dfrac{\mathcal{H}_{s_i^t,j}^k}{\max\limits_{k' \in \mathcal{P}_{s_i^t,j}} \mathcal{H}_{s_i^t,j}^{k'}} \right\} & ; s_i^t \neq j \end{cases} \tag{9}$$

Here $i \in \mathcal{C}; j \in \mathcal{S}$ and $Cost_{i,j}^{back}$ is ranging in [0,1], Δ_r and Δ_h are the weights associated respectively with the available data rate (bandwidth) and the hop count costs such that $\Delta_r + \Delta_h = 1$. Also, the fractions' max and min expressions are used for normalization purpose. Therefore, with the decision vector α, the overall user backhaul cost can be obtained as:

$$Cost^{back}(\alpha) = \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{S}} \alpha_{i,j} Cost_{i,j}^{back} \tag{10}$$

2) Load balancing cost: To ensure the service quality while taking into account the service delay, the model favours containers' migration from over-loaded ENs to release resources for future nearby users requests. Also, to avoid the unbalanced network load, e.g. some links are highly loaded while some others are less loaded, the links traffic load metric is introduced. The main intuition behind balancing this load is to select paths that best balance the traffic loads across different links and keep critical links available for future traffic. Accordingly, the load balancing cost involves the processing or computation load cost of the running containers on all ENs and the traffic load of all available links. The processing load

ratios $\theta_{j,m}$ of resource r_m in ES j and their mean value $\bar{\theta}_m$ are defined as follows:

$$\theta_{j,m}(\alpha) = \frac{r^u_{j,m}(\alpha)}{r^c_{j,m}} \in [0,1] \qquad ; j \in \mathcal{S}; m \in \mathcal{R} \tag{11}$$

$$\overline{\theta_m}(\alpha) = \sum_{j \in \mathcal{S}} \frac{\theta_{j,m}}{\sigma_s} \qquad ; m \in \mathcal{R} \tag{12}$$

Then, the processing load related to ES j with regards to all resource types is defined as:

$$Cost^{proc}_j(\alpha) = \sum_{m \in \mathcal{R}} \frac{|\theta_{j,m}(\alpha) - \overline{\theta_m}(\alpha)|}{\sigma_r} \qquad ; j \in \mathcal{S} \tag{13}$$

which gives the following overall processing load:

$$Cost^{proc}(\alpha) = \sum_{j \in \mathcal{S}} \sum_{m \in \mathcal{R}} \frac{|\theta_{j,m}(\alpha) - \overline{\theta_m}(\alpha)|}{\sigma_s \sigma_r} \in [0,1] \tag{14}$$

On the other hand, the network load balancing cost shows the distribution ratio of the links' load or indicates whether containers receive a fair share of data transfer resources. Hence, the lack of capacity ϑ_ℓ of link ℓ using the the allowable bandwidth $\mathcal{B}^k_{s^c_i,j}$ is given by:

$$\vartheta_\ell(\alpha,\beta) = max\left\{0; \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{S}} \sum_{k \in \mathcal{P}_{s^c_i,j}} \left(\alpha_{i,j} \beta^k_{i,j} \delta^{k,\ell}_{i,j} \mathcal{B}^k_{s^c_i,j}\right) - b(\ell)\right\}; \ell \in \mathcal{L} \tag{15}$$

Accordingly, this lack of capacity is chosen as the network load unbalance cost related to link ℓ which gives the following overall network traffic load unbalance cost:

$$Cost^{netw}(\alpha,\beta) = \sum_{\ell \in \mathcal{L}} \vartheta_\ell(\alpha,\beta) \tag{16}$$

Moreover, as $\vartheta_\ell(\alpha,\beta) \leqslant b(l) * (\sigma_c - 1)$, herein the following normalization sum is presented:

$$L^{cap} = (\sigma_c - 1) \sum_{\ell \in \mathcal{L}} b(\ell) \tag{17}$$

Finally, the following weighted sum is adopted to asses the overall load balancing cost where Δ_p and Δ_n are the weights associated respectively with both processing and network loads such that $\Delta_p + \Delta_n = 1$. Also, the denominators in this expression are used for normalization purpose.

$$Cost^{load}(\alpha,\beta) = \Delta_p Cost^{proc}(\alpha) + \Delta_n \frac{Cost^{netw}(\alpha,\beta)}{L^{cap}} \in [0,1] \tag{18}$$

IV. The UBL-MDC Problem Formulation

A. Multi-objective Cost Function

Now, to get the overall cost model, a multi-criteria migration and deployment decisions by considering all the three cost metrics within the proposed edge computing system is designed. The proposed multi-objective function is formulated as a weighted sum of these four costs using the following function:

$$Cost(\alpha,\beta) = \Delta_b \frac{Cost^{back}(\alpha)}{\sigma_c} + \Delta_l Cost^{load}(\alpha,\beta) \tag{19}$$

Here Δ_b, and Δ_l are regulatory weights constants to balance this cost function. Their values are ranging in [0,1] such that $\Delta_b + \Delta_l = 1$. By deciding these weights one can adjust the priority to attribute to each metric. Here, the variables are given by α (two dimensions binary array $[\sigma_c \times \sigma_s]$) and β (two dimensions array $[\sigma_c \times \sigma_s]$ of vectors where $\beta_{i,j}$ is a vector of binaries of length $|\mathcal{P}_{i,j}|$).

