Optimising QoS-assurance, Resource Usage and Cost of Fog Application Deployments

Antonio Brogi[1], Stefano Forti[1], and Ahmad Ibrahim[2]

[1] Department of Computer Science, University of Pisa, Italy
{brogi, stefano.forti}@di.unipi.it
[2] School of Computer Science, University of Birmingham, UK
a.ibrahim@bham.ac.uk

Abstract. Identifying the best application deployment to distribute application components in Fog infrastructures – spanning the IoT-to-Cloud continuum – is a challenging task for application deployers. Indeed, it requires fulfilling all application requirements, whilst determining a trade-off among different objectives (i.e., QoS assurance, Fog resource consumption and cost), resulting in a complex and time-consuming decision-making process to be tuned manually. In this paper, we present a simple multi-objective optimisation scheme that permits selecting the best placement of application components, balancing the trade-off among QoS-assurance, Fog resource consumption and monthly deployment costs. We exploit our prototype, extended with parallel Monte Carlo simulations, and a motivating example to show how IT experts can benefit from our approach.

Keywords: Fog computing, application deployment, multi-objective optimisation, cost models, Monte Carlo method.

1 INTRODUCTION

Fog computing [5] aims at moving computation closer to the source of IoT data by relying on a multitude of heterogeneous devices (e.g., personal devices, gateways, micro-data centres, embedded servers) spanning the continuum from the Cloud to the IoT[3]. Meanwhile, modern applications are made from a set of independently deployable components (or services, or micro-services) that interact together and must meet some requirements. The interactions (component-component or component-Things) may have firm Quality of Service (QoS) requirements – latency, bandwidth – to be fulfilled for the application to work as expected [16].

Deployment of Fog applications requires placing their components (e.g., control loops, operational support, business intelligence) over the available infrastructure that consists of Cloud, Fog and IoT devices (spread over a possibly large geographical area, and inter-connected via heterogeneous communication technologies), based upon specific application and user requirements. Determining eligible deployments of multi-component applications to given Fog infrastructure is proved to be NP-hard [7].

[3] Hereinafter, the word *Things* is used to refer to IoT devices, both sensors and actuators.

Naturally, financial considerations play a role too in deployment selection as industry and businesses usually aim at maximising their revenues, whilst minimising deployment operational costs [41]. If Cloud offerings are limited to a few large providers, Fog computing envisions many other small and medium players (e.g., single Fog node or Things owners) that will offer virtual instances or IoT capabilities at different pricing schemes, making it more challenging to identify cost-effective deployments among the possible ones.

Overall, there is a need for tools that can actually support application deployment to the Fog and that should feature *(i) QoS-awareness* to achieve latency reduction, bandwidth savings and to enforce business policies, *(ii) context-awareness* to suitably exploit local and remote resources, *(iii) cost-awareness* to enact cost-effective deployments.

To support all these objectives, we developed a prototype (FogTorchΠ) [7–9] that given a Fog infrastructure (1) determines application deployments that meet all (hardware, software IoT and QoS) requirements, (2) estimates their *QoS-assurance* and *Fog resource consumption* by simulating latency and bandwidth variations of communication links as per given probability distributions, and (3) estimates the monthly cost of application deployments over the input infrastructure. It is worth noting that, even after a set of eligible deployments has been identified, the application deployers still have to identify a best candidate deployment. Trading-off among different metrics like QoS assurance, Fog resource consumption and cost can make this complex decision making process time-consuming and difficult to be tuned manually (since some constraints are orthogonal to each other). Indeed, suppose application deployers aim at minimising deployment cost and Fog resource consumption for a given application deployment. Such preferences may lead their choice to a candidate deployment that minimises the two considered metrics, missing to actually guarantee a good level of compliance to QoS-requirements (i.e., QoS-assurance). With the support of trade-off analysis tools, application deployers might discover some other eligible deployments that can provide a more balanced compromise between all considered metrics.

In this paper, we extend the work of [9] – in which we introduced a cost model to estimate monthly deployment cost of Fog applications – so to include a simple multi-objective optimisation scheme that permits selecting the best placement of application components, balancing the trade-off among QoS-assurance, Fog resource consumption and monthly costs. With respect to [9] we also extended FogTorchΠ to perform parallel Monte Carlo simulations when estimating deployment QoS-assurance, thus taming the complexity of the described algorithms. We show, over a motivating example, how these extensions of our methodology, particularly the multi-objective optimisation, can further help IT experts (or new businesses coming onto the Fog market) in deciding how to distribute application components to Fog infrastructures in a QoS-, context- and cost-aware manner.

The rest of this paper is organised as follows. After introducing a motivating example of a smart building application (Section 2), we briefly describe FogTorchΠ (Section 3) and present the cost model extension (Section 4). Then, we discuss multi-objective optimisation (Section 5), present the results obtained by applying the extended version of FogTorchΠ methodology to the motivating example (Section 6), and discuss some related work (Section 7). Finally, we draw some concluding remarks (Section 8).

2 MOTIVATING EXAMPLE

A simple Fog application (Figure 1) manages fire alarm, heating and A/C systems, interior lighting, and security cameras of a smart building and is made from three microservices:

- IoTController, directly controlling the connected cyber-physical systems,
- DataStorage, storing all sensed data for future use and employing machine learning techniques to update sense-act rules at the IoTController so to optimise heating and lighting management based on previous experience and/or on people behaviour, and
- Dashboard, aggregating and visualising collected sensor data and videos, as well as providing an interface to users interacting with the application.

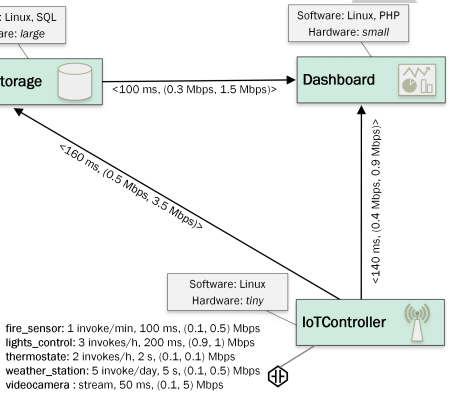

Fig. 1. Fog application of the motivating example as in [9].

Each microservice is an independently deployable component of the application [39] and has some hardware and software requirements to be fulfilled in order to function properly (the grey box associated with each component). Hardware requirements are expressed in terms of the virtual machine (VM) types[4] that will host the component once deployed. Table 1 lists all VM types used in this example and the corresponding hardware specification.

Furthermore, end-to-end communication links supporting component-component interactions need to feature suitable latency and bandwidth (e.g., the latency between IoTController and DataStorage should be less than 160 ms and the free bandwidth should be at least 0.5 Mbps download and 3.5 Mbps upload[5]). Finally, interactions between the IoTController and Things are subject to similar constraints, and also specify the expected sampling rate for the component to query Things at runtime.

[4] Adapted from OpenStack Mitaka flavours: `https://docs.openstack.org/`.
[5] Arrows on the links in Figure 1 indicate the upload direction.

Table 1. Hardware specification for different VM types [9].

VM Type	vCPUSs	RAM (GB)	HDD (GB)
tiny	1	1	10
small	1	2	20
medium	2	4	40
large	4	8	80
xlarge	8	16	160

System integrators in charge of deploying the smart building application for one of their customers have two Cloud data centres, three Fog nodes and nine Things (Figure 2) available in the target infrastructure. The deployed application will have to utilise the Things connected to Fog 1 and the weather_station_3 at Fog 3. For the system integrators, deploying components to Fog 2 involves no additional cost since their customers own that node, and can use it free of charge.

