Received ; | Revised ; | Accepted

DOI: xxx/xxxx

SPECIAL ISSUE

How to Place Your Apps in the Fog
State of the Art and Open Challenges

Antonio Brogi[1] | Stefano Forti[1] | Carlos Guerrero[2] | Isaac Lera[2]

[1]Department of Computer Science, University of Pisa, Pisa, Italy
[2]Department of Computer Science, University of the Balearic Islands, Balearic Islands, Spain

Correspondence
Stefano Forti, Email: stefano.forti@di.unipi.it

Funding Information
This research was supported by the University of Pisa project "*DECLWARE: Declarative methodologies of application design and deployment*" (PRA_2018_66) and by the Spanish Government (Agencia Estatal de Investigación) and the European Commission (Fondo Europeo de Desarrollo Regional), Grant Number: TIN2017-88547-P (MINECO/AEI/FEDER, UE)

Summary

Fog computing aims at extending the Cloud towards the IoT so to achieve improved QoS and to empower latency-sensitive and bandwidth-hungry applications. The Fog calls for novel models and algorithms to distribute multi-component applications in such a way that data processing occurs wherever it is best-placed, based on both functional and non-functional requirements.

This survey reviews the existing methodologies to solve the application placement problem in the Fog, while pursuing three main objectives. First, it offers a comprehensive overview on the currently employed algorithms, on the availability of open-source prototypes, and on the size of test use cases. Second, it classifies the literature based on the application and Fog infrastructure characteristics that are captured by available models, with a focus on the considered constraints and the optimised metrics. Finally, it identifies some open challenges in application placement in the Fog.

KEYWORDS:
fog computing, application placement, service placement, application deployment, optimisation algorithms

1 | INTRODUCTION

CISCO expects more 50 billion of connected entities (people, machines and connected Things) by 2021, and estimates they will have generated around 850 Zettabytes of information by that time, of which only 10% will be useful to some purpose[1,2]. As a consequence of this trend, enormous amounts of data – the so-called *Big Data*[3] – are collected by IoT sensors and stored in Cloud data centres[4]. Once there, data are subsequently analysed to determine reactions to events or to extract analytics or statistics. Whilst data-processing speeds have increased rapidly, bandwidth to carry data to and from data centres has not increased equally fast[5]. On one hand, supporting the transfer of data from/to billions of IoT devices is becoming hard to accomplish due to the volume and geo-distribution of those devices. On the other hand, the need to reduce latency for time-sensitive applications, to eliminate mandatory connectivity requirements, and to support computation or storage closer to where data is generated 24/7, is evident[6].

In this context, a new utility computing paradigm took off, aiming at connecting the ground (IoT) to the sky (Cloud), and it has been named *Fog computing*[7]. The Fog aims at better supporting time-sensitive and bandwidth hungry IoT applications by selectively pushing computation closer to where data is produced and by exploiting a geographically distributed multitude of heterogeneous devices (e.g., personal devices, gateways, micro-data centres, embedded servers) spanning the continuum from

[0]**Abbreviations:** FAPP - Fog Application Placement Problem.

the IoT to the Cloud. In its reference architecture[1], the OpenFog Consortium (OFC)[9], which is fostering academic and industrial research in the field since 2015, gives the following definition of Fog computing:

> *Fog computing is a system-level horizontal architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from Cloud to Things, thereby accelerating the velocity of decision making. Fog-centric architecture serves a specific subset of business problems that cannot be successfully implemented using only traditional cloud-based architectures or solely intelligent edge devices.*

The NIST has also recently proposed a conceptual architecture for Fog computing[10]. Fog configures as a powerful enabling complement to the IoT+Edge and to the IoT+Cloud scenarios, featuring a new intermediate layer of cooperating devices that can autonomously run services and complete specific business missions, contiguously with the Cloud and with cyber-physical systems at the edge of the network[11].