B. Constraints

In the proposed model model, the case when container i is not migrated is represented by setting $\alpha_{i,s^c_i} = 1$ and $\alpha_{i,j} = 0$ for $j \in \mathcal{S} \setminus \{s^c_i\}$ and if migrated, only one target server is selected. Accordingly, the migration decision of container i has to meet the following constraint:

$$\sum_{j \in \mathcal{S}} \alpha_{i,j} = 1 \qquad ; i \in \mathcal{C}; \tag{20}$$

By selecting a path $p = \left(\mathcal{L}_{s^c_i,s_1}, \mathcal{L}_{s_1,s_2}, ..., \mathcal{L}_{s_{|p|-1},j}\right)$ in $\mathcal{P}_{s^c_i,j}$ to serve the transfer flow of container c_i from node s^c_i to j, many constraints have to be satisfied. With the start node s^c_i of path p, its last node j must be the placement node of container i which is expressed as:

$$\sum_{k \in \mathcal{P}_{s^c_i,j}} \beta^k_{i,j} = \alpha_{i,j} \qquad ; i \in \mathcal{C}; j \in \mathcal{S} \tag{21}$$

Also, the resource capacity in EN j must satisfy all the containers resource requirements that are decided to be deployed in or migrated to ES j for all resource types, which is finally formulated as:

$$\sum_{i \in \mathcal{C}} \alpha_{i,j} r^d_{i,m} \leq r^c_{j,m} \; ; j \in \mathcal{S}; m \in \mathcal{R} \tag{22}$$

Lastly, the serving bandwidth constraint after the placement of container c_i using the maximal available bandwidth in (7) is formulated as:

$$\mathcal{B}_i \geqslant B^{ser}_i \qquad ; i \in \mathcal{C} \tag{23}$$

C. Formulation

In light of the above clarifications of the studied problem, the formulation of the proposed UBL-MDC framework which aims to efficiently deploy and migrate the involved containers while considering their priorities is presented. The joint deployment, migration and route selection are made while deciding the best placements to minimize the objective consisting of the costs related to the resulting users backhaul bandwidth and the load balancing degree. Finally, the following optimization problem $\mathscr{P}1$ generates the minimal deployment and migration cost with resource allocation and traffic routing while maximum number of priority containers are satisfied.

$$\mathscr{P}1 : \underset{\{\alpha,\beta\}}{\text{minimize}}\; Cost(\alpha,\beta)$$
$$\text{s.t. } (1),(2),(20),(21),(22),(23)$$

Indeed, this formulation minimizes the aforementioned four metrics influencing the performance of the studied mobile edge-cloud system and the users' satisfaction according to their priorities.

D. The UBL-MDC Problem Complexity

Since problem $\mathscr{P}1$ is a binary integer programming problem, it is considered to be NP-complete. This is highlighted when showing its search space dimension that is $2^{\sigma_c \sigma_s}\left(\sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{S}}\left(2^{|\mathcal{P}_{i,j}|}\right)\right)$. For example, when $\sigma_c = 20$, $\sigma_s = 5$ and $|\mathcal{P}_{i,j}| = 10$, the search space size is $2^{100} \times (100 \times 2^{10}) \simeq 1.298 \times 10^{35}$. As such, the search space's exponential growth with the problem's dimension is obvious and one can observe the excessive computational requirement to solve such a problem. Therefore, the following section shows the development procedure of a low-complexity heuristic scheme.

V. Problem Resolution

A. The BFS-PS Exact Solution

To get the optimal containers' migration and deployment decision given by problem $\mathscr{P}1$, an exhaustive search is performed over all possible solutions using a Brute Force Search with Path Selection that is denoted (BFS-PS). It is presented in Algorithm 1. Unfortunately, this search is an $O(\sigma_c \times \sigma_s \times N)$ time complexity solution where $N = \prod_{i=1}^{\sigma_c}\left(\sum_{j=1}^{\sigma_r}(|\mathcal{P}_{i,j}|)\right)$ and is feasible for limited settings. Indeed, when $\sigma_c = 20$, $\sigma_s = 5$ and $|\mathcal{P}_{i,j}| = 10$, the iterations' count $N \simeq 9.536 \times 10^{33}$, which is already not feasible.

