Deployment of Application Microservices in Multi-Domain Federated Fog Environments

Francescomaria Faticanti[*,‡], Marco Savi[†,*], Francesco De Pellegrini[◇], Petar Kochovski[⋆], Vlado Stankovski[⋆] and Domenico Siracusa[*]

***Abstract*—In this paper we consider the problem of initial resource selection for a single-domain fog provider lacking sufficient resources for the complete deployment of a batch of IoT applications. To overcome resources shortage, it is possible to lease assets from other domains across a federation of cloud-fog infrastructures to meet the requirements of those applications: the fog provider seeks to minimise the number of external resources to be rented in order to successfully deploy the applications' demands exceeding own infrastructure capacity. To this aim, we introduce a general framework for the deployment of applications across multiple domains of cloud-fog providers while guaranteeing resources locality constraints. The resource allocation problem is presented in the form of an integer linear program, and we provide a heuristic method that explores the resource assignment space in a breadth-first fashion. Extensive numerical results demonstrate the efficiency of the proposed approach in terms of deployment cost and feasibility with respect to standard approaches adopted in the literature.**

***Index Terms*—Fog Computing, Microservices, Federation, Resources Allocation, Virtual Network Embedding, Locality Constraints.**

I. Introduction

Fog computing is a novel paradigm proposed as an effective means to offload computation at the network's edge by leveraging concepts and functionalities of cloud computing [1]. The main objective of this technology is to close the gap between cloud and IoT domains, by empowering the elaboration of data much closer to the objects where it is produced through the adoption of well-designed applications. Such an approach brings tangible benefits, spanning from service latency reduction [2] to bandwidth savings [3].

Nowadays, one of the main trends in the design and development of cloud-native applications are microservice-oriented service architectures [4]. Microservice-oriented applications consist of a cascade of loosely-coupled components/modules (that is, the microservices) that can be containerized independently. Each microservice performs specific computations on input data and forwards the resulting output to other microservices downstream for further processing. Microservice design is preferred to monolithic application development because applications adopting such an architecture deliver same intended functionalities but attain higher degree of flexibility, reliability and scalability [5]. In fog computing, on the other hand, available resources are geographically spread so that a microservice-oriented architecture appears a natural choice for fog-native applications, because it is possible to split them into components to be executed at the edge or in the cloud, respectively.

Even though the diffusion of fog computing is still at its infancy, some fog-like solutions have been already commercialized [6][7]. However, the main drawback of existing platforms is that they are bound to work *within a single administrative domain*. Hence, they require exclusive ownership of resources spanning from cloud to things. Nevertheless, given the wide geographical diffusion and heterogeneity of fog computing resources, single-domain solutions are likely to be unfit for the deployment of fog-native applications. In fact, they come – by their own nature – with some hard-constraining *locality* requirements, dictated by the geographical location where some processing needs to be executed.

This work contributes to current research efforts towards the definition of a multi-domain federated fog ecosystem, where fog providers (i.e., owners of fog resources) can stipulate contracts among them to rent additional resources (otherwise not accessible) and ensure smooth execution of their own applications. Whilst the technical feasibility of such approach – requiring the interface to a specific resource brokerage platform – has been already proven [8][9], a different angle is considered in this work. We take the perspective of a fog provider and pose the following question: *what resources should I rent to ensure that a given set of applications is correctly deployed in the federated fog while minimising external resource usage?*

To answer this question, we assume that the fog provider implements a mechanism (e.g. based on matchmaking [10]) to retrieve from the resource brokerage platform a set of potential rentable resources owned by other providers, based, e.g., on applications' locality needs. Given such a set of potential rentable resources and the self-owned resources, the objective is to find a feasible application deployment to minimises the cost of external resource rental, while deploying over the federated fog environment as many applications as possible to maximise the provider's revenues.

By exploiting integer linear programming (ILP) methods, we formally formulate the *application deployment* problem: the aim is to allocate enough federated fog resources to satisfy the applications' requirements while self-owned resources are used as much as possible. More formally, an application deployment is a *map* of the applications' components, namely microservices and data flows among them, to the available fog resources. Finding such an optimal map is a *virtual network embedding (VNE) problem*, which is known to be NP-hard [11][12]. Hence, a practically viable approach is to design algorithmic solutions consisting of tailored heuristics to approximate an optimal solution. We thus propose a scalable heuristic algorithm solving the application deployment problem while taking into account locality constraints. Experimental results validate our solution with respect to the optimal deployment and a state-of-the-art approach in terms of deployment cost and feasibility percentage.

A. Related work and main contributions

Service placement in federated clouds has been deeply studied in the literature [13][14], exploring dynamic and adaptive solutions for the allocation of resources. In [15], authors proposed a distributed and adaptive approach for service placement on a distributed and heterogeneous cloud federation. Authors of [16] described a multi-objective optimization for resource allocation able to address changes

[*]Fondazione Bruno Kessler, Italy, [‡]University of Trento, Italy, [†]University of Milano-Bicocca, Italy, [◇]University of Avignon, France, [⋆]University of Ljubljana, Slovenia. This work has received funding from the EU H2020 R&I Programme under Grant Agreement no. 815141 (DECENTER: Decentralised technologies for orchestrated Cloud-to-Edge intelligence).