Overall, Fog computing should ensure that computation over the collected data happens wherever it is *best-placed*, based on various application (e.g., hardware, QoS) or stakeholders (e.g., cost, business policies) requirements. Since modern software systems are more and more often made from distributed, (numerous) interacting components (e.g., service-oriented and micro-service based architectures), it is challenging to determine where to deploy each of them so to fulfil all set requirements. Lately, following this line, a significant amount of research has considered the problem of *optimally* placing application functionalities (i.e., computation) based on different – and sometimes orthogonal – constraints. However, to the best of our knowledge, no comprehensive and systematic survey of these efforts exists in the literature.

This work aims precisely at offering an exhaustive overview of the solutions proposed for the application placement problem in the Fog, by providing the reader with three complementary perspectives on this topic. Namely:

P1. an *algorithmic* perspective that reviews state-of-the-art contributions based on the methodologies that they employed to address Fog application placement, along with a study on the available prototypes and experiments,

P2. a *modelling* perspective that analyses which (functional and non-functional) constraints and which optimisation metrics have been considered in the literature to determine best candidate application placements,

P3. a *future work* perspective that, based on both P1 and P2, identifies and discusses some of the open research challenges that should be studied on this topic.

The rest of this paper is organised as follows. After describing the methodology followed to realise this survey (Section 2), a comprehensive analysis of state-of-the-art related to Fog application placement under P1 (Section 3.1) and P2 (Section 3.2) is presented. Finally, as per P3, some open problems and future research challenges that have been poorly addressed by the existing literature are pointed out (Section 4).

2 | SETTING THE STAGE

This survey includes research articles that deal with Fog application placement with the objective of optimising non-functional requirements of the system. To set the stage, we start by formally defining the considered application placement problem.

Definition – Let A be a multi-component application with a set of requirements R and let I be a distributed (Fog) infrastructure. Solutions to the *Fog Application Placement Problem* (FAPP) are mappings from each component of A to some computational node in I, meeting all requirements set by R and optimising a set of objective metrics O used to evaluate their quality. Solution mappings can be many-to-many, i.e. a component can be placed onto one or more nodes and a node can host more than one component.

It is worth noting that FAPP can be solved or adjusted also at runtime (i.e., when A is running), in case that some requirements in R cannot be met by the current solution mapping or whenever O can be further optimised as Fog infrastructure conditions change over time. In what follows, focussing on existing approaches to solve FAPP, we will exclude those works that only deal

[1] Very recently, the OFC reference architecture of Fog computing was adopted as the IEEE 1934-2018 standard[8].

with dispatching or scheduling of (user or IoT) requests, as they represent a phase that is subsequent to the placement of the application components.

The concept of FAPP and the underlying technology are very recent, and the boundary with other technologies is not always clear. In this survey, we include all the articles that deal with the placement of applications that are generally available for the users and that have been traditionally requested to Cloud providers. Similarly, we will exclude research in the field of task offloading (i.e., outsourcing of the computation of a given function to get a result back) from one node to a better (e.g., more powerful or reliable, closer) one, which is covered in detail by other recent surveys[12,13]. Finally, research in the field of mobile offloading, shifting the processing and execution from mobile terminals to the network or edges nodes, will be also excluded.

To the best of our knowledge, this is the first survey that covers the state of the art for FAPP. Our search criteria was formed by the following terms: $(\mathsf{Fog\ computing} \wedge (\mathsf{service} \vee \mathsf{application}) \wedge (\mathsf{placement} \vee \mathsf{deployment}))$. The search was carried out, with the help of Google Scholar, in the following libraries: IEEE Xplore Digital Library, Wiley Online Library, ACM Digital Library and Web of Science. To fully capture the advances in the field, both journal and conference articles were collected during the search phase. Additionally, the references in selected articles were also analysed to find more related work in the FAPP domain. At the end of this first step, the 110 articles we collected were carefully screened. After a more accurate and deeper selection process, where references in the field of offloading, dispatching and scheduling were removed, we selected 38 articles to be analysed further.

Table 1 offers a complete overview of the set of the analysed papers with respect to all the factors that were included in our survey of the state of the art, showing the classification with respect to the algorithms and models employed in different works that approached FAPP. Focussing the definition of FAPP from an optimisation point of view, the common elements in any optimisation process are the following:

Decision variables They are the items whose values need to be determined during the optimisation process. In the case of the FAPP, they can be the binary decision variables (or, equivalently, the mapping functions) that indicate if a component of A is allocated (or not) to a Cloud or Fog node of I. All the articles that we have included in our study rely on similar decision variables.