Algorithm 1 : BFS based Containers' Migration and Deployment

Require: $\mathcal{C}$,$\mathcal{S}$,$\mathcal{P}$,Ω
Ensure: optimal decisions α^*, β^* with cost Γ^*

$\Gamma^* \leftarrow \infty$
$N \leftarrow \prod_{i=1}^{\sigma_c}\left(\sum_{j=1}^{\sigma_r}(|\mathcal{P}_{i,j}|)\right)$;
for $l = 0$ to $N-1$ **do**
  build β from l;
  for each container i in $\mathcal{C}$ **do**
    for each node j in $\mathcal{S}$ **do**
      if $\sum_{k=1}^{|\mathcal{P}_{i,j}|}\beta_{i,j}^k == 0$ **then**
        $\alpha_{i,j} \leftarrow 0$;
      else
        $\alpha_{i,j} \leftarrow 1$;
      end if
    end for
  end for
  if constraints of $\mathscr{P}1$ are satisfied **then**
    $X \leftarrow Cost(\alpha, \beta)$ according to (19);
    if $X < Cost^*$ **then**
      $(\alpha^*, \beta^*, \Gamma^*) \leftarrow (\alpha, \beta, X)$
    end if
  end if
end for
return $(\alpha^*, \beta^*, \Gamma^*)$

As input, Algorithm 1 requires the parameters' vector Ω as well as the information regarding containers, servers and paths. The main for loop of the algorithm iterates N times over the instructions' bloc that tries to enhance the best solution using variables α and β that are built using the current iteration value.

B. ACS-PS Approximate Algorithm

To get a feasible containers' migration and deployment decisions, hereafter an efficient discrete ACS-based algorithm with Paths Selection (ACS-PS) is designed with two different migration strategies. To compare its performance, two other meta-heuristic algorithms based on simulated annealing (SA) and genetic algorithms (GA) are used. The first is summarized in Algorithm 4 whereas the second is based on the work in [15].

1) Algorithm description: ACS schemes adopt pheromone evaporation and sharing strategies to share the learned experience among different ants' groups. They simulate the feeding process of ants to simulate the decision of containers' migration and deployment. The main pieces of this algorithm are summarized as follows:

- Ants are randomly placed in the containers to be transferred.
- every ant A_a selects a mapping tuple $< c_i; s_j >$ with a probability $p_{i,j}$, referring the transfer of container c_i to node s_j using path p_k according to the pheromones $\phi_{i,j}$ and the heuristic information $\psi_{i,j}$. Then, c_i is placed into tabu list $Tabu_a$ of A_a.
- To get its migration plan, ant A_a returns to the next container in the transfer containers set $\mathcal{C}$, and repeats the previous process to complete the next migration allocation.
- That all the ants complete the allocation of all the transfer containers in $\mathcal{C}$ once, can be regarded as one iteration.
- The algorithm terminates when the maximum iterations' number is reached.

2) Algorithm skeleton: In practice, ants use a kind of chemical substance named pheromone to share information with each other [18]. Its initial value is defined as follows:

$$\phi_0 = \frac{1}{\sigma_c} \tag{24}$$

Pheromone variation rules: When transferring the containers, the ACS algorithm dumps the ants' search experience using the matrix $[\phi]$ of size $\sigma_c \times \sigma_s$. Each element $\phi_{i,j}$ saves the pheromone amount that informs ants about the tendency to choose pair $(c_i; s_j)$.
The next equations are the rules serving to update the pheromone locally and globally, respectively:

$$\phi_{i,j}^{new} = \phi_{i,j}^{old} \times (1 - \Delta_\phi^l) \tag{25}$$

$$\phi_{i,j}^{new} = \phi_{i,j}^{old} + \Delta_\phi^g \times \Delta_{i,j}^a \tag{26}$$

here Δ_ϕ^l and Δ_ϕ^g are the local and global pheromone evaporation rates respectively. $\Delta_{i,j}^a$ is its increment of additional

pheromone defined by:

$$\Delta_{i,j}^{a} = \begin{cases} \dfrac{1}{Cost(X_a^+)} & ; \text{ if } \alpha_{i,j} = 1 \text{ in } X_a^+ \\ 0 & ; otherwise \end{cases} \quad (27)$$

where $Cost(X_a^+)$ is the cost value of an iteration's best solution found by ant A_a. Actually, when the mapping relation tuple $< c_i; s_j >$ is chosen, the ant updates locally the pheromone value of this path using Eq. (25). On the other hand, when the mapping relation tuples of all current solutions is completed, the best one w.r.t. $Cost$ is chosen to perform pheromone update globally using Eq. (26) in order to maintain the experience of the global best solution.