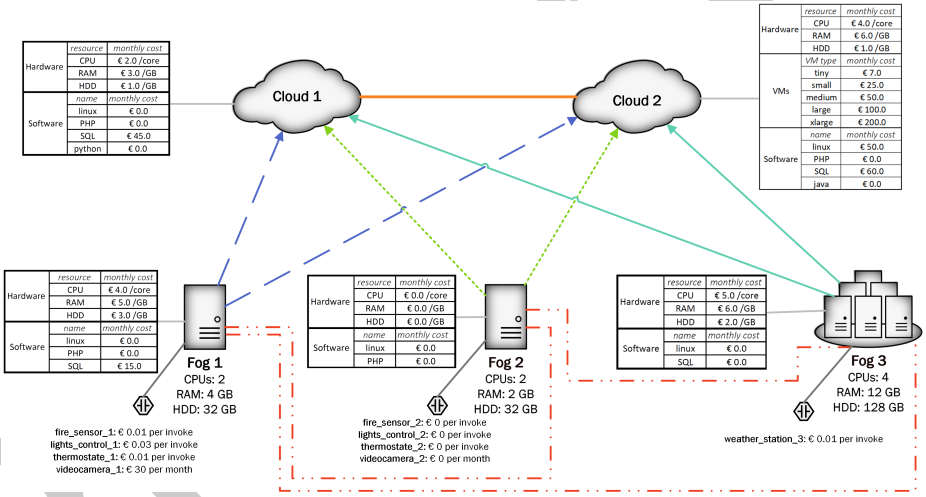

Fig. 2. Fog infrastructure of the motivating example as in [9].

All Fog and Cloud nodes are associated with their pricing schemes. Those scheme consider the possibility of either buying a ready-made instance of a certain VM type (e.g., a *small* instance at Cloud 2 costs €25 per month) as well as the possibility of assembling on-demand instances (by selecting the required number of cores and the needed amount of memory and storage to support a given component).

Fog nodes offer software capabilities with limited hardware resources (i.e., RAM, HDD, CPUs), as indicated in Figure 2. Cloud nodes also offer software capabilities and we assume that they offer potentially unbounded hardware resources (under the assumption that one can always purchase extra or larger instances on-demand).

QoS profiles of the available communication links[6] are listed in Table 2. They are based on real data[7] and represented as probability distributions to account for QoS variations. **Green** color links at Fog 2 initially feature a 3G Internet access. We assume Fog and Cloud nodes are able to access directly connected Things as well as Things (or the data they produce) at neighbouring nodes via a specific middleware layer (through the associated communication links) which is in accordance with the current technical proposals (e.g., [5] and [42]).

Table 2. QoS profiles of communication links as in [9].

Dash Type	Profile	Latency	Download	Upload
	Satellite 14M	40 ms	98%: 10.5 Mbps 2%: 0 Mbps	98%: 4.5 Mbps 2%: 0 Mbps
	3G	54 ms	99.6%: 9.61 Mbps 0.4%: 0 Mbps	99.6%: 2.89 Mbps 0.4%: 0 Mbps
	4G	53 ms	99.3%: 22.67 Mbps 0.7%: 0 Mbps	99.4%: 16.97 Mbps 0.6%: 0 Mbps
	VDSL	60 ms	60 Mbps	6 Mbps
	Fibre	5 ms	1000 Mbps	1000 Mbps
	WLAN	15 ms	90%: 32 Mbps 10%: 16 Mbps	90%: 32 Mbps 10%: 16 Mbps

The system integrators, planning to sell the deployed solution for €1,500 a month, set the limit of the monthly deployment cost at €850. On the other hand, the customer is willing to pay them only if the application can comply to the specified QoS requirements at least 98% of the time. Then, interesting questions for the system integrators before the first deployment of the application are, for instance:

> **Q1(a)** — *Is there any eligible deployment of the application reaching the needed Things at Fog 1 and Fog 3, and meeting the financial (at most €850 per month) and QoS-assurance (at least 98% of the time) constraints mentioned above?*
>
> **Q1(b)** — *Which eligible deployment represent the most balanced trade-off optimising QoS-assurance, Fog resource consumption and monthly deployment cost of the smart building application?*

Suppose that with an extra monthly investment of €20, system integrators can exploit a 4G connection at Fog 2. Then:

> **Q2** — *Does the upgrade from 3G to 4G at Fog 2 make it possible to determine a deployment with better trade-off on QoS-assurance, Fog resource consumption and monthly deployment cost?*

In Section 6, we will show how the FogTorchΠ – suitably extended with the cost model described in Section 4 – can be exploited, along with multi-objective optimisation techniques, to obtain answers to all the above questions.

[6] Arrows on the links in Figure 2 indicate the upload direction.

[7] Satellite: `https://www.eolo.it/`, 3G/4G: `https://www.agcom.it`, VDSL: `http://www.vodafone.it`.

3 OVERVIEW OF FogTorchΠ

FogTorchΠ [8] is an open-source Java prototype[8] that permits describing Fog infrastructures and applications so to determine QoS-, context-, and cost-aware application deployments. Before detailing the cost-aware extension to FogTorchΠ, we summarise the overall functioning of our prototype. FogTorchΠ takes as input:

1. a *Fog infrastructure I*, with the specification of the Fog and Cloud nodes available for deployment (each with its hardware, software and IoT capabilities), the probability distributions of the network QoS (viz., latency, bandwidth) featured by end-to-end communication links interconnecting such nodes[9], and the cost for purchasing sensed data and Cloud/Fog virtual instances,
2. a *multi-component application A*, specifying all hardware (i.e., CPU, RAM, storage), software (i.e., OS, libraries, frameworks) and IoT requirements (i.e., which type of Things to exploit) of each component, and the minimum QoS (i.e., latency and bandwidth) needed to suitably support component-component and component-Thing interactions at runtime,
3. a *Things binding* ϑ, mapping each IoT requirement of an application component to an actual Thing in *I*, and
4. a *deployment policy* $\delta(\gamma)$, white-listing the nodes where component γ of *A* can be deployed[10] in accordance to security or business-related considerations.

Based on such input, FogTorchΠ determines all eligible deployments of the components of *A* to Cloud or Fog nodes in *I*. An *eligible deployment* Δ maps each component γ of *A* to a Cloud or Fog node *n* of *I* so that (1) $n \in \delta(\gamma)$ and it meets the hardware and software requirements of γ, (2) hardware resources are enough to deploy *all* components of *A* simultaneously mapped to *n*, (3) Things exploited by γ (and specified in ϑ) are reachable from *n*, and (4) component-component and component-Thing interactions mapped to the same end-to-end communication link do not exceed the available bandwidth and meet their latency requirements.

FogTorchΠ relys on the Monte Carlo method [22] to estimate the *QoS-assurance* of output deployments, by aggregating the eligible deployments obtained when varying the QoS of communication links according to the associated probability distributions for latency and bandwidth. Figure 3 lists the pseudocode of FogTorchΠ functioning. First, an empty dictionary *D* is created (line 2) to contain key-value pairs $\langle \Delta, \texttt{counter} \rangle$, where the key ($\Delta$) represents an eligible deployment and the value (`counter`) keeps track of how many times Δ will be generated during the Monte Carlo simulation. Then, at the beginning of each run of the simulation, a new state I_s of the infrastructure is sampled based on the probability distributions of the QoS of the communication links in *I* (line 4).