Objective function The objective function measures the suitability of a solution (a specific value assigned to the decision variables) for the optimisation process. The optimisation can be addressed to maximise or minimise the value of the objective function. In a more general way, the objective function represents the concerns of the optimisation and defines the metrics that are optimised by fixing the values of the decision variables. In our case, the objective functions measure the metrics O of candidate solutions to FAPP.

Constraints over the decision variables They define the requirements R that must be satisfied by each specific case of the decision variables. Solutions (values of the decision variables) that do not satisfy the constraints are rejected as possible solutions to FAPP.

Domain variables or parameters They commonly are the fixed values which are known previously to the optimisation process. But we also include here other variable, metrics or items related to the optimisation but that are not constraints neither optimisation objectives.

Algorithms They are the algorithms proposed to find the values of the decision variables that achieve the best objective value while the constraints are satisfied. The problem of finding solution mappings between application components in A and Fog/Cloud nodes in I, whilst minimising/maximising the values of the objectives functions O and satisfying the constraints in R, is NP-hard. Moreover, when several opposite optimisation objectives are considered, it is necessary to determine a trade-off between the objectives because some of them may increase when the other decrease. Indeed, FAPP needs to be solved by evaluating – at worst – all the possible solutions, i.e., assuming the application is made of m modules and the infrastructure is composed of n nodes, n^m different candidate solutions. Such complexity can be tamed by using heuristics which permit to find sub-optimal solutions. It is important to consider that Fog computing is a large-scale and dynamic domain, where real implementations have to deal with huge quantities of Fog nodes and applications.

Overall, we have organised the description and presentation of the articles from the point of view of the elements of the optimisation process: algorithms (Section 3.1), constraints and parameters (Section 3.2.1), and objective functions, presented

TABLE 1 Articles according with the considered constraints and optimised metrics

		Considered Constraints										Optimised & Comparison Metric(s)								
		Network				Nodes		Energy	Application			Network		Nodes	Energy	Performance			Cost	
	Ref.	Latency	Bandwidth	Link Reliability	Topology	Hardware	Software	Energy	Workload	Dependencies	User preferences	Delay	Bandwidth	Hardware	Energy	QoS-assurance	Execution time	Migrations	Cost	Open prototype
Search	14 15	✓	✓		✓	✓						✓	✓		✓		✓			✓
	16	✓				✓				✓						✓				✓
	17	✓			✓	✓			✓	✓		✓								
	18	✓	✓		✓				✓	✓		✓	✓					✓		
	19	✓	✓			✓	✓						✓			✓				
	20	✓	✓			✓	✓				✓		✓			✓				✓
	21 22	✓	✓			✓	✓						✓			✓			✓	
	23					✓	✓					✓					✓			
	24		✓	✓		✓				✓									✓	
	25	✓	✓			✓										✓			✓	
	26		✓			✓							✓	✓	✓		✓			
Mathematical Programming	27											✓						✓		
	28		✓			✓			✓	✓									✓	
	29				✓	✓			✓	✓		✓		✓				✓	✓	
	30					✓						✓								
	31					✓				✓		✓					✓			
	32	✓				✓						✓								
	33	✓	✓	✓	✓	✓		✓							✓					
	34					✓			✓							✓				✓
	35				✓	✓					✓			✓		✓				
	36				✓	✓				✓				✓		✓				
	37				✓							✓					✓		✓	
	38				✓										✓					
	39	✓										✓			✓		✓			
	40		✓		✓	✓										✓				
	41 42					✓										✓				✓
	43	✓	✓	✓	✓	✓					✓	✓					✓			✓
Other Algorithms	44																✓			
	45		✓	✓		✓						✓					✓			
	46	✓																	✓	
	47	✓												✓						
	48	✓	✓		✓	✓									✓	✓	✓		✓	
	49	✓				✓		✓						✓	✓	✓		✓	✓	
	50					✓				✓			✓		✓	✓			✓	
	51				✓	✓						✓						✓		

(Section 3.2.2). Table 1 presents a summary of the characteristics of the articles by following this organisation into algorithms, constraints and optimisation metrics.