Heuristic information: The proposed model uses heuristic information $\psi_{i,j}$ that is obtained based on the maximum allowable bandwidth to transfer container c_i to node n_j that is expressed as:

$$\psi_{i,j} = \max_{k' \in \mathcal{P}_{s_i^t,j}} \mathcal{B}_{s_i^t,j}^{k'} \quad (28)$$

Usually ants tend to choose the path with more pheromones and higher expectations of the ongoing path. Nevertheless, this deterministic choice has the disadvantage to fall into local optimum. Accordingly, ACS algorithm reacts by using a pseudorandom rule where ants probabilistically select the next mapping transfer tuple $< c_i, s_j, p_k >$ using a probabilistic rule. First, Eq. (29) defines the set $\omega_a(i)$ of possible target nodes j' related to ant A_a and their leading routes k' that verify all constraints in (31). Each element in this set represents a possible candidate placement node j with its associated possible leading routes that are given by the set $\omega_{a,j}(i)$. The set of candidate placement nodes only in $\omega_a(i)$ is denoted $\overline{\omega_a}(i)$.

$$\omega_a(i) = \{(j', k') \mid \text{if (31) are satisfied} \} \quad (29)$$

$$\overline{\omega_a}(i) = \{ \quad j \quad | \quad (j,k) \in \omega_a(i)\} \quad (30)$$

$$\begin{cases} \sum_{i' \in \mathcal{C}} \alpha_{i',j} r_{i',m}^{d} + r_{i,m}^{d} \leq r_{j',m}^{c} & m \in \mathcal{R} \\ \mathcal{B}_{s_i^t,j'}^{k'} \geqslant \mathcal{B}_i \end{cases} \quad (31)$$

The nodes selection: The next pair container-node is chosen based on the following equation:

$$j = \begin{cases} \underset{j' \in \overline{\omega_a}(i)}{\text{argmax}} \{(\phi_{i,j})^{\varepsilon_1} \times (\psi_{i,j})^{\varepsilon_2}\} & if q \leqslant q_0 \\ \text{Roulette Wheel}\{\overline{\omega_a}(i); \chi_{i,j}\} & otherwise \end{cases} \quad (32)$$

where q is a uniformly distributed random number ranging in [0, 1] and $q_0 \in [0, 1]$ is a threshold parameter. ε_1 and ε_2 are pheromone and heuristic information parameters, respectively. When $q \leqslant q_0$, A_a choose pair (i, j) with the maximum value to transfer c_i to node j. Otherwise, the pair (i, j) is chosen with the Roulette Wheel procedure (see Alg.(2)) within the set $\overline{\omega_a}(i)$ using probabilities $\chi_{i,j}$ defined in Eq. (33).

$$\chi_{i,j} = \frac{(\phi_{i,j})^{\varepsilon_1} \times (\psi_{i,j})^{\varepsilon_2}}{\sum_{j' \in \overline{\omega_a}(i)} (\phi_{i,j'})^{\varepsilon_1} \times (\psi_{i,j'})^{\varepsilon_2}} \quad (33)$$

The node-path pair selection: if container c_i is selected for transfer, the model proposes to select the pair $s_j - p_k$ denoted (j, k) as the target node and the path of its transfer. The adopted path selection strategy uses two versions: the first strategy denoted (ACS-PS-1) select the path with the maximum allowable bandwidth, while the second one denoted (ACS-PS-2) adopts a random selection strategy. With the first strategy ACS-PS-1, the following equation that gives the maximum transfer bandwidth while choosing path p_k is adopted:

$$k \leftarrow \underset{k' \in \omega_{a,j}(i)}{\text{argmax}} \left\{ \mathcal{B}_{s_i^c,j}^{k'} \right\} \quad (34)$$

Algorithm 2 : Roulette Wheel Rule Algorithm for Container c_i using $\overline{\omega_a}(i)$.

Require: $\mathcal{S}$,$\mathcal{P}$,$\overline{\omega_a}(i)$,Ω_i, ε_1 and ε_2
Ensure: the candidate node j_0;
1: **for** each node j in $\mathcal{S}$ **do**
2: **if** j in $\overline{\omega_a}(i)$ **then**
3: calculate $\chi_{i,j}$ using Eq. (33)
4: **else**
5: $\chi_{i,j} \leftarrow 0$;
6: **end if**
7: **end for**
8: $q1 \leftarrow random(0, 1) * \chi^{total}$;
9: $p \leftarrow 0$;
10: **for** each node j in $\overline{\omega_a}(i)$ **do**
11: $p \leftarrow p + \chi_{i,j}$;
12: **if** $q1 \geqslant p$ **then**
13: $j_0 \leftarrow j$;
14: break;
15: **end if**
16: **end for**
17: **return** j_0

3) Algorithm pseudo-code: The pseudo-code of the proposed algorithm is summarized in Algorithm 3 where a solution X_a is given by the variables' arrays (α, β) and X is the solutions' set of all ants.

As input, Algorithm 3 requires the sets $\mathcal{C}$,$\mathcal{S}$ and $\mathcal{P}$; the parameters' vector Ω, the maximum iterations count parameter n^{max}, the ants' count σ_a, the pheromone initial value q_0, the local and global pheromone evaporation rates Δ_{ϕ}^{l} and Δ_{ϕ}^{g}; ε_1, ε_2 the pheromone and heuristic information parameters and the path selection strategy s. In lines 1 to 3, the initial solution's vectors are built and the optimal cost F^* associated with the optimal solution (α^*, β^*) is initialized. In line 4, the general for loop repeat the process using n^{max} iterations where in each iteration all ants are involved using the loop in line 6. At each ant step, probability matrix is updated (lines 7-11), the containers' placement decisions with paths' selection are performed using Eq. (32) and strategy s which results in the vectors α and β (lines 12-35); and the local update of pheromone is executed. Then the iteration solutions corresponding to all ants are examined with a global pheromone update(lines 38-40) using the best solution and Eq.(26).