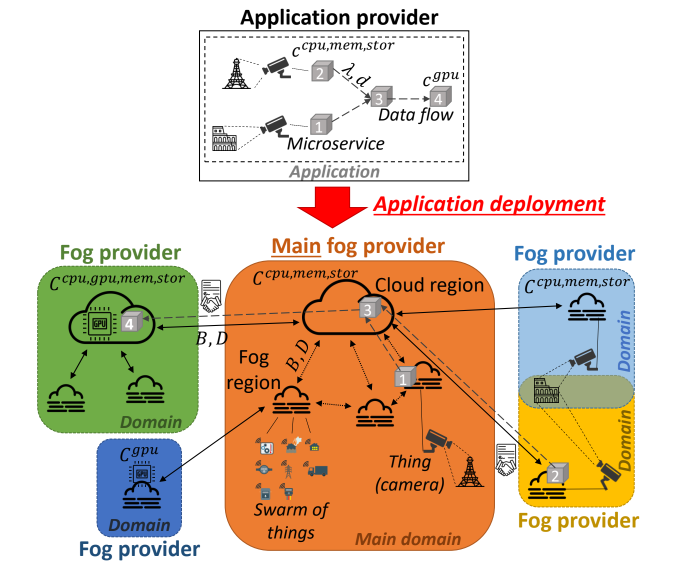

Fig. 1. Application deployment in a multi-domain federated fog ecosystem.

in the applications' behaviour. Although these problems are crucial for the dynamic deployment of applications in federated clouds, in that context IoT locality constraints are not considered as application requirements. But, such kind of constraints are what distinguishes the most a federated fog from a federated cloud environment, besides the higher heterogeneity of available resources. Our approach considers locality constraints in the deployment of fog applications in the said fog federation. Furthermore, the applications considered in the aforementioned works are monolithic whereas, as already mentioned, the most promising paradigm for the design of fog applications is the microservice-oriented architecture. With this assumption, the application deployment becomes a virtual network embedding problem, an NP-hard problem for which many heuristic solutions have been proposed in literature [17], especially with respect to Virtual Network Function (VNF) placement [11]. Such problems are usually solved by greedily deploying one virtual network at the time. In our context, due to locality constraints, it is instead recommended to consider the whole batch of application graphs at once. We do so with a Breadth-First Search (BFS) visit driven by the applications' topological order, an approach that provably reduces the cost of deployment while ensuring higher feasibility rates with respect to existing solutions, especially under hard locality constraints.

The remainder of the paper is structured as follows. In the next section we introduce the reference scenario and the system model, whereas in Sec. III we provide the formalisation of the resource allocation problem. The proposed algorithmic solution is described in Sec. IV, whereas its numerical performance evaluation is reported in Sec. V. A concluding section ends the paper.

II. System Model

A. Involved stakeholders and scenario

Figure 1 resumes the scenario envisioned in this work. Two main stakeholders are involved:

Application provider: it designs a *microservice-oriented application* that can be used by its customer(s) to accomplish some specific tasks. As an example, consider an application for number-plate-based car access control to a restricted traffic zone. The customer (a municipality) requires an application that can access a well-located camera, convert the number plate from the video stream into text, store the number plate in a database and compare it with authorized number plates. Such kind of application can be easily designed following a microservice-oriented architecture and would clearly benefit from video stream computation close to the camera, e.g., to reduce bandwidth needs and improve privacy.

Fog provider: it owns a *fog infrastructure* that can host different microservice-oriented applications. A fog infrastructure can either be geographically distributed (spanning from cloud to edge) or confined to a specific geographical area (no cloud, only edge). It (i) provides computational, memory and storage capacity and (ii) can ensure access to things deployed on specific areas and owned by the fog provider, if any (e.g. video cameras or IoT sensors). We refer to a fog infrastructure owned by a fog provider as *fog domain* (or *domain*).

As already mentioned, in our investigated scenario we take the perspective of a fog provider – called *main fog provider* from now on – that wants to deploy multiple applications over its fog infrastructure. However, depending on the applications to be deployed, it may happen that the fog provider's infrastructure alone cannot meet the applications' requirements. This may happen for two reasons, that is:

Locality: an application requires some processing capability in a specific geographical location, e.g. it requires video streaming acquisition and elaboration from a camera owned by another fog provider.

Peculiar resource usage: an application requires the usage of some peculiar resources that are not owned (or do not suffice for an appropriate application execution) by the main fog provider. For example, an application may need a GPU for fast processing, but no GPU is owned by such a provider.