The function FINDDEPLOYMENTS(A, I_s, ϑ, δ) (line 5) employs an exhaustive (backtracking) search [7] to determine the set *E* of eligible deployments Δ of *A* to I_s, i.e.

[8] Available at `https://github.com/di-unipi-socc/FogTorchPI/tree/multithreaded/`.

[9] Actual implementations in Fog landscapes can rely on monitoring tools (e.g., [6], [23]) to update the information available on *I*.

[10] When δ is not specified for a component γ of *A*, γ can be deployed to any compatible node in *I*.

```
1:  procedure MONTECARLO(A, I, ϑ, δ, n)
2:      D ← ∅                                          ▷ dictionary of ⟨Δ, counter⟩
3:      parallel for n times do
4:          I_s ← SAMPLELINKSQOS(I)
5:          E ← FINDDEPLOYMENTS(A, I_s, ϑ, δ)
6:          D ← UNIONUPDATE(D, E)
7:      end parallel for
8:      for Δ ∈ keys(D) do
9:          D[Δ] ← D[Δ]/n
10:     end for
11:     return D
12: end procedure
```

Fig. 3. Pseudocode of the parallel Monte Carlo simulation in FogTorchΠ.

deployments of A that meet all hardware, software, IoT, and QoS requirements in that particular state of the infrastructure. In order to tame the worst-case exponential complexity of such search step, the current version of FogTorchΠ features a multi-thread implementation of the *for* loop of lines 3–7, which assigns n/w runs to each of the w threads available on the system and executes them in parallel afterwards [10].

The objective of the Monte Carlo step is to look for eligible deployments in different underlying network conditions. At the end of each run, the set E of eligible deployments of A to I_s is used to update D. The function UNIONUPDATE(D, E) (line 6) updates D by adding deployments $\langle \Delta, 1 \rangle$ discovered during the last run ($\Delta \in E \setminus keys(D)$) and by incrementing the `counter` of those deployments that had already been found in a previous run ($\Delta \in E \cap keys(D)$).

After the simulation has run for a significantly large number of times ($n \geq 100,000$), the QoS-assurance of each deployment $\Delta \in keys(D)$ is obtained by dividing the `counter` associated to Δ by n (lines 8–10). Thus, the QoS-assurance is the percentage of runs a certain deployment Δ was output by FINDDEPLOYMENTS(A, I_s, ϑ, δ). Such percentage offers an estimate on how likely Δ is to satisfy all QoS constraints of A, against variations in the communication links as per their historical behaviour. Finally, dictionary D is output (line 11).

Each output deployment Δ also contains information about its Fog resource consumption, which is computed during the search as the aggregate percentage averaging RAM and HDD consumed over the set F of all Fog nodes[11]:

$$\frac{1}{2}\left(\frac{\sum_{\gamma \in A} RAM(\gamma)}{\sum_{f \in F} RAM(f)} + \frac{\sum_{\gamma \in A} HDD(\gamma)}{\sum_{f \in F} HDD(f)}\right)$$

The next section details the cost model exploited by FogTorchΠ, which permits predicting the monthly deployment cost of output deployments. As for the resource consump-

[11] FogTorchΠ permits to compute Fog resource consumption also on a specified subset of Fog nodes $\overline{F} \subset F$.

tion, FINDDEPLOYMENTS(A, I_s, ϑ, δ) computes and associates each eligible deployment Δ with an estimate of its monthly cost[12], which we detail in the next section.

4 COST MODEL

Our cost model extends to Fog computing previous efforts in Cloud VM cost modelling [21], and includes software costs, and costs typical of the IoT [41]. With respect to related work, which – to the best of our knowledge – only exploited linear cost models based on unit cost for different types of hardware resource, we also consider the possibility of purchasing bundle offers at the IoT, the Cloud and the Fog layer (i.e., for data, hardware and software).

A hardware offering H, available at any Cloud or Fog node n, can be either a *default VM* (Table 1) featuring a fixed monthly price or an *on-demand VM*, assembled by selecting any amount of processors (CPU), memory (RAM) and storage (HDD). By assuming that $R = \{CPU, RAM, HDD\}$ is the set of resources considered when assembling on-demand instances, our cost model estimates the monthly cost for a hardware offering H at node n as

$$p(H,n) = \begin{cases} c(H,n) & \text{if } H \text{ is a default VM} \\ \sum\limits_{\rho \in R} [H.\rho \times c(\rho,n)] & \text{if } H \text{ is an on-demand VM} \end{cases}$$

where $c(H,n)$ is the monthly cost of a default VM H at Fog or Cloud node n, whilst $H.\rho$ indicates the amount of resource $\rho \in R$ used by[13] the on-demand VM represented by H, and $c(\rho,n)$ is the unit monthly cost at n for resource ρ.

Analogously, a software offering S at any Cloud or Fog node n can be either a ready-made *bundle* or an *on-demand* subset of the software capabilities offered by n (each purchasable as a separate item). The monthly cost for S at node n is estimated as

$$p(S,n) = \begin{cases} c(S,n) & \text{if } S \text{ is a bundle} \\ \sum\limits_{s \in S} c(s,n) & \text{if } S \text{ is on-demand} \end{cases}$$

where $c(S,n)$ is the price for the software bundle S at node n, and $c(s,n)$ is the monthly cost of a single software s at n.

Finally, in Sensing-as-a-Service [43] scenarios, a Thing offering T exploiting an actual Thing t can be made available at a monthly *subscription* fee (i.e., covering a certain amount of data exchanges) or through a *pay-per-invocation* mechanism (i.e., per exchanged message)[35]. Hence, the cost for offering T at Thing t can be estimated as

$$p(\mathrm{T},t) = \begin{cases} c(\mathrm{T},t) & \text{if T is subscription based} \\ \mathrm{T}.k \times c(t) & \text{if T is pay-per-invocation} \end{cases}$$

[12] Cost computation is performed *on-the-fly*. This is done during the search step, considering the possibility to rely on the cost prediction as a heuristic to lead backtracking towards best candidate deployments.

[13] Bounded by the maximum amount purchasable from any chosen Cloud or Fog provider.

where $c(\mathrm{T},t)$ is the monthly subscription fee for T at t, while $\mathrm{T}.k$ is the number of monthly invocations expected over t and $c(t)$ is the cost per invocation at t (including Thing exploitation and/or data transfer costs).

In what follows, we assume that Δ is an eligible deployment for an application A to an infrastructure I, as defined in Section 3. In addition, let $\gamma \in A$ be a component of the considered application A, and let $\gamma.\overline{\mathcal{H}}$, $\gamma.\overline{\Sigma}$ and $\gamma.\overline{\Theta}$ be its hardware, software and Things requirements, respectively. By summing up all the presented pricing schemes, the monthly cost for a given deployment Δ can be first approximated as:

$$cost(\Delta,\vartheta,A) = \sum_{\gamma\in A}\left[p(\gamma.\overline{\mathcal{H}},\Delta(\gamma)) \;+\; p(\gamma.\overline{\Sigma},\Delta(\gamma)) \;+\; \sum_{r\in\gamma.\overline{\Theta}} p(r,\vartheta(r))\right]$$

This first estimate of the monthly deployment cost, however, does not feature a way to match application (hardware, software and IoT) requirements to "best" (Cloud, Fog and IoT) offerings at chosen nodes or Things. Particularly, it may lead the choice always to on-demand and pay-per-invocation offerings when the application requirements do not match exactly default or ready-made offerings, or when a Cloud provider does not offer a particular VM type (e.g., starting its offerings from *medium*). This may incur in overestimating monthly deployment costs.