C. The SA-PS Approximate Algorithm

In this section, the proposed Simulated Annealing based heuristic solution with Paths Selection (SA-PS) is described. This heuristic optimization technique is characterized by its simplicity and general applicability features. In terms of speed,

Algorithm 3 : ACS-Based Container Transfer Algorithm with Path Selection (ACS-PS)

Require: $\mathcal{C}$,$\mathcal{S}$,$\mathcal{P}$,Ω, n^{max}, σ_a, q_0, Δ^l_ϕ, Δ^g_ϕ, ε_1, ε_1 and strategy s
Ensure: the Global solution (α^*, β^*);

Generate an initial solution (α, β)
Calculate $F = Cost(\alpha, \beta)$ according to (19);
$(\alpha^*, \beta^*, F^*) \leftarrow (\alpha, \beta, F)$
for $n = 1$ to n^{max} **do**
  $X \leftarrow \{\}$
  for $a = 1$ to σ_a **do**
    for each container i in $\mathcal{C}$ **do**
      for each node j in $\mathcal{S}$ **do**
        calculate $\chi_{i,j}$ using Eq. (33)
      end for
    end for
    for each container i in $\mathcal{C}$ **do**
      choose pair $< s_{j_0}; p_{k_0} >$ from $\omega_a(i)$ using
      Eq. (32) and strategy s;
      for each node j in $\mathcal{S}$ **do**
        if $j = j_0$ **then**
          $\alpha^j_i \leftarrow 1$;
          for each path k in $\mathcal{P}_{s^c_i, j_0}$ **do**
            if $k = k_0$ **then**
              $\beta^k_{i,j_0} \leftarrow 1$;
            else
              $\beta^k_{i,j_0} \leftarrow 0$;
            end if
          end for
        else
          $\alpha^j_i \leftarrow 0$;
          for each path k in $\mathcal{P}_{s^c_i, j}$ **do**
            $\beta^k_{i,j} \leftarrow 0$;
          end for
        end if
      end for
      update the local pheromone according to
      Eq. (25);
      put c_i into $Tabu_a$;
    end for
    put solution $X_a = (\alpha, \beta)$ into X;
  end for
  $X^+ \leftarrow \underset{X_a \in X}{\operatorname{argmin}} \{Cost(X_a)\}$;
  $F \leftarrow Cost(X^+)$
  update the global pheromone according to Eq. (26);
  if $F < F^*$ **then**
    $(\alpha^*, \beta^*, F^*) \leftarrow (\alpha, \beta, F)$
  end if
end for

it is considered among the main efficient heuristics compared to other techniques. Probabilistically, this algorithm accepts not only cost gain, but also cost degradation in order to leave the local minima. Inspired by the Very Fast Simulated Annealing [19] variant, this algorithm use the cost function $Cost$ as the thermodynamic system's energy. During the solutions' space probabilistic iteration, the acceptance of the current state is done such that new states with less energy compared to the previous energy are accepted; otherwise, the new state is accepted when the probability $exp\left(\frac{|F-F_{new}|}{temp}\right)$ is greater than a random generated float using a uniform distribution $U[0, 1]$. Also, with decreasing temperature process, the chance for the system to accept such penalizing transitions decreases. The temperature schedule in this algorithms is given by:

$$temp_k = temp_0 e^{\left(-0.5k^{\frac{1}{2\sigma_c}}\right)} \tag{35}$$

where k is the current iteration number and $temp_0$ is the initial temperature parameter. The detail of the solution is presented in Algorithm (4).

Algorithm 4 : SA-Based Container Transfer Algorithm with Path Selection (SA-PS)

Require: $\mathcal{C}$,$\mathcal{S}$,$\mathcal{P}$,Ω, $k^{max}$and $temp_0$.
Ensure: the Global solution (α^*,β^*);

1: Generate an initial solution (α,β)
2: Calculate F=$Cost(\alpha,\beta)$ according to (19);
3: $(\alpha^*,\beta^*,F^*)\leftarrow(\alpha,\beta,F)$
4: **for** n=1 to k^{max} **do**
5: $temp\leftarrow temp_0 e^{-0.5n^{\frac{1}{2\sigma_c}}}$;
6: $\alpha_{new}\leftarrow rand\_neighbour(\alpha)$;
7: Build best β_{new} using α_{new}
8: Calculate F_{new}=$Cost(\alpha_{new},\beta_{new})$ using (19);
9: $\Delta_F\leftarrow F_{new} - F$
10: **if** Δ_F<0 or $e^{\frac{-|\Delta_F|}{temp}} \geqslant random(0,1)$ **then**
11: $(\alpha,\beta,F)\leftarrow(\alpha_{new},\beta_{new},F_{new})$
12: **if** F<F^* **then**
13: $(\alpha^*,\beta^*,F^*)\leftarrow(\alpha,\beta,F)$
14: **end if**
15: **end if**
16: **end for**
17: **return** (α^*,β^*)