Throughout the paper we assume that the main fog provider can rely on a multi-domain federated fog ecosystem (and its related infrastructure), where resources from other fog providers can be rented to meet application requirements – including locality and peculiar resource usage – otherwise exceeding own capacity. Figure 1 depicts this reference scenario. In the following, the model (i) for the aforementioned multi-domain federated fog infrastructure are presented and (ii) for the applications to be deployed.

B. Multi-domain federated fog infrastructure modelling

The considered infrastructure consists of several geographically distributed fog or cloud *regions* belonging to different fog domains, including the main fog domain. From a modelling point of view, the only difference between cloud and fog regions is their processing, memory and storage capabilities. Each region is composed by multiple *hosts*, which are the atomic unit where a microservice can be deployed. An example of such an infrastructure is shown in Figure 1.

The multi-domain infrastructure is described by a weighted graph $G_I = (V_I, E_I)$ where V_I is the set of regions of the infrastructure and $E_I \subseteq V_I \times V_I$. Furthermore, cost function $w : V_I \to \mathbb{R}^+$ specifies the cost for application deployment in a specific region of the infrastructure. Nodes and links/edges composing the infrastructure are also called *physical nodes* and *physical links/edges*. Each physical link $(u, v) \in E_I$ is characterized by a couple $(D_{u,v}, B_{u,v})$ modelling the latency and the bandwidth capacity of the link, respectively. We assume that hosts within a region are interconnected through high-performance connections, where bandwidth is never a bottleneck and latency can be considered negligible. Conversely, regions (either belonging to different domains or within the same domain of the federation) are interconnected through best-effort network connections, meaning that bandwidth and latency constraints must be considered.

C. Application modelling

We denote $\mathcal{A}$ as the set of applications to be deployed on the infrastructure and each application $A \in \mathcal{A}$ is described by a weighted DAG

TABLE I
MAIN NOTATION USED THROUGHOUT THE PAPER.

Symbol	*Meaning*
$G_I = (V_I, E_I)$	Infrastructure network graph: V_I physical nodes (regions), E_I physical edges (network connections)
$\mathcal{A}$	Set of applications to be deployed on G_I
$G_A = (V_A, E_A)$	Graph for application $A \in \mathcal{A}$: V_A virtual nodes (modules) and E_A are virtual edges (data flows)
$L_A \subseteq V_A$	Subset of virtual nodes to be deployed on a specific physical node of the infrastructure
$r_A : L_A \rightarrow V_I$	Maps virtual node v_A to a given physical node
$c^r_{v_A}$	Resource requirements of virtual node $v_A \in V_A$, with $r \in \{cpu, gpu, mem, stor\}$
$\lambda_A(u_A, v_A)$	Max. throughput on edge $(u_A, v_A) \in E_A$
$d_A(u_A, v_A)$	Max. tolerated delay on edge $(u_A, v_A) \in E_A$
S_v	Set of available hosts in physical node $v \in V_I$
$C^r_{v,i}$	Resource capacity of the i-th host in physical node $v \in V_I$, with $r \in \{cpu, gpu, mem, stor\}$
$D_{u,v}$	Latency on the physical link $(u, v) \in E_I$
$B_{u,v}$	Capacity of the physical link $(u, v) \in E_I$
$w(v)$	Cost of physical node $v \in V_I$
P	Set of computed paths p between any couple of physical nodes
$P_{u,v} \subseteq P$	Set of computed paths between $v \in V_I$ and $u \in V_I$
s_p, d_p	First and last node of path $p \in P$

$G_A = (V_A, E_A)$, where V_A is the set of microservices (or modules, or components) composing the application and E_A represents the set of microservices dependencies. Each directed edge (u_A, v_A) of an application A is characterised by (i) the maximum throughput generated on that link, $\lambda_A(u_A, v_A)$ and (ii) the maximum tolerated delay on that link, $d_A(u_A, v_A)$. Each node v_A of an application A has computational requirements in terms of CPU, GPU, memory and storage, $c^r_{v_A}$, where $r \in \{cpu, gpu, mem, stor\}$. Furthermore, each application has a set of locality constraints representing the regions where some microservices must be deployed to acquire data from specific devices belonging to that regions. We model this fact by introducing a set $L_A \subseteq V_A$ for each application $A \in \mathcal{A}$. This set contains all the nodes that need to acquire data from a tagged device placed on a specific region: hence, they must be placed in that region. Function r_A specifies for each node in L_A the region where it should be deployed. Nodes and links/edges composing an application are also called *virtual nodes* and *virtual links/edges*.

III. PROBLEM FORMULATION

In this section we provide the ILP formulation of the resource allocation problem.