The result of the analysis of the algorithms proposed in the articles has shown that seven different types of algorithm have been proposed to optimise FAPP. We have grouped and analysed the articles in Section 3.2.2 by those optimisation algorithms resulting in two main groups, with the highest number of articles proposing mathematic programming and heuristic/search solutions. The rest of algorithms have been only studied in a more reduced number of articles: bio-inspired algorithms, game theory, deep learning, dynamic programming, and complex networks.

In the case of the constraints, we analysed the articles and classified the constraints into a two-level taxonomy. In a first level, we considered constraints related to the main elements in a Fog architecture: application, nodes (or Fog devices), network, and energy. With the analysis of the articles, we detected: network constraints related to the communication time (latency), bandwidth, link reliability, and the organisation and interrelation between the network components (topology), node constraints related to their hardware characteristics and the features of their operating systems or available frameworks (software), application constraints related to the type and number of requests that are generated in the systems (workload), the organisation and interrelation between application modules (dependencies), and the possibility of the users to establish conditions in the application placement or to choose between a set of placement alternatives (user preferences). The energy constraints are only related to the power consumption and a second classification was not considered for them.

From the analysis of the optimisation objectives, we observed that the articles could be described by considering if they optimised or analysed metrics with respect to: the network, in particular, the delay due to the communication that is included in the execution time of the applications (network delay), and the usage of the network (network bandwidth), the nodes or devices, more specifically, the quantity of hardware resources that are consumed by the applications in the nodes (hardware), the energy, i.e., the power consumption that the execution of the applications generates in the Fog nodes, the performance of the system, and more concretely, the Quality of Service (QoS), commonly measured as the number of requests executed before a specific deadline (QoS-assurance), the execution time of the application requests (execution time), and the overhead that generates in the system the migration of application modules through the Fog nodes (migrations), and finally, the price to be paid for the use of the Fog resources (cost).

3 | ANALYSIS OF THE STATE OF THE ART

3.1 | Algorithms

In this section, all reviewed approaches are analysed according to the algorithms or methodologies that they use for solving FAPP. We identified three main classes of algorithms which have been exploited in the literature. Namely:

- *search-based* algorithms, such as (heuristic) backtracking search, or first- and best-fit application placement,
- *mathematical programming* algorithms, such as integer or mixed integer linear programming,
- *other* algorithms like game theoretical or deep learning.

The rest of this section is organised according to this classification with the goal of providing the reader with a complete overview of the state of the art related to frameworks used for solving FAPP (Sections 3.1.1 to 3.1.3). For each reviewed approach, we indicate the availability of open source research prototypes and the size of the experiment used for assessing or testing it.

3.1.1 | Search-based Algorithms

Being FAPP an NP-hard problem, the first solutions which have been exploited to solve it are traditional search algorithms along with greedy or heuristic behaviour[52].

Among the first proposals investigating this direction, Gupta et al.[14] and Mahmud et al.[15] proposed a Fog-to-Cloud search algorithm as a first way to determine an eligible (composite) application placement of a given (Directed-Acyclic) IoT application to a (tree-structured) Fog infrastructure. The search algorithm proceeds Edge-ward, i.e. it attempts the placement of components *Fog-to-Cloud* by considering hardware capacity only, and allows merging homologous components from different application deployments. An open-source tool – iFogSim – implementing such strategy has been released and used to compare it with *Cloud-only* application placement over two quite large use cases from VR gaming (3 application components scaled up to 66, 1 Cloud, up to 80 Fog nodes) and video-surveillance (4 application components, scaled up to 19, 1 Cloud, up to 17 Fog nodes). Building on top of iFogSim, Mahmud et al.[16] refined the Edge-ward algorithm to guarantee the application service delivery deadlines and to optimise Fog resource exploitation.