As an example, consider the infrastructure of Figure 2 and this hardware requirements $r = \{\texttt{CPU}:1,\texttt{RAM}:1\texttt{GB},\texttt{HDD}:20\texttt{GB}\}$ of a component to be deployed to Cloud 2. Since no exact matching between the requirement and an offering at Cloud 2 exists, this first version the current cost model selects an on-demand instance, and estimates a cost of €30[14]. However, Cloud 2 also provides a *small* instance that can accommodate the component requirements at a (lower) cost of €25.

Indeed, larger VM types always satisfy smaller hardware requirements, bundled software offerings may satisfy multiple software requirements at a lower price, and subscription-based Thing offerings can be more or less convenient depending on the number of invocations on a given Thing. Therefore, some mechanism must be adopted to choose the "best" offering that can fulfil all software, hardware and Thing requirements of an application component. In what follows, we propose a small refinement to our cost model, necessary to capture this important aspect.

A *requirement-to-offering matching policy* $p_m(r,n)$ matches hardware or software requirements r of a component ($r \in \{\gamma.\overline{\mathcal{H}},\gamma.\overline{\Sigma}\}$) to the estimated monthly cost of the offering that will support them at Cloud or Fog node n, and a Thing requirement $r \in \gamma.\overline{\Theta}$ to the estimated monthly cost of the offering that will support r at Thing t.

Overall, this refinement to our cost model permits estimating the monthly cost of Δ including a cost-aware matching between application requirements and infrastructure offerings (for hardware, software and IoT), that are chosen as per p_m. Hence, we get:

$$cost(\Delta,\vartheta,A) = \sum_{\gamma\in A}\left[p_m(\gamma.\overline{\mathcal{H}},\Delta(\gamma)) \;+\; p_m(\gamma.\overline{\Sigma},\Delta(\gamma)) \;+\; \sum_{r\in\gamma.\overline{\Theta}} p_m(r,\vartheta(r))\right]$$

The cost-aware version of FogTorchΠ, including the described cost model, exploits a *best-fit lowest-cost* policy for choosing hardware, software and Thing offerings. In-

[14] €30 = 1 CPU × €4/core + 1 GB RAM × €6/GB + 20 GB HDD × €1/GB

deed, it selects the cheapest between the first default VM (from *tiny* to *xlarge*) that can support $\gamma.\overline{\mathcal{H}}$ at node n and the on-demand offering built as per $\gamma.\overline{\mathcal{H}}$. Similarly, software requirements in $\gamma.\overline{\Sigma}$ are matched with the cheapest compatible version available at n, and Thing per invocation offer is compared to monthly subscription so to select the cheapest[15] offering possible. The requirement-to-offering matching policy used in FogTorchΠ can be defined formally as:

$$p_m(\overline{\mathcal{H}},n) = \min\{p(H,n) \mid H \in \{\text{default VMs, on-demand VM}\} \wedge\ H \models \overline{\mathcal{H}}\}$$

$$p_m(\overline{\Sigma},n) = \min\{p(S,n) \mid S \in \{\text{on-demand, bundle}\}\ \wedge\ S \models \overline{\Sigma}\}$$

$$p_m(r,t) = \min\{p(T,t) \mid T \in \{\text{subscription, pay-per-invocation}\}\ \wedge\ T \models r\}$$

where $O \models R$ reads as offering *O satisfies requirements R*.

It is worth noting that the proposed cost model is general enough to include both IaaS and PaaS Cloud offerings since it separates the cost of purchasing virtual instances from the cost of purchasing software. Furthermore, even if we referred to VMs as the only deployment unit for application components, a straightforward extension to the model can account for other types of virtual instances (e.g., containers).

5 MULTI-OBJECTIVE OPTIMISATION

Exploiting the cost model of Section 4, it is naturally possible to minimise the obtained estimate for the monthly deployment cost and choose a candidate application deployment accordingly. However, decision makers in Fog application deployment processes (like the system integrators in Section 2), have often to determine a best placement of application components as a trade-off among various – sometimes orthogonal – requirements and metrics.

As described in Section 3 and 4, FogTorchΠ methodology enables to predict QoS-assurance, Fog resource consumption and monthly cost of eligible application deployments, i.e. those meeting application hardware, software and QoS constraints. Each of the mentioned metrics represents an objective that application deployers aim at optimising along with the others. Indeed, we are facing a multi-objective optimisation (MOO) problem [19].

MOOs are generally defined as

$$\min_{\Delta} F(\Delta) = [f_1(\Delta), f_2(\Delta), \cdots, f_m(\Delta)]$$

$$\text{subject to: } \Delta \in \overline{D}$$

where $f_i(\cdot)$ denote a set objective functions and $\overline{D}$ denote a feasible design space in which $F(\cdot)$ should be optimised. Usually, as it can be in our Fog application deployment

[15] Other policies are also possible such as, for instance, selecting the largest offering that can accommodate a component, or always increasing the component's requirements by some percentage (e.g., 10%) before selecting the matching.

scenarios, no solution Δ exists that minimises all set objectives at the same time. In this situations, Pareto optimal solutions are looked for. A solution is Pareto optimal when it is not possible to improve (at least) one objective function without making another worst. The set of all Pareto optimal solutions forms a so-called Pareto frontier.

A common aggregation technique to solve MOOs and to determine the best trade-off among all set objectives is the *linear weighted sum* method [37]. Such strategy allows to convert MOOs into single objective problems by linearly combining their (normalised) objective functions. As we will show over our motivating example in the next section, this permits to determine a best solution deployment, also accounting for the user preferences specified for some of the objectives (e.g., the minimum QoS-assurance of 98% and the maximum monthly cost of €850 required by system administrators) which specify the feasible design space $\overline{D}$. Last but not least, under the assumptions that functions weights are positive, the found solution is Pareto optimal.

Therefore, in this work, similarly to [27], given a deployment Δ, we will try to optimise the objective function

$$r(\Delta) = \sum_{f_i \in F} \omega_i \cdot \widehat{f_i(\Delta)}$$

where F is the set of m metrics to be optimised, ω_i is the weight[16] assigned to each metrics (so that $\sum_{i=1}^{m} \omega_i = 1$) and $\widehat{f_i(\Delta)}$ is the normalised value of the objective function $f_i(\cdot)$ for deployment Δ, which – given the set D of candidate deployments – is computed as:

- $\widehat{f_i(\Delta)} = \frac{f_i(\Delta) - \min_{d \in D}\{f_i(d)\}}{\max_{d \in \overline{D}}\{f_i(d)\} - \min_{d \in D}\{f_i(d)\}}$ when the $f_i(\Delta)$ is to be maximised, and
- $\widehat{f_i(\Delta)} = \frac{\max_{d \in D}\{f_i(d)\} - f_i(\Delta)}{\max_{d \in D}\{f_i(d)\} - \min_{d \in D}\{f_i(d)\}}$ when $f_i(\Delta)$ is to be minimised.