Decision variables. We introduce two types of variables:
1) A binary variable for the assignment of each application module to a physical node:

$$x^{A,v_A}_{v,i} = \begin{cases} 1, & \text{if module } v_A \text{ of application } A \\ & \text{is deployed on host } i \text{ of node } v \\ 0, & \text{otherwise} \end{cases}$$

where $A \in \mathcal{A}$, $v_A \in V_A$, $v \in V_I$ and $i \in S_v$.
2) A binary variable for the assignment of virtual links of each application to physical paths:

$$y^{(u_A,v_A)}_p = \begin{cases} 1, & \text{if link } (u_A, v_A) \text{ is mapped to path } p \\ 0, & \text{otherwise} \end{cases}$$

where $A \in \mathcal{A}$, $(u_A, v_A) \in E_A$, and $p \in P$.

Objective Function. Given a subset of nodes belonging to the main fog provider, we want to minimise the total *deployment cost* for all the application requests of that provider. We assign a weight w for the deployment of a virtual node onto a physical node v of the fog infrastructure. We assume that weights are larger when such deployment is performed towards nodes belonging to other fog domains, since their resources need to be rented. More formally, we have a weight function defined for each physical node, $w : V_I \rightarrow \mathbb{R}^+$. Finally, the objective function writes as follows

$$\sum_{A\in\mathcal{A}} \sum_{v_A \in V_A} \sum_{v \in V_I} \sum_{i \in S_v} w(v)\, x^{A,v_A}_{v,i}. \quad (1)$$

Constraints. First, we have integrality constraints on the applications' deployment on the fog infrastructure. In fact, all modules of an application must be deployed and each such module must be placed only once. These conditions are expressed through the following constraint (2) and (3), respectively.

$$\sum_{v_A \in V_A} \sum_{v \in V_I} \sum_{i \in S_v} x^{A,v_A}_{v,i} = |V_A|, \quad \forall A \in \mathcal{A}, \quad (2)$$

$$\sum_{v \in V_I} \sum_{i \in S_v} x^{A,v_A}_{v,i} = 1, \quad \forall A \in \mathcal{A}, \forall v_A \in V_A. \quad (3)$$

Second, we have the constraints on the resource capacity for each host in the infrastructure:

$$\sum_{A\in\mathcal{A}} \sum_{v_A \in V_A} c^{res}_{v_A} x^{A,v_A}_{v,i} \leq C^r_{v,i} \quad (4)$$

where $v \in V_I$, $i \in S_v$, and resource $r \in$ {cpu, gpu, mem, stor}.

Third, there are constraints related to virtual and physical links capacity. We start by introducing the structural constraints binding variables related to nodes and virtual links:

$$\sum_{p \in P_{u,v}} y^{(u_A,v_A)}_p \leq \sum_{i \in S_u} x^{A,u_A}_{u,i}, \quad (5)$$

$$\sum_{p \in P_{u,v}} y^{(u_A,v_A)}_p \leq \sum_{j \in S_v} x^{A,v_A}_{v,j}, \quad (6)$$

$$\sum_{p \in P_{u,v}} y^{(u_A,v_A)}_p \geq \sum_{i \in S_u} x^{A,u_A}_{u,i} + \sum_{j \in S_v} x^{A,v_A}_{v,j} - 1, \quad (7)$$

where $(u, v) \in V_I \times V_I$, $A \in \mathcal{A}$, and virtual link $(u_A, v_A) \in E_A$.

Constraints (5), (6) and (7) ensure that a unique physical path is used by a virtual link whenever the virtual nodes connected by such link are the extreme nodes of the path.

The following are constraints on the bandwidth capacity for all physical links $(u, v) \in E_I$

$$\sum_{A\in\mathcal{A}} \sum_{(u_A,v_A) \in E_A} \sum_{p \in P:(u,v) \in p} \lambda_A(u_A, v_A)\, y^{(u_A,v_A)}_p \leq B_{u,v}, \quad (8)$$

and the applications' delay constraints, namely

$$y^{(u_A,v_A)}_p \sum_{(u,v) \in p} D_{u,v} \leq d_A(u_A, v_A), \quad (9)$$

where $A \in \mathcal{A}$, $(u_A, v_A) \in E_A$, and $p \in P$.

Finally, we have locality constraints: they impose the placement of a subset of nodes of the applications on specific regions, since data is acquired by devices located on those specific regions:

$$\sum_{i \in S_{r_A(v_A)}} x^{A,v_A}_{r_A(v_A),i} = 1, \quad \forall A \in \mathcal{A}, \forall v_A \in L_A. \quad (10)$$

IV. Heuristic Algorithms

The problem formulated in the previous section is a VNE problem which, as already said, is a well-known NP-hard problem. Hence, it is important to look for fast and scalable algorithms whose output solutions have an acceptable cost.

The VNE problem has been extensively studied in the literature, and several heuristic solution methods have been proposed [17]. In the following, we briefly resume the main idea of the most adopted greedy approaches, which are based on a Depth-First Search (DFS) of the deployment space. We part from this approach and propose a solution tailored for the specific requirements of our problem, based instead on a Breadth-First Search method.