Recently, exploiting iFogSim, Guerrero et al.[17] 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. Each Fog node takes local decision to optimise the application placement, based on request rates and free available resources. The *SocksShop*[53] application (9 components) is used to evaluate the proposed decentralised solution against the Edge-ward policy of iFogSim –varying the replication factor in the range $1-5$ and the number of Fog nodes in the range $1-25$. The results showed a substantial improvement in network usage and service latency for the most frequently requested services. Also, Ottenwalder et al.[18] proposed a distributed solution – MigCEP – both to FAPP and to runtime migration. Such proposal reduces the network usage in Fog environments and ensures end-to-end latency constraints by searching for the best migration, based on probabilistic mobility of application users. The OMNeT++[54] simulator was used to show how MigCEP improves live migration against a static and a greedy strategy over a dynamic large-scale infrastructure (i.e., 1000 emulated connected cars). No codebase was released for either the works of Guerrero et al.[17] or Ottenwalder et al.[18].

In this context, Brogi et al.[19] proposed both an exhaustive and a greedy backtracking algorithm to solve FAPP based on the (hardware, software, IoT and QoS) requirements of multi-component applications, as well as to estimate QoS-assurance, resource consumption in the Fog layer[20] and monthly deployment cost[21] of eligible placements. The devised approach works on arbitrary application and infrastructure graph topologies. The greedy heuristic attempts the placement of components sorted in ascending order on the number compatible nodes (i.e., *fail-first*), considering candidate nodes one by one sorted in decreasing order on the available resources (i.e., *fail-last*). QoS-assurance is estimated by means of Monte Carlo simulations so to consider variations in the QoS of end-to-end communication links and to predict how likely each application placement is to comply with the desired network QoS[20]. An open-source prototype – FogTorchΠ – implementing the whole methodology has been released and used on different simple use cases (i.e., 3 application components, 2 Clouds, 3 Fog nodes) from smart agriculture and smart building scenarios. Despite exploiting worst-case exponential-time algorithms, the prototype was shown to scale[22] also on the larger VR game example proposed by Gupta et al.[14]. FogTorchΠ was also modularly extended by other researchers to simulate mobile task offloading in Edge computing[55].

Significantly inspired by Brogi and Forti[19], Xia et al.[23] proposed a backtracking solution to FAPP 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. A motivating IoT application[56] (i.e., 6 application components replicated up to $\simeq$ 800 times) was used to assess the algorithms on very large random infrastructures (i.e., 1 Cloud, up to $\simeq$ 20000 Fog nodes). The exhaustive search handled at most 150 nodes, whilst the first-fit and the heuristic strategy scaled up to 20000 nodes, the latter showing a 40% improvement on the response time with respect to first-fit. Prototype implementations were not released.

Limiting their work to linear application graphs and tree-like infrastructure topologies, Wang et al.[24] described an algorithm for optimal online placement of application components, with respect to load balancing. The algorithm searches for cycle-free solutions to FAPP, and shows quadratic complexity in the number of considered computational nodes. An approximate extension handling tree-like application is also proposed, which considers placing each (linear) branch of the application separately and shows increased time complexity. The approach is simulated on 100 applications to be placed featuring 3 to 10 components each and infrastructures with 2 to 50 nodes. Finally, the achieved performance is compared against the Vineyard algorithm[57] and a greedy search minimising resource usage.

Hong et al.[25] proposed a (linearithmic) heuristic algorithm that attempts deployments prioritising placement of smaller applications to devices with less free resources. A face detection application made from 3 components is used to evaluate the algorithm on a small real testbed (1 server acting as Cloud, 5 Fog nodes). Along the same line, Taneja and Davy[26] proposed a similar search algorithm that assigns application components to the node with the lowest capacity that can satisfy application requirements, also featuring linearithmic time complexity in the number of considered nodes. Binary search on the candidate nodes is exploited as a heuristic to find the best placement for each component, by attempting deployment to Fog nodes first (i.e., Fog-to-Cloud). The medium-scale experiments (i.e., 6 application components, 1 Cloud, up to 13 Fog nodes) were carried in iFogSim, but the code to run them has not been made publicly available.