As input, Algorithm 4 requires the sets $\mathcal{C}$,$\mathcal{S}$ and $\mathcal{P}$; the parameters' vector Ω, the maximum iterations count parameter k^{max}, the initial temperature value $temp_0$. In lines 1 to 3, the initial solution's vectors are built and the optimal cost F^* associated with the optimal solution (α^*, β^*) is initialized. Then a for loop (line 4) is used in order to repeat the annealing process using k^{max} iterations. At each step, the temperature value $temp$ is updated (line 5); then, a neighboring state α_{new} of the current state α in line 6 is generated and its corresponding paths selection vector is built in line 7. Then, the new cost F_{new} is evaluated in line 8. Then, the new state is accepted if generating more profit; otherwise it is accepted using a probabilistic test(lines 10 to 15). Here, $random(0, 1)$ is a function's call that uniformly generates a random number in $[0, 1]$.

VI. Evaluation and Results

In this section, the proposed experiments used in order to compare the proposed solutions are presented based on the execution time and the cost function metrics.

A. Simulation Setup

All developed simulation programs were ran using a 2.4GHz Intel Core i5 processor in a PC with a maximum 8GB

of RAM. Moreover, the basic parameters of the simulation experiments are listed in Table III.

TABLE III. SIMULATIONS' PARAMETERS

Parameter	values
σ_s; σ_r	5; 3
$\lvert\mathcal{P}_{i,j}\rvert$	$[\![3; 5]\!]$
n^{max}; k^{max}	100; 200
q_0	0.3
Δ_ϕ^l; Δ_ϕ^g	0.1; 0.7
Δ_r;Δ_h	0.5; 0.5
Δ_p;Δ_n	0.5; 0.5
Δ_l;Δ_b	0.5; 0.5
ε_1	1
ε_2	2
$temp_0$	200

B. Exact vs. Heuristic Performance

To investigate the feasibility and limitation of Algorithm 1, the first experiment is carried where the achieved costs are measured and the execution time of all five solutions is recorded. In fact, the performance of the optimal BFS based solution is studied compared to the proposed heuristic solutions where the ACS-PS algorithm is studied relatively to both proposed strategies denoted ACS-PS-1 and ACS-PS-2. Accordingly, the containers' count (σ_c) is varied between 2 and a maximum feasible experimentation value $\sigma_c = 9$ while the nodes' count $\sigma_s = 5$, and $\lvert\mathcal{P}_{i,j}\rvert \in [\![3; 5]\!]$. The obtained results are depicted in Fig. 3.

The obtained normalized cost for the proposed solutions is

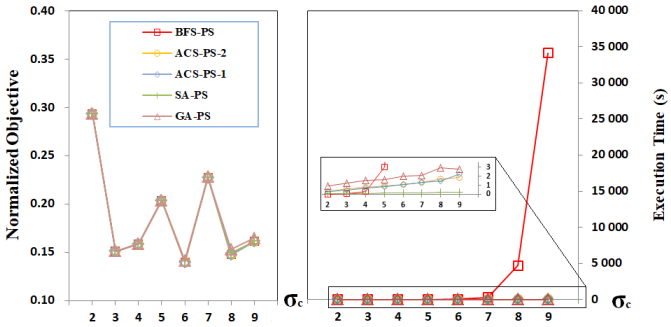

Fig. 3. Normalized Cost and Execution Time with σ_c; $\sigma_s = 5$.

shown in the left part of this figure. The variations of the same curve in this figure is only the result of using different data from one point to another and does not carry any information. Thus, the figure shows similar results for all solutions when $\sigma_c \in [2, 7]$; elsewhere the results of the GA-PS solution only deviate little from the optimal BFS-PS solution. The right side of this figure shows the variation of the execution time of the studied solutions. For clarity reason in this part of the figure, the results are zoomed to show the achievements of the heuristic solutions. Accordingly, the exponential growth of the BFS-PS solution execution times is demonstrated. Indeed, it achieves better performance for $\sigma_c \in [2, 4]$ compared to all other solution; elsewhere, it enormously goes beyond feasible times. For instance, it reaches $34123.15s$ with $\sigma_c = 9$. The SA-PS solution achieves the minimum execution times by little exceeding the achievements of the ACS-PS and GA-PS solutions. In fact, with $\sigma_c = 9$ it reaches only $0.192s$; whereas ACS-PS-1, ACS-PS-2 and GA-PS solutions respectively attain 2.242, 1.883 ans 2.784 seconds. This experiment shows a slightly stable execution time for the heuristic solutions and the infeasibility of the BFS-PS solution beyond the value $\sigma_c = 5$.