A. Depth-First Search Approach

The most popular heuristic methods for VNE adopt a greedy embedding procedure. It receives as input a sorted list of application deployment requests and returns a mapping between each request and a subset of physical nodes/edges of the infrastructure. Our reference DFS-based algorithm is an adaptation of the one presented in [18].

For each application in the input batch, the following three steps are executed sequentially:
1) *Topological sorting of the application graph*: it is obtained by performing a visit of the application graph; as a result, an order of the DAG nodes is established.
2) *Virtual node mapping*: it defines, for each virtual node, a set of possible placements. Once defined, selects the nodes placement.
3) *Virtual link mapping*: it finds a path between each couple of physical nodes where virtual nodes are mapped in the previous step.

The DAG-based deployment order on the application's modules as established in the first step accounts for their dependencies. A topological sort, indeed, ensures that for each couple of nodes $u_A, v_A \in V_A$, if $(u_A, v_A) \in E_A$ then u_A precedes v_A in the topological order, that is u_A is a *predecessor* of v_A. Once the order is defined by topological sort, the application's node mapping to the infrastructure's resources is done. At this stage, node mapping accounts for the actual resources occupation, and requires to establish a certain priority function; it defines, for each application, an admissible set for each module of the application. If this set contains at least one region, it selects the region $v \in V_I$ that maximises:

$$res_{CPU}(v) \left\{ \sum_{u|(u,v)\in E_I} res_{BW}(u,v) + \sum_{u|(v,u)\in E_I} res_{BW}(v,u) \right\}, \tag{11}$$

where $res_{BW}(u,v)$ and $res_{CPU}(v)$ are the residual bandwidth of the physical link (u,v) and the total residual CPU capacity in region v, respectively [18]. If the set of admissible regions is empty for at least one module of the application, such application cannot be deployed. Instead, if a mapping for all the modules of the application is found, the next step is the link mapping procedure. For each virtual edge, the algorithm takes the two physical nodes assigned to the virtual nodes it connects (chosen by the second step) and considers the least-congested path that satisfies both bandwidth and delay requirements of the virtual link. If all the virtual links can be mapped to a set of paths of the infrastructure, the algorithm has found a complete map of the application onto the physical infrastructure. This operation is iterated for all the applications to be deployed.

B. Breadth-First Search Approach

The basic idea of our novel approach is to deploy the batch of applications in a breadth-first search fashion. This means that we do not deploy a single application at the time, conversely, at each step we consider all the application nodes: at each iteration we consider for placement the first virtual node of each application, as determined by the topological sorting of the application's graph. Every time a virtual node is mapped to a host, it is popped up from the stack of the topological order of its application. The rationale of this procedure is that applications' locality constraints can be better matched. Indeed, a depth-first greedy procedure can quickly saturate all the resources of a particular region for the deployment of certain applications without considering that some applications in the batch may have a hard locality constraint on that region. This typically renders the deployment of the whole batch unfeasible, as will be better highlighted in Section V.

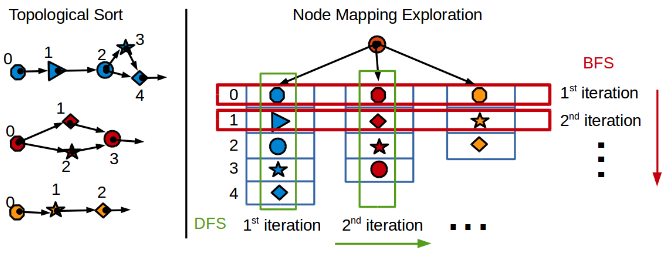

Fig. 2. DFS vs. BFS approaches after the topological sorting step.

Given a batch of applications to be deployed on the multi-domain infrastructure, our algorithm consists of three steps:
1) *Sorting of the batch of applications*: it sorts all the application on the basis of their total bandwidth consumption.
2) *Topological sorting of the application graph*: it establishes a topological sort of each application graph following the applications' order decided in the previous step. A stack is created for each application: the first node of the topological order is put on its top.
3) *Virtual node and link mapping*: it iterates over all the applications in a breadth-first manner, exploring all of them jointly and level by level. At each iteration, all the virtual nodes on top of the applications' stacks are popped up and a node mapping is performed in the order established by step 2: each module is assigned to a host of the infrastructure. Link mapping is then performed in conjunction with node mapping. Figure 2 illustrates the iterations of the third step of the algorithm comparing this breadth-first approach with respect to a depth-first visit.

More in detail, the selection of the regions to perform the virtual node mapping (step 3) is operated by choosing for deployment a set of admissible regions that takes into account the requirements and the locality constraints of each application module. Once such set is defined, the algorithm selects the regions with the lowest *deployment cost*. Afterwards, to choose the most suitable region, it greedily selects the placement option that has the smaller relative increment of occupied nodes' resources with respect to CPU, GPU, memory, storage and bandwidth.