3.1.2 | Mathematical Programming

Mathematical programming[58] is often exploited to solve optimisation problems by systematically exploring the domain of an objective function with the goal of maximising (or minimising) its value, i.e., identifying a best candidate solution. Many of the reviewed approaches tackled FAPP with such a mathematical framework, relying on Integer Linear Programming (ILP), Mixed-Integer Linear Programming (MILP) or Mixed-Integer Non-Linear Programming (MINLP).

Velasquez et at.[27] proposed a framework and an architecture for application placement in Fog computing. An ILP implementation is suggested but it is not realised nor evaluated by the authors. On the other hand, Arkian et al.[28], in addition to proposing a Fog architectural framework, formulated FAPP as a MINLP problem, where application components (i.e., VMs) are to be deployed to Fog nodes so to satisfy end-to-end delay constraints. The problem is then solved (along with the problem of task dispatching) by linearisation into a MILP and the solution is evaluated on data traces from IoT service demands (from 10-100 thousand devices) in the province of Teheran, Iran. The results of the experiment showed that the Fog promises to improve latency, energy consumption and costs for routing and storage. Similarly, Yang et al.[29] tackled both FAPP and its cost-aware extension with the problem of balancing request dispatching. The proposed methodology attempts optimising the trade-off between access latency, resource usage, and (data) migrations (costs). It accounts for constraints on the available resources as well as for workload variations depending on users' service accessing patterns. A novel greedy heuristic (solving a relaxed LP problem and approximating a solution for the full-fledged ones) is shown to outperform other benchmark algorithms (i.e., classic ILP and GA) both in terms of obtained results and algorithm execution time over an example with 20 Fog nodes and 30 services to be placed.

Alike to these proposals, Gu, Zeng et al.[30,31] solved FAPP along with task scheduling, converting a MINLP into a MILP, solved with the commercial tool Gurobi[59]. A simulation and comparison is provided against server-greedy (i.e., Cloud-ward) and client-greedy (i.e., Edge-ward) placement strategies, showing good improvements on response time when using the proposed approach in a medium (i.e., 15 to 25 application images, 11 Fog/Cloud servers[31]) and large (i.e., 45 to 80 Fog nodes[30]) sized use case. Still relying on Gurobi (together with PuLP[60]), Souza et al.[32] solved FAPP as an ILP problem, aimed at optimising latency/delay needed for resource allocation. Resources are modelled uniformly as available slots at different nodes. Out of the services to be deployed, few of them were considered to require large amounts of resources (i.e., *elephants*, 10%) and many more required instead few resources (i.e., *mice*, 90%). In the experimental settings (90 applications, 1 Cloud, 6 Fog nodes) both sequential and parallel resource allocation were considered and the benefit of Fog computing was shown in terms of reduced delays in service access.

Alternatively, Barcelo et al.[33] combined FAPP with the routing of requests across an IoT-Cloud (i.e., Fog) infrastructure. They modelled FAPP as a minimum (energy) cost mixed-cast flow problem, considering unicast downstream and multicast upstream, typical of IoT. Solution to FAPP is then provided by means of known poly-time algorithms, which are simulated via the LP solver Xpress-MP[61] on mock data traces from three use cases (i.e., smart city, smart building, smart mobility). The experiments simulated tens of devices and showed some performance improvements (as per latency, energy, reliability) with respect to traditional IoT deployments that do not rely on Fog nodes. Mahmud et al.[34] proposed instead a QoE extension of iFogSim – based on an ILP modelling of users expectation – which exploited fuzzy logic and achieved improvements in network conditions and service quality.

Skarlat et al. designed a hierarchical modelling of Fog infrastructures, consisting of a centralised management system to control Fog nodes organised per *colonies*[35,36,37]. Particularly, Skarlat et al.[35] adopted an ILP formulation of the problem of allocating computation to Fog nodes in order to optimise (user-defined) time deadlines on application execution, considering IoT devices needed to properly run the application. A simple linear model for the Cloud costs is also taken into account. The proposed approach is compared via simulation to first-fit and Cloud-only deployment strategies, showing good margins for improvement, on a medium-sized use case (i.e., up to 80 services, 1 Cloud, 11 Fog nodes).