Since, in this case, we assumed that the higher the value of $r(\Delta)$ the better is deployment Δ, we will choose $\overline{\Delta}$ such that $r(\overline{\Delta}) = \max_{\Delta \in D}\{r(\Delta)\}$.

Considering our problem, deployers will most probably aim at maximising QoS-assurance, whilst minimising monthly deployment costs (in which we include the cost for the 4G connection at Fog 2 when needed). However, different system integrators may want to either maximise or minimise the Fog resource consumption of their deployment, i.e. they may look for *Fog-ward* or for *Cloud-ward* deployments. Hence, in the next section, we will consider both situations and compare the results obtained when trying to maximise Fog resource usage (i.e., Fog-ward deployments) to those obtained when trying to minimise it (i.e., Cloud-ward deployments).

6 MOTIVATING EXAMPLE (CONTINUED)

In this section, we discuss the results of running the cost-aware version of FogTorchΠ over the smart building example of Section 2. We rely on the linear weighted sum

[16] For the sake of simplicity, we assume here $\omega_i = \frac{1}{|F|} = \frac{1}{m}$, which can be tuned differently depending on the needs of the application operator.

method presented in Section 5 to answer all questions coming from the system integrators.

FogTorchΠ outputs the eligible deployments (as per Section 3) along with their predicted QoS-assurance, Fog resource consumption and monthly deployment cost (as per Section 4). Table 3 lists all eligible deployments output by FogTorchΠ, entries indicated with a * are only output when 4G is available at Fog 2.

Table 3. Eligible deployments generated by FogTorchΠ for **Q1** and **Q2**[17] as in [9].

Dep. ID	IoTController	DataStorage	Dashboard
Δ1	Fog 2	Fog 3	Cloud 2
Δ2	Fog 2	Fog 3	Cloud 1
Δ3	Fog 3	Fog 3	Cloud 1
Δ4	Fog 2	Fog 3	Fog 1
Δ5	Fog 1	Fog 3	Cloud 1
Δ6	Fog 3	Fog 3	Cloud 2
Δ7	Fog 3	Fog 3	Fog 2
Δ8	Fog 3	Fog 3	Fog 1
Δ9	Fog 1	Fog 3	Cloud 2
Δ10	Fog 1	Fog 3	Fog 2
Δ11	Fog 1	Fog 3	Fog 1
Δ12*	Fog 2	Cloud 2	Fog 1
Δ13*	Fog 2	Cloud 2	Cloud 1
Δ14*	Fog 2	Cloud 2	Cloud 2
Δ15*	Fog 2	Cloud 1	Cloud 2
Δ16*	Fog 2	Cloud 1	Cloud 1
Δ17*	Fog 2	Cloud 1	Fog 1

Figure 4 shows a 3D-plot of the same output deployments along the predicted metrics (i.e., the objective functions) that FogTorchΠ can estimate for them.
Before continuing, we recall the questions posed by the system integrators in Section 2:

Q1(a) — *Is there any eligible deployment of the application reaching the needed Things at Fog 1 and Fog 3, and meeting the financial (at most €850 per month) and QoS-assurance (at least 98% of the time) constraints mentioned above?*

Q1(b) — *Which eligible deployment represent the most balanced trade-off optimising QoS-assurance, Fog resource consumption and monthly deployment cost of the smart building application?*

Q2 — *Does the upgrade from 3G to 4G at Fog 2 make it possible to determine a deployment with better trade-off on QoS-assurance, Fog resource consumption and monthly deployment cost?*

[17] Results and Python code to generate 3D plots as in Figures 5 and 6 are available at: `https://github.com/di-unipi-socc/FogTorchPI/tree/costmodel/results/SMARTBUILDING18/`.

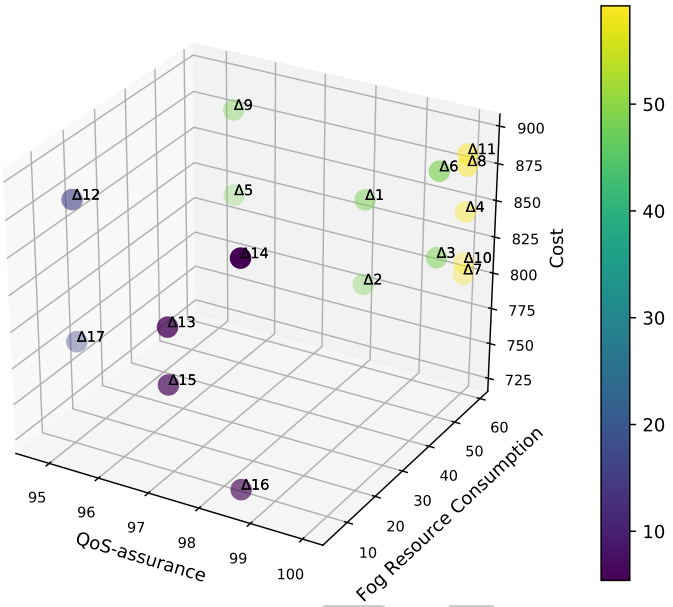

Fig. 4. FogTorchΠ output deployments and predicted metrics.

Figure 5 shows the feasible space for questions **Q1(a)** and **Q1(b)**, as defined by the QoS-assurance and budget constraints declared by the system integrators. Only Δ2, Δ3, Δ4, Δ7 and Δ10 meet those constraints. Analogously, Figure 6 shows the feasible space for **Q2**, obtained when upgrading Fog 2 to 4G, and also includes Δ16 as feasible.

The answer to question **Q1(a)** is positive. Indeed, FogTorchΠ outputs eleven deployments (Δ1 — Δ11 in Table 3) which are in the feasible solution space, determined as per the algorithms of Section 3.

It is worth recalling that we consider remote access to IoT devices connected to Fog nodes from other Cloud and Fog nodes. Indeed, some output deployments map components to nodes that do not directly connect to all the bound Things. As an example, in the case of Δ1, IoTController is deployed to Fog 2 but the required Things (fire_sensor_1, light_control_1, thermostate_1, video_camera_1, weather_station_3) connected to Fog 1 and Fog 3, which are still reachable with suitable network QoS (viz., latency and bandwidth).

To answer questions **Q1(b)** and **Q2** we employ the multi-criteria optimisation methodology described before. Table 4 shows the values of the single objectives, and of the aggregate Fog-ward (i.e., $r_F(\Delta)$) and Cloud-ward (i.e., $r_C(\Delta)$) objective functions, which the system integrators are trying to optimise, when deploying the smart building application.

In the Fog-ward case, when looking for the best trade-off among QoS-assurance, resource consumption and cost, the most promising deployment is always Δ7, i.e., the one with the highest value for $r_F(\Delta)$. In the Cloud-ward case, when the upgrade to 4G is

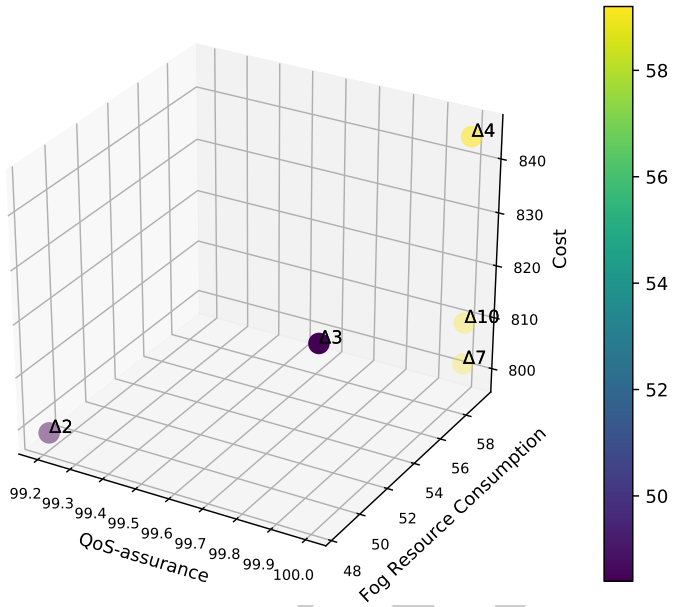

Fig. 5. Feasible space for **Q1(a)–(b)** as in [9]. Colormap refers to Fog resource consumption.