Finally, to perform link mapping, each time an application node v_A is mapped to a region R, the algorithm takes the list of all the regions assigned to the predecessors (in the topological order of A) of v_A and selects the least-congested paths between them and R.

V. Performance Evaluation

In this section we validate our solution for applications deployment on a multi-domain federated infrastructure. Our goal is twofold: (i) prove that a Depth-First Search approach for the deployment of applications can negatively impact on either the *feasibility* of the solution (i.e., the deployment of the batch of applications can be

TABLE II
APPLICATIONS' MICROSERVICES REQUIREMENTS [12].

Requirement	*Mean Value*	*Range*
CPU ($c^{cpu}_{v_A}$)	1250 MIPS	$[500, 2000]$ MIPS
Memory ($c^{mem}_{v_A}$)	1.2 Gbytes	$[0.5, 2]$ Gbytes
Storage ($c^{stor}_{v_A}$)	3.5 Gbytes	$[1, 8]$ Gbytes
Throughput (λ_A)	3 Mbps	$[1, 5]$ Mbps
Delay (d_A)	262.5 ms	$[25, 500]$ ms

incomplete) or on its *optimality*; (ii) show that our Breadth-First Search approach leads to a good trade-off between feasibility and optimality, especially when some locality constraints are specified. We measure the *feasibility* as the percentage of instances that admit a feasible solution (*i.e.,* meet all the constraints), for the problem described in Section III, among all the generated instances.

A. Simulation settings

We describe how we generate the test network topologies and the batch of applications to be deployed. Note that in our performed experiments we consider only *CPU*, *memory* and *storage* as computational resources within any region and as resource requests by applications (that is, we do not consider GPU). In fact, a GPU requirement can be expressed in terms of MIPS (Million Instructions Per Second) in the same way as it is done for CPU: a GPU request can thus be translated to a CPU requirement with high MIPS.

Network topology: the multi-domain fog infrastructure is modelled as a directed network graph with a number of fog regions K and a number of domains D. For the main fog domain, we consider a central cloud region that is always connected to fog regions in a star topology. For each randomised topology realisation, links among different fog regions (either belonging to the main fog provider or to other external providers) are instead added according to an Erdős-Rényi random graph model, where a link between two regions exists with probability $pr = 0.5$. Eventually, each link in the resulting topology is assigned an average bandwidth of 60 Mbps and an average delay of 10 ms representing the average values of modern communication links [12]. The hosts available within each region belong to three classes, depending on the resources they are equipped with, namely *low* (CPU: 5000 MIPS, memory: 2 GB, storage: 60 GB), *medium* (CPU: 15000 MIPS, memory: 8 GB, storage: 80 GB) and *high* (CPU: 44000 MIPS, memory: 16 GB, storage: 120 GB). To well dimension the computational resources within any region, the aggregated demand of the batch of applications to be deployed (in terms of CPU, memory and storage) is equally split among all the regions excluding the main cloud. Then, the set of hosts of a certain region is generated by iteratively and randomly choosing hosts of different types until the aggregated demand fraction assigned to that region is satisfied.

Application batch: a batch of applications $\mathcal{A}$ is generated for each experiment; we consider $|\mathcal{A}| = \{10, 15, 20, 25, 30\}$. The demand of each applications' module in terms of CPU, storage, memory and throughput are uniform independent random variables. The distribution values for each microservice are enlisted in Table II. Each application is generated as a DAG ordering all the nodes and adding an edge only between predecessors and successors.

We developed a Python-based simulator for the evaluation of the above algorithms since well-known fog simulators do not support scenarios with multiple domains and fog regions yet [19]. The Gurobi solver has been used to solve the optimal placement ILP problem (*OPT*). Each data point in the shown graphs is the average value over 30 randomized instances, where the infrastructure is fixed and

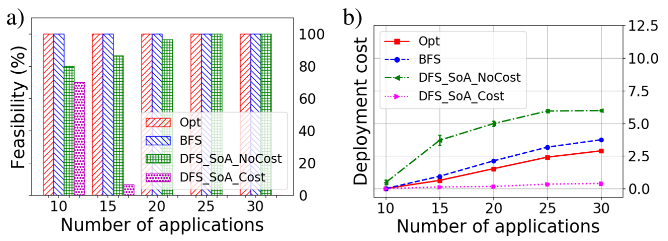

Fig. 3. Feasibility-optimality tradeoff. a) Feasibility percentage; b) Total deployment cost for each application.

the batch of applications and host distribution change. All the results are shown with their corresponding 95% confidence interval. We evaluate the proposed solutions in a scenario with $K = 6$ regions, $D = 3$ domains. The optimization problem is solved from the *main* fog provider perspective: such domain contains one cloud and one fog region, while the other fog regions are distributed among the remaining fog providers' domains (*external* providers). We consider a unit cost for all the deployments outside the main domain ($w = 1$) and zero-cost for the deployments inside the main domain ($w = 0$). To highlight the importance of *locality constraints*, we impose that the first module of each application must reside in the fog region of the main domain.