C. Heuristic Solutions Comparison

The second experiment studies the heuristic solutions' performance only. In this experiment, the containers' number (σ_c) is taken such that $\sigma_c \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. With regard to the total number of containers, Fig. 4 shows the achieved Normalized Cost obtained as the value of the objective function defined in Eq. (19). The results demonstrate the superiority in performance of the ACS-PS solution for both strategies. In particular, the ACS-PS-2 solution gives the best results compared to all other solutions for all values of σ_c. Also, the results of the solutions based on GA-PS and SA-PS are slightly bigger in that order compared to those of ACS-PS.

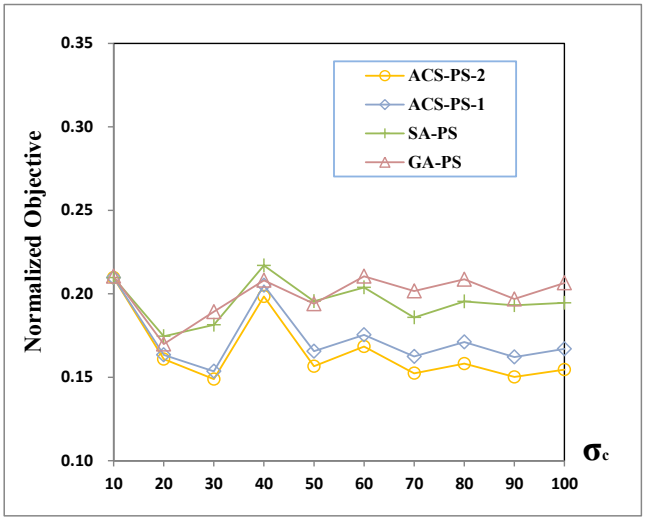

Fig. 4. Normalized Cost with σ_c; $\sigma_s = 5$.

D. The Total Backhaul Cost

Now, the next evaluation is introduced where the performance related to the total backhaul cost achievements for all heuristic solutions is studied. The reported values are obtained using Eq. (10). First, the experience is performed such that $\sigma_c \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ with $\sigma_s = 5$ and record the overall backhaul costs using ACS-PS, SA-PS and GA-PS heuristic methods. In each value of σ_c the overall achieved backhaul cost is sown without normalization using different settings. Thus, the variation shape of the same curve does not provide any information. Hence, Fig. 5 depicts the obtained results of this first experiment. Once again, the ACS-PS-2 solution gives the best results compared to all other solutions for all values of σ_c. Mainly, the performance results of the solutions based on ACS-PS widely exceed the performance of GA-PS and SA-PS although there is no clear and fixed preference between GA-PS and SA-PS in terms of results. Indeed, ACS-PS-1, ACS-PS-2, SA-PS and GA-PS attain respectively 0.221, 0.101, 1.244 and 1.203 for $\sigma_c = 100$,

whereas they attain 0.302, 0.169, 1.071 and 1.169 for $\sigma_c = 90$. Now, the following second part of the experiment studies the

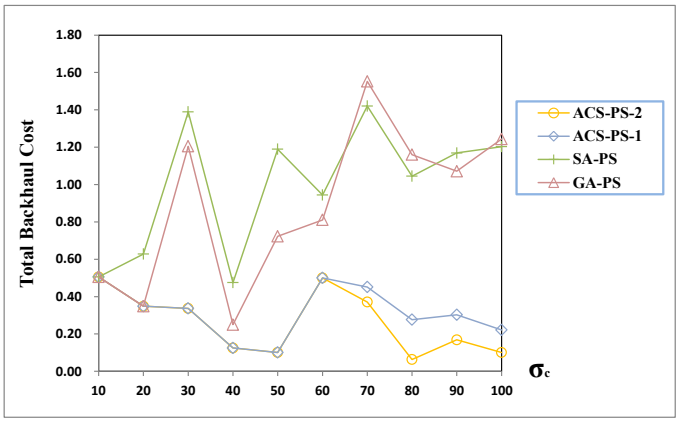

Fig. 5. Total Backhaul Costs with σ_c; $\sigma_s = 5$.

impact of the regulatory factor Δ_b on the result in terms of the total backhaul cost. Δ_b is taken in the interval $[0.05, 0.5]$ with the setting $\sigma_c = 30$; $\sigma_s = 5$; $\Delta_l = 1.0 - \Delta_b$. The obtained results are reported in Figure 6.