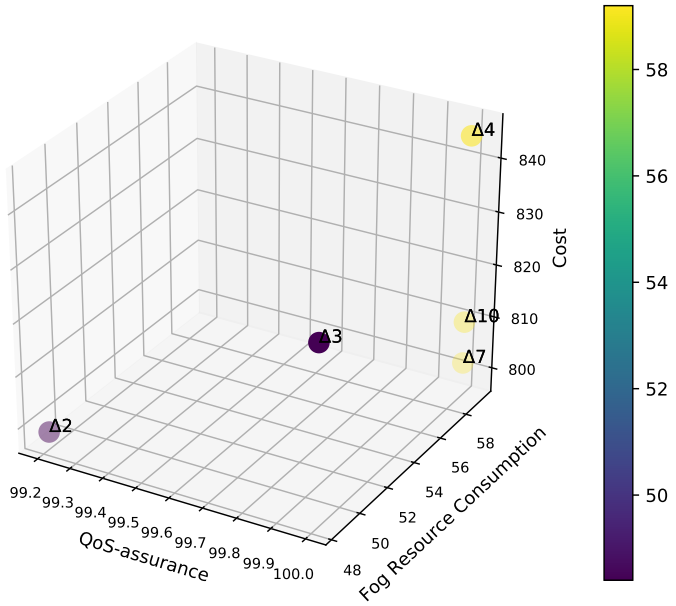

Fig. 6. Feasible space for **Q2** as in [9]. Colormap refers to Fog resource consumption.

Table 4. Ranking of eligible deployments.

Dep. ID	IoTController	DataStorage	Dashboard	QoS	Cost	Resources	$r_F(\Delta)$	$r_C(\Delta)$
$\Delta 2$	Fog 2	Fog 3	Cloud 1	98.6%	€798.7	48.4%	0.42	0.22
$\Delta 3$	Fog 3	Fog 3	Cloud 1	100%	€829.7	48.4%	0.65	0.45
$\Delta 4$	Fog 2	Fog 3	Fog 1	100%	€844.7	59.2%	0.67	0.33
$\Delta 7$	Fog 3	Fog 3	Fog 2	100%	€801.7	59.2%	0.81	0.48
$\Delta 10$	Fog 1	Fog 3	Fog 2	100%	€809.7	59.2%	0.79	0.45
$\Delta 16^*$	Fog 2	Cloud 1	Cloud 1	98.6%	€727.7 (+20)	5.4%	0.33	0.67

not considered, $\Delta 7$ still represents the best candidate deployment. These considerations altogether answer **Q1(b)**.

The 4G upgrade at Fog 2, which makes it possible to enact also $\Delta 16$, is not worth the investment when looking for Fog-ward deployments due to the much lower score achieved in the ranking with respect to $\Delta 7$. Conversely, when upgrading to 4G in the Cloud-ward case, despite investing €20 more every month, $\Delta 16$ is the best option (actually leading to €50 of monthly savings with respect to $\Delta 7$). These considerations altogether answer **Q2**.

In [9], $\Delta 2$ and $\Delta 16$ were chosen by the system integrators, in the 3G and 4G scenario respectively. However, their goal in [9] was to minimise deployment costs and Fog resource consumption (going Cloud-ward), only guaranteeing QoS-assurance above 98% (without trying to maximise it). Here, we are instead looking for the most balanced trade-off among the predicted metrics (which are all weighted equally[18]) and $\Delta 7$ clearly constitutes a better compromise towards this end, guaranteeing 100% QoS-assurance and costing only a few euro more than $\Delta 2$, despite using 10% more Fog resources.

7 RELATED WORK

Few approaches have been proposed so far to specifically model Fog infrastructures and applications, as well as to determine and compare eligible deployments for an application to a Fog infrastructure under different metrics. On the other hand, the problem of deciding how to deploy multi-component applications has been extensively studied in Cloud scenarios. Projects like SeaClouds [12], Aeolus [20] or Cloud-4SOA [14], for instance, proposed model-driven optimised planning solutions to deploy software applications across different (IaaS or PaaS) Clouds. [33] proposed to use OASIS TOSCA [13] to model IoT applications in Cloud+IoT scenarios. Also, solutions to automatically provision and configure software components in Cloud (or multi-Cloud) scenarios are currently used by the DevOps community to automate application deployment or to lead deployment design choices (e.g., Puppet [2] and Chef [1]). However, only few efforts in Cloud computing considered non-functional requirements *by-design* [38, 15] or uncertainty of execution (as in Fog nodes) and security risks among interactive and interdependent components [50, 29].

[18] By tuning ω_i differently and by considering the Cloud-ward case, we can obtain the same results of [9], e.g. assigning weight 0.50 to both resource consumption and cost, and 0.0 to QoS-assurance $\Delta 2$ is ranked 0.34 whilst $\Delta 7$ scores 0.22.

Fog introduces new problems with respect to the Cloud paradigm, mainly due to its pervasive geo-distribution and heterogeneity, need for QoS-awareness, dynamicity and support to interactions with the IoT, that were not taken into account by previous works (as reported in [46, 49, 3]).

[44] was among the first attempts to evaluate service latency and energy consumption of the new Fog paradigm applied to the IoT, as compared to traditional Cloud scenarios. The model of [44], however, deals only with the behaviour of software already deployed over Fog infrastructures and simulates it mathematically.

Also investigating this new lines, [28] proposed a Fog-to-Cloud search algorithm as a first way to determine an eligible deployment of (multi-component) DAG applications to tree-like Fog infrastructures. Their placement algorithm proceeds Edge-ward, i.e. it attempts the placement of components *Fog-to-Cloud* by considering hardware capacity only. An open-source simulator – iFogSim – has been released to test the proposed policy against Cloud-only deployments. Building on top of iFogSim, [34] refines the Edge-ward algorithm to guarantee the application service delivery deadlines and to optimize Fog resource exploitation. Limiting their work to linear application graphs and tree-like infrastructure topologies, [48] used iFogSim to implement an algorithm for optimal online placement of application components, with respect to load balancing. An approximate extension handling tree-like application was also proposed. Recently, exploiting iFogSim, [26] proposed a distributed search strategy to find the best service placement in the Fog, which minimises the distance between the clients and the most requested services, based on request rates and available free resources. Their results showed a substantial improvement on network usage and service latency for the most frequently called services. [30] proposed a (linearithmic) heuristic algorithm that attempts deployments prioritising placement of smaller applications to devices with less free resources. Along the same line, [45] proposed an Edge-ward linearithmic algorithm that assigns application components to the node with the lowest capacity that can satisfy all application requirements.