B. Numerical results

In Figure 3 we evaluate the tradeoff between optimality and feasibility of the proposed solutions. Figures 3a) and b) report on feasibility and optimality with two variants of the state-of-the-art DFS approach. *DFS_SoA_NoCost* represents the variant where *deployment cost optimization* is not taken into account: this means that the objective of this strategy is just maximising the number of deployed applications. Conversely, *DFS_SoA_Cost* objective is to minimise the deployment cost. *BFS* refers instead to our proposed novel strategy. From Figure 3a), we can notice that *OPT*, *BFS* and *DFS_SoA_NoCost* have high feasibility. Especially, the first two strategies have a feasibility of 100%, meaning that they are able to deploy the complete batch of applications in all the 30 randomized instances (note that *feasibility* refers to the percentage of instances that are feasible).

Additionally, looking at the deployment cost (computed as in eq. 1) in Figure 3b), we can see that *BFS* offers a solution very close to the optimal one (*OPT*). *DFS_SoA_NoCost*, on the other hand, leads to a high deployment cost even though it has a good feasibility percentage. Conversely, in *DFS_SoA_Cost* the feasibility percentage is low as well as the deployment cost. This behaviour is reasonable given the greedy nature of the DFS approach. Indeed, if we include a cost optimization in such an approach, the algorithm prioritizes all the regions with the lowest cost (that is, the regions in the main domain) for the deployment of all the applications' modules. In this manner, resources with lower cost are quickly saturated precluding the possibility to satisfy the locality constraints for the applications that have not been deployed yet. On the other side, if we do not consider cost optimization, all the regions are treated in the same way, increasing the chance of having a feasible solution while increasing the deployment cost too. From this perspective, a BFS approach is beneficial since it does not evaluate the deployment of each application at the time, but it considers the deployment of a part of every application at each iteration. Thanks to this property, this method leads to a high feasibility percentage, since it helps guarantee locality constraints, and to a strong reduction of the deployment cost.

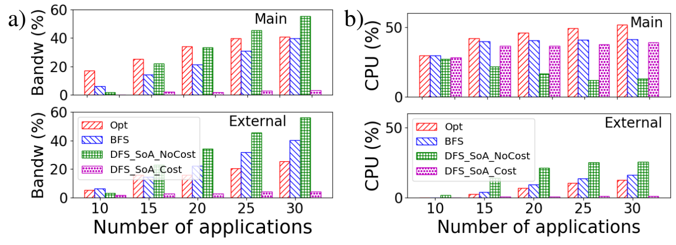

Fig. 4. Bandwidth and CPU usage. a) Percentage of bandwidth usage within the main domain and in external domains; b) CPU usage in the main domain and in external domains.

TABLE III
EXECUTION TIME (SEC).

$\|\mathcal{A}\|$	*OPT*	*BFS*	*DFS_SoA_NoCost*	*DFS_SoA_Cost*
10	3.58	0.03	0.02	0.02
15	26.90	0.05	0.05	0.03
20	40.30	0.07	0.07	0.03
25	60.72	0.09	0.09	0.03
30	79.10	0.13	0.12	0.04

In summary, Figure 3 confirms that our solution explores the best tradeoff between optimality and feasibility with respect to both variants of DFS.

Figure 4a) reports on the bandwidth usage of the proposed solutions within and outside the main domain. The bandwidth consumption of *DFS_SoA_Cost* is almost constant and low in both the main and external domains since it tries to avoid the deployment of applications towards external domains and mostly deploy applications on the main fog region until it becomes saturated. *OPT* and *BFS* solutions present instead a similar trend on bandwidth usage as the size of the application batch increases. The bandwidth consumption of the link between the main cloud and fog region is slightly higher for *OPT* than for *BFS* since the optimal deployment can accommodate more applications in the main domain. With respect to external domains, the two solutions behave the opposite confirming the slightly higher cost of the *BFS* solution. Finally, *DFS_SoA_NoCost* presents a similar behaviour on both main and external domains, since it maximises relation (11) without distinguishing between main/external domains.

In Figure 4b) we report the percentage of CPU usage of all the proposed solutions on the main domain (upper figure) and on external domains. Reasonably, the CPU usage of *OPT* is slightly greater than of *BFS* in the main domain since it deploys more applications there, as confirmed by Figure 3b). *DFS_SoA_Cost*, given its greedy nature, presents a high and constant percentage of CPU usage in the main domain, meanwhile it has very low CPU consumption in external domains. On the other hand, the *DFS_SoA_NoCost* has the opposite trend, showing an increasing usage from external domains and decreasing usage from the main domain as the number of applications increases. Note that memory and storage usage have a similar trend as CPU usage and thus are not reported in this section for the sake of conciseness.