The balance effect is well observed from this figure. Indeed,

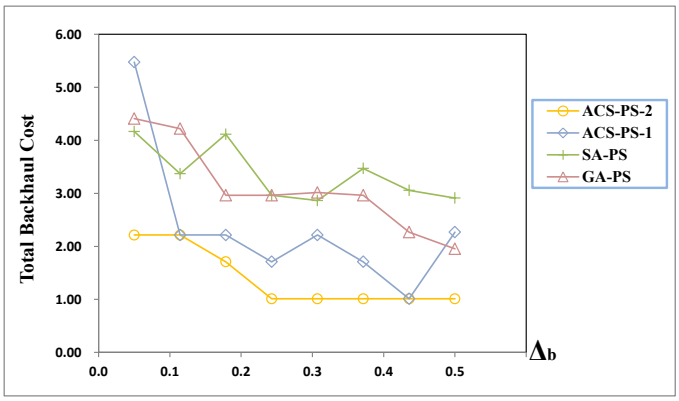

Fig. 6. Total Backhaul with Δ_b; $\sigma_c = 30$; $\sigma_s = 5$; $\Delta_l = 1.0 - \Delta_b$;.

when the value of Δ_b increases, the total backhaul cost generally decreases except for a few values where small tolerable increases are observed. these increases can be explained by the probabilistic aspect of these heuristic solutions which remain acceptable. The same figure further demonstrates superior performance of the ACS-PS-2 solution. Indeed, for this solution only, the variation of the total backhaul cost remains decreasing for all values of the experience. Consequently, this experiments shows that the factor Δ_b, used as regulator coefficient to balance the importance of the backhaul bandwidth cost among the other metrics, really fulfills its role.

VII. Conclusions and Perspectives

In this paper, a containers' deployment and migration problem with resource consideration within a multi-server mobile edge-cloud system is studied. The model considers a set of containers to deploy and migrate to a set of edge-cloud nodes where the transfer is compellable to users' backhaul bandwidth constraints. The formulated optimization problem minimizes a derived multi-objective function that jointly minimizes end-users perceived bandwidths and the system's load balance degree. Accordingly, the optimal transfer decisions are established by solving the obtained optimization problem. To handle its high complexity, two moderate complexity algorithms based respectively on Ant Colony System and Simulated Annealing are proposed. Then, a set of simulation experiments are performed to study their performance. The results reveal that the proposed BFS-based exact method is inefficient with big settings and it is highly time consuming. Furthermore, the ACS-PS is considerably efficient and gives good result with more acceptable execution time, whereas the SA-based solution is very efficient in terms of execution time. Moreover, the balance effect of the Δ_b factor serving to balance the importance degree of the backhaul cost is well established. Finally, we plan as perspectives to involve the transfer delays regarding the migrations types in the studied edge-cloud system.

References

[1] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.

[2] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.

[3] M. El Ghmary, M. O. Cherkaoui Malki, Y. Hmimz, and T. Chanyour, "Energy and computational resources optimization in a mobile edge computing node," in *2018 9th International Symposium on Signal, Image, Video and Communications (ISIVC)*, pp. 323–328, IEEE, 2018.

[4] Y. Zhao, W. Wang, Y. Li, C. C. Meixner, M. Tornatore, and J. Zhang, "Edge computing and networking: A survey on infrastructures and applications," *IEEE Access*, vol. 7, pp. 101213–101230, 2019.

[5] H. Abdah, J. P. Barraca, and R. L. Aguiar, "Qos-aware service continuity in the virtualized edge," *IEEE Access*, vol. 7, pp. 51570–51588, 2019.

[6] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23511–23528, 2018.

[7] F. A. Salaht, F. Desprez, and A. Lebre, "An overview of service placement problem in fog and edge computing," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–35, 2020.

[8] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Transactions on Vehicular Technology*, 2020.

[9] J. Li, X. Shen, L. Chen, D. P. Van, J. Ou, L. Wosinska, and J. Chen, "Service migration in fog computing enabled cellular networks to support real-time vehicular communications," *IEEE Access*, vol. 7, pp. 13704–13714, 2019.

[10] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, 2018.

[11] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, and H. Flinck, "Fast service migration in 5g trends and scenarios," *IEEE Network*, vol. 34, no. 2, pp. 92–98, 2020.

[12] W. Wang, S. Ge, and X. Zhou, "Location-privacy-aware service migration in mobile edge computing," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2020.

[13] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on markov decision process," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1272–1288, 2019.

[14] F. Tang, C. Liu, K. Li, Z. Tang, and K. Li, "Task migration optimization for guaranteeing delay deadline with mobility consideration in mobile edge computing," *Journal of Systems Architecture*, p. 101849, 2020.

[15] Z. Ma, S. Shao, S. Guo, Z. Wang, F. Qi, and A. Xiong, "Container migration mechanism for load balancing in edge network under power internet of things," *IEEE Access*, vol. 8, pp. 118405–118416, 2020.

[16] L. Ma, S. Yi, N. Carter, and Q. Li, "Efficient live migration of edge services leveraging container layered storage," *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 2020–2033, 2018.

[17] C. Ling, W. Zhang, H. He, and Y.-c. Tian, "Network perception task migration in cloud-edge fusion computing," *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–16, 2020. https://doi.org/10.1186/s13677-020-00193-8.

[18] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial life*, vol. 5, no. 2, pp. 137–172, 1999.

[19] D. Pei, J. A. Quirein, B. E. Cornish, D. Quinn, and N. R. Warpinski, "Velocity calibration for microseismic monitoring: A very fast simulated annealing (vfsa) approach for joint-objective optimization," *Geophysics*, vol. 74, no. 6, pp. WCB47–WCB55, 2009.