Cost is an important parameter in choosing an eligible deployment, yet pricing models for the Fog are still to be developed. In the case of Cloud scenarios, pricing models are well established (e.g., [21], [41] and references therein) yet they do not consider costs generated by the usage of IoT devices. Cloud pricing models are generally divided into two types, pay per use scheme and subscription-based. For both types, the total cost of deployment is calculated based on user requirements (e.g., the number of CPU cores, VM types, time duration, type of instance (reserved or pre-emptible). Since multiple offers among cloud providers can satisfy user needs, a cloud broker can be needed to choose a best VM instance(s) based upon the pricing model [21]. In the case of IoT, the providers normally process data coming from the IoT devices and sell the processed information as value added service to the users. IoT providers can sometime federate their services and create new offers for the end-users [41]. Depending upon data demand end-users can estimate the total cost of using IoT services by comparing pay-per-use and subscription-based offers. A cost model that considers various parameters (e.g., the type and number of sensors, number of data request and uptime of VM) to estimate the cost of running an application for IoT+Cloud scenario over a certain period

of time was proposed in [36]. In Fog scenario, however, there is a need to compute IoT costs at a finer level, also accounting for data-transfer costs (i.e., event-based).

Trade-off analysis is needed to allow application deployers to identify the best candidate deployment and prioritise some metrics over the rest. The principles of Pareto optimality are widely used in this regard[18]. Such optimization has been applied in the cloud to optimize resource utilization and Virtual machine placement ([52, 25]). It is also employed in Fog scenario to address other challenges [31, 4, 40]. For instance, [31] used multi-objective optimization to extend iFogSim [28] to support the automated gateways selection and fog devices clustering. [4, 40] applied the optimization principle for virtual machine (VM) placement and resource utilization on fog nodes to satisfy the application requirement. To summarise, most of the surveyed approaches focussed on one among Cloud, Fog or IoT, or mainly considered linear cost models based on unit cost for different types of hardware resource. Our attempt is, to the best of our knowledge, the first to model costs in the Fog scenario that extends Cloud pricing schemes to the Fog layer, integrates them with costs of IoT deployments and at the same time enable the trade-off analysis between different metrics.

Inspired by our work on FogTorchΠ methodologies [7, 8, 10], Xia et al. [51] proposed a backtracking solution to minimise the average response time of deployed IoT applications. Two new heuristics were devised. The first one sorts the nodes considered for deploying each component in ascending order with respect to the (average) latency between each node and the IoT devices required by the component. The second one considers a component that caused backtracking as the first one to be mapped in the next search step. Despite discussing improved results on latency with respect to exhaustive backtracking and first-fit strategies, no prototype implementations were released. Finally, FogTorchΠ was also modularly extended by De Maio et al. to simulate mobile task offloading in Edge computing scenarios [17].

8 CONCLUDING REMARKS

In this paper, we extended the work presented in [9] by illustrating how the proposed cost model for Fog application deployments can be used in a multi-objective optimisation framework to determine deployments that achieve a *best* trade-off among predicted QoS-assurance, Fog resource consumption and monthly deployment cost.

Indeed, finding a good compromise among those (often orthogonal) objectives is not a trivial task to be accomplished without actual support or, worst, by trial-and-error. By means of a lifelike motivating example, we have shown how our prototype, FogTorchΠ, can help (multi-component) application deployers to determine such optimal trade-off according to their preferences. The deployment resulting from the multi-objective optimisation are clearly QoS- (i.e., accounting for variations in the QoS of communication links), context- (i.e., exploiting the contextually available resources), and cost-aware (i.e., considering Cloud, Fog and IoT related costs).

We envision the possibility of exploiting such optimisation methodology to evaluate, at design time, the feasibility of different application deployments, whilst taming the complexity, scale and intrinsic heterogeneity of Fog infrastructures. Furthemore, new businesses coming to market can use tools like FogTorchΠ not only to decide on

where to deploy their application components but also to design their SLAs and billing schemes for the services they will offer.
We see three main directions for future work:

- since security represents a concern that should be addressed *by-design* at all architectural levels of Fog computing [42], we aim at devising a novel methodology to perform quantitative assessments of the security level of Fog application deployments and to show how it can synergically work with FogTorchΠ,
- by approximating the estimate of the proposed objectives (in particular of the QoS-assurance) and by envisioning the possibility for application components to be deployed in different flavours (like in *Osmotic Computing*[47]), we aim at exploiting other optimisation frameworks such as bio-inspired or swarm intelligence techniques, and
- finally, to assess our predictive analyses in support of Fog application deployment we aim at using them in available simulation environments for the management of Fog applications (e.g., [28, 32, 24] and, if possible, in experimental testbed settings (e.g., using as a starting point the Fog application of [11]).

References

1. Opscode. Chef. http://www.opscode.com/chef/
2. Puppetlabs. Puppet. http://puppetlabs.com
3. Arcangeli, J.P., Boujbel, R., Leriche, S.: Automatic deployment of distributed software systems: Definitions and state of the art. Journal of Systems and Software **103**, 198–218 (2015)
4. Aryal, R.G., Altmann, J.: Dynamic application deployment in federations of clouds and edge resources using a multiobjective optimization AI algorithm. In: Third International Conference on Fog and Mobile Edge Computing, FMEC 2018, Barcelona, Spain, April 23-26. pp. 147–154 (2018). https://doi.org/10.1109/FMEC.2018.8364057
5. Bonomi, F., Milito, R., Natarajan, P., Zhu, J.: Fog computing: A Platform for Internet of Things and Analytics. In: Big Data and Internet of Things: A Roadmap for Smart Environments, pp. 169–186 (2014)
6. Breitbart, Y., Chan, C.Y., Garofalakis, M., Rastogi, R., Silberschatz, A.: Efficiently monitoring bandwidth and latency in IP networks. In: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. vol. 2, pp. 933–942. IEEE (2001)
7. Brogi, A., Forti, S.: QoS-aware Deployment of IoT Applications Through the Fog. IEEE Internet of Things Journal **4**(5), 1185–1192 (Oct 2017). https://doi.org/10.1109/JIOT.2017.2701408
8. Brogi, A., Forti, S., Ibrahim, A.: How to best deploy your Fog applications, probably. In: Rana, O., Buyya, R., Anjum, A. (eds.) Proceedings of 1st IEEE International Conference on Fog and Edge Computing (ICFEC), Madrid. pp. 105–114 (May 2017). https://doi.org/10.1109/ICFEC.2017.8
9. Brogi, A., Forti, S., Ibrahim, A.: Deploying Fog Applications: How Much Does It Cost, By the Way? In: Proceedings of the 8th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,. pp. 68–77. INSTICC, SciTePress (2018). https://doi.org/10.5220/0006676100680077
10. Brogi, A., Forti, S., Ibrahim, A.: Predictive Analysis to Support Fog Application Deployment. In: Buyya, R., Srirama, S.N. (eds.) Fog and Edge Computing: Principles and Paradigms. Wiley (2018), *In press*