Finally, Table III reports on the average execution time of the proposed solutions over the 30 instances. The values of *OPT* refer to executions that are stopped after 5 minutes if the solver has not completed the computation in that time range. The higher scalability of all heuristic approaches is apparent compared to *OPT*. Note that *DFS_SoA_Cost* has lower execution time than the other approaches because its execution is generally stopped earlier, i.e., when the algorithm cannot deploy one of the applications and the deployment is considered infeasible. The time efficiency of heuristic methods is given by their polynomial time complexity: indeed, we can see the deployment of the batch of applications as a visit of a graph composed by a root dummy node connected with an edge to all the subgraphs represented by the applications, as shown in Figure 2. In this problem, given the existence of locality constraints, a breadth-first visit results to be more efficient in terms of both feasibility and optimality.

VI. CONCLUSIONS

We have considered the problem of deploying fog applications onto a federated cloud-fog environment. In this context, solving the problem of initial resource selection is crucial to reduce offloading costs, satisfy all the applications' requirements and accommodate future requests. By considering a microservice paradigm for fog applications, a virtual network embedding problem is faced, which is known to be NP-hard. As a consequence, standard heuristic solutions need to trade-off feasibility, cost-efficiency and scalability. This work proposed a new deployment technique for batches of fog applications, based on a breadth-first visit of all the applications' graphs to deal efficiently with locality constraints. It has been shown to provide a near-optimal performance and yet excellent feasibility rate, outperforming standard depth-first greedy heuristics. In future works, we shall extend the proposed framework to account for transactions between different domains, paving the way to the design of new exchange mechanisms for fog computing.

REFERENCES

[1] A. V. Dastjerdi and R. Buyya, "Fog Computing: Helping the Internet of Things Realize Its Potential," *IEEE Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[2] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog Computing: Towards Minimizing Delay in the Internet of Things," in *IEEE International Conference on Edge Computing (EDGE)*, 2017.

[3] C. C. Byers, "Architectural Imperatives for Fog Computing: Use cases, Requirements, and Architectural Techniques for Fog-Enabled IoT Networks," *IEEE Communications Mag.*, vol. 55, no. 8, pp. 14–20, 2017.

[4] I. Nadareishvili, R. Mitra, M. McLarty *et al.*, "Microservice architecture: aligning principles, practices, and culture," O'Reilly Media Inc., 2016.

[5] Y. Gan and C. Delimitrou, "The Architectural Implications of Cloud Microservices," *IEEE Computer Architecture Letters*, vol. 17, no. 2, pp. 155–158, 2018.

[6] "AWS IoT Greengrass," https://aws.amazon.com/greengrass/.

[7] "Azure IoT Edge," https://azure.microsoft.com/en-us/services/iot-edge/.

[8] E. Carlini, M. Coppola, P. Dazzi *et al.*, "BASMATI: Cloud Brokerage Across Borders for Mobile Users and Applications," in *Springer Advances in Service-Oriented and Cloud Computing Workshop*, 2018.

[9] M. Savi, D. Santoro, K. Di Meo *et al.*, "A Blockchain-based Brokerage Platform for Fog Computing Resource Federation," in *Conference on Innovation in Clouds, Internet and Networks (ICIN)*, 2020.

[10] X. Li, H. Ma, F. Zhou, and X. Gui, "Service Operator-Aware Trust Scheme for Resource Matchmaking across Multiple Clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1419–1429, 2015.

[11] X. Cheng, S. Su, Z. Zhang *et al.*, "Virtual Network Embedding through Topology-aware Node Ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, p. 38–47, 2011.

[12] A. Brogi, S. Forti, and A. Ibrahim, "How to Best Deploy Your Fog Applications, Probably," in *IEEE International Conference on Fog and Edge Computing (ICFEC)*, 2017.

[13] B. Rochwerger, D. Breitgand, E. Levy *et al.*, "The Reservoir Model and Architecture for Open Federated Cloud Computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 1–4, 2009.

[14] A. J. Ferrer, F. Hernandez, J. Tordsson *et al.*, "OPTIMIS: A Holistic Approach to Cloud Service Provisioning," *Elsevier Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, 2012.

[15] E. Carlini, M. Coppola, P. Dazzi, M. Mordacchini *et al.*, "Self-optimising Decentralised Service Placement in Heterogeneous Cloud Federation," in *IEEE International Conference on Self-adaptive and Self-organizing Systems (SASO)*, 2016.

[16] R. G. Aryal and J. Altmann, “Dynamic Application Deployment in Federations of Clouds and Edge Resources using a Multiobjective Optimization AI Algorithm,” in *IEEE International Conference on Fog and Mobile Edge Computing (FMEC)*, 2018.

[17] H. Cao, H. Hu, Z. Qu *et al.*, “Heuristic Solutions of Virtual Network Embedding: A Survey,” *China Communications*, vol. 15, no. 3, pp. 186–219, 2018.

[18] M. Yu, Y. Yi, J. Rexford *et al.*, “Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[19] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh *et al.*, “iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments,” *Wiley Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.