11. Brogi, A., Forti, S., Ibrahim, A., Rinaldi, L.: Bonsai in the fog: An active learning lab with fog computing. In: Fog and Mobile Edge Computing (FMEC), 2018 Third International Conference on. pp. 79–86. IEEE (2018)
12. Brogi, A., Ibrahim, A., Soldani, J., Carrasco, J., Cubo, J., Pimentel, E., D'Andria, F.: SeaClouds: a European project on seamless management of multi-cloud applications. ACM SIGSOFT SEN **39**(1), 1–4 (2014)
13. Brogi, A., Soldani, J., Wang, P.: TOSCA in a Nutshell: Promises and Perspectives, pp. 171–186. Proceedings of ESOCC 2014. (2014)
14. Corradi, A., Foschini, L., Pernafini, A., Bosi, F., Laudizio, V., Seralessandri, M.: Cloud PaaS Brokering in Action: The Cloud4SOA Management Infrastructure. In: VTC 2015. pp. 1–7 (2015)
15. Cucinotta, T., Anastasi, G.F.: A heuristic for optimum allocation of real-time service workflows. In: Service-Oriented Computing and Applications (SOCA), 2011 IEEE Int. Conf. on. pp. 1–4. IEEE (2011)
16. Dastjerdi, A.V., Buyya, R.: Fog Computing: Helping the Internet of Things Realize its Potential. Computer **49**(8), 112–116 (2016)
17. De Maio, V., Brandic, I.: First Hop Mobile Offloading of DAG Computations (2018), *In press*
18. Deb, K.: Multi-Objective Optimization, pp. 273–316. Springer US, Boston, MA (2005)
19. Deb, K.: Multi-objective optimization. In: Search methodologies, pp. 403–449. Springer (2014)
20. Di Cosmo, R., Eiche, A., Mauro, J., Zavattaro, G., Zacchiroli, S., Zwolakowski, J.: Automatic deployment of software components in the cloud with the aeolus blender. In: ICSOC 2015, pp. 397–411 (2015)
21. Díaz, J.L., Entrialgo, J., García, M., García, J., García, D.F.: Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing. Future Generation Computer Systems **71**, 129–144 (2017)
22. Dunn, W.L., Shultis, J.K.: Exploring Monte Carlo Methods. Elsevier (2011)
23. Fatema, K., Emeakaroha, V.C., Healy, P.D., Morrison, J.P., Lynn, T.: A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives. Journal of Parallel and Distributed Computing **74**(10), 2918–2933 (2014)
24. Forti, S., Ibrahim, A., Brogi, A.: Mimicking FogDirector Application Management. Computer Science - Research and Development (2018), *In press*
25. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences **79**(8), 1230 – 1242 (2013)
26. Guerrero, C., Lera, I., Juiz, C.: A lightweight decentralized service placement policy for performance optimization in fog computing. Journal of Ambient Intelligence and Humanized Computing (Jun 2018). https://doi.org/10.1007/s12652-018-0914-0
27. Guerrero, C., Lera, I., Juiz, C.: Resource optimization of container orchestration: a case study in multi-cloud microservices-based applications. The Journal of Supercomputing **74**(7), 2956–2983 (Jul 2018)
28. Gupta, H., Dastjerdi, A.V., Ghosh, S.K., Buyya, R.: iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments. Software: Practice and Experience **47**(9), 1275–1296 (2017). https://doi.org/10.1002/spe.2509
29. Haithem, M., Mokhtar, S., Jaber, K.: Security-aware SaaS placement using swarm intelligence. Journal of Software: Evolution and Process **0**(0), e1932. https://doi.org/10.1002/smr.1932, https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1932

30. Hong, H.J., Tsai, P.H., Hsu, C.H.: Dynamic module deployment in a fog computing platform. In: 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS). pp. 1–6 (Oct 2016). https://doi.org/10.1109/APNOMS.2016.7737202
31. Kimovski, D., Ijaz, H., Surabh, N., Prodan, R.: An Adaptive Nature-inspired Fog Architecture. CoRR **abs/1803.03444** (2018)
32. Lera, I., Guerrero, C.: YAFS - Yet Another Fog Simulator (for python). https://yafs.readthedocs.io/en/latest/
33. Li, F., Vögler, M., Claeßens, M., Dustdar, S.: Towards automated IoT application deployment by a cloud-based approach. In: SOCA 2013. pp. 61–68 (2013)
34. Mahmud, R., Ramamohanarao, K., Buyya, R.: Latency-aware application module management for fog computing environments. ACM Transactions on Internet Technology (TOIT) (2018)
35. Markus, A., Kertesz, A., Kecskemeti, G.: Cost-aware iot extension of dissect-cf. Future Internet **9**(3), 47 (2017)
36. Markus, A., Kertesz, A., Kecskemeti, G.: Cost-Aware Iot Extension of DISSECT-CF. Future Internet **9**(3) (2017). https://doi.org/10.3390/fi9030047, http://www.mdpi.com/1999-5903/9/3/47
37. Marler, R.T., Arora, J.S.: The weighted sum method for multi-objective optimization: new insights. Structural and multidisciplinary optimization **41**(6), 853–862 (2010)
38. Nathuji, R., Kansal, A., Ghaffarkhah, A.: Q-Clouds: Managing Performance Interference Effects for QoS-Aware Clouds. Association for Computing Machinery, Inc. (April 2010)
39. Newman, S.: Building microservices: designing fine-grained systems. " O'Reilly Media, Inc." (2015)
40. Nguyen, D.T., Le, L.B., Bhargava, V.: Price-based resource allocation for edge computing: A market equilibrium approach. CoRR **abs/1805.02982** (2018)
41. Niyato, D., Hoang, D.T., Luong, N.C., Wang, P., Kim, D.I., Han, Z.: Smart data pricing models for the internet of things: a bundling strategy approach. IEEE Network **30**(2), 18–25 (2016)
42. OpenFog: OpenFog Reference Architecture (2016)
43. Perera, C.: Sensing as a Service for Internet of Things: A Roadmap. Lulu. com (2017)
44. Sarkar, S., Misra, S.: Theoretical modelling of fog computing: a green computing paradigm to support IoT applications. IET Networks **5**(2), 23–29 (2016)
45. Taneja, M., Davy, A.: Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). pp. 1222–1228 (May 2017). https://doi.org/10.23919/INM.2017.7987464
46. Varshney, P., Simmhan, Y.: Demystifying Fog Computing: Characterizing Architectures, Applications and Abstractions. In: 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC). pp. 115–124 (May 2017). https://doi.org/10.1109/ICFEC.2017.20
47. Villari, M., Fazio, M., Dustdar, S., Rana, O., Ranjan, R.: Osmotic computing: A new paradigm for edge/cloud integration. IEEE Cloud Computing **3**(6), 76–83 (2016)
48. Wang, S., Zafer, M., Leung, K.K.: Online Placement of Multi-Component Applications in Edge Computing Environments. IEEE Access **5**, 2514–2533 (2017). https://doi.org/10.1109/ACCESS.2017.2665971
49. Wen, Z., Yang, R., Garraghan, P., Lin, T., Xu, J., Rovatsos, M.: Fog Orchestration for Internet of Things Services. IEEE Internet Computing **21**(2), 16–24 (Mar 2017). https://doi.org/10.1109/MIC.2017.36
50. Wen, Z., Cała, J., Watson, P., Romanovsky, A.: Cost effective, reliable and secure workflow deployment over federated clouds. IEEE Transactions on Services Computing **10**(6), 929–941 (2017)

51. Xia, Y., Etchevers, X., Letondeur, L., Coupaye, T., Desprez, F.: Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed IoT applications in the fog. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. pp. 751–760. ACM (2018)
52. Xu, J., Fortes, J.A.: Multi-objective virtual machine placement in virtualized data center environments. In: IEEE/ACM GreenCom and CPSCom. pp. 179–188. IEEE (2010)