Some of the methodologies proposed in the literature combine ILP with other optimisation techniques. Huang et al.[38], for instance, modelled the problem of mapping IoT services to Edge/Fog devices as a quadratic programming problem, afterwards simplified into an ILP and into a Maximum Weighted Independent Set (MWIS) problem. The services are described as a co-location graph, and heuristics are used to find a solution that minimises energy consumption. A (promising) evaluation is performed over large-scale simulation settings (i.e., 50 services, 100 to 1000 Fog nodes).

Similarly, Deng et al.[39] followed a hybrid approach to model FAPP and to determine the best trade-off between power consumption and network delays (exploiting $M/M/n$ Markov models to describe network capabilities). After decomposing FAPP into three sub-problems (power-delay vs fog communication, power-delay vs cloud communication and minimising WAN delay) balanced workload solutions are looked for exploiting different optimisation methods (i.e., convex optimisation, generalised Benders' decomposition and Hungarian method). A quite large simulation setup in MATLAB[62] was used to evaluate the proposed approach (i.e, from 30000 to 60000 nodes).

Unfortunately, none of the approaches previously discussed in this section released the code to run the experiments. Conversely, based on the hierarchical model of Skarlat et al.[36], Venticinque and[40] proposed a software platform to support optimal application placement in the Fog, within the framework of the CoSSMic European Project[63]. Envisioning resource, bandwidth and response time constraints, their approach permits to choose among a Cloud-only, a Fog-only or a Cloud-to-Fog deployment policy, which were evaluated in a small testbed (i.e., 1 Cloud, 1 Fog node) where a composite application (8 components) from Smart Energy scenarios[64] was deployed and run over emulated IoT data traces.

Finally, Cardellini et al.[41,42,43] discussed and released S-ODP, an open-source extension of Apache Storm that solves FAPP by means of the CPLEX[65] optimiser with the goal of minimising end-to-end application latency and availability of stream-based applications. Extensive experiments (i.e., up to 50 application components and up to 100 Fog nodes) showed scalability of the ILP approach (with respect to a traffic-aware extension of Storm[66]), which can be easily extended to include bandwidth constraints and a network-related objective function considering network usage, traffic and elastic energy computation.

3.1.3 | Other Algorithms

Bio-inspired Algorithms Genetic algorithms (GAs) implement meta-heuristics to solve optimisation and search problems based on bio-inspired operators such as mutation, crossover and selection[67]. Naturally, some of the reviewed works exploited such bio-inspired search algorithms to explore the solution space of FAPP, and to solve it. Wen et al.[44] surveyed Fog orchestration-related issues and offered a first description of the applicability of GAs and parallel GAs to FAPP.

Retaking the ILP model of Skarlat et al.[35] based on Fog colonies and hierarchical control nodes, Skarlat et al.[36] also proposed a GA solution implemented in iFogSim and compared to a greedy (first-fit) heuristic and to an exact optimisation obtained with CPLEX, over a small example (i.e., 5 application components, 10 Fog nodes). Whilst the first-fit strategy does not manage to guarantee user-defined application deadlines, both the exact solution and the GA do. Overall, the GA solutions are on average 40% more costly than the optimal ones, despite guaranteeing lower deployment delays.

Mennes et al.[45] required the user to provide a minimum reliability measure and a maximum number of replicas for each (multi-component) application to be deployed. It employed a distributed GA, by using Biased Random-Key arrays to represent solutions (instead of binary arrays). The placement ratio (placed/required) is used to evaluate the proposed algorithm., which showed near-optimal results against a small example (i.e., 10 applications, 5 Fog nodes) and faster execution time with respect to ILP solvers. Regrettably, open-source implementations are not provided for any of the works exploiting GA.

Game Theory Game theoretical models[68] – which describe well multi-agent systems where each agent aims at maximising its profit whilst minimising its loss – were fruitfully applied to solve FAPP in[46] and[47].

Zhang et al.[46] modelled FAPP as Stackelberg games between data service subscribers and data service providers, the latter owning fog nodes. A first game is used to determine the number of computing resource blocks that users should purchase (based on latency requirements, block prices and utility). A second game is used to help providers set their prices so to maximise revenues. A matching game is used to map providers to Fog nodes based on their preferences. And, finally, matching between fog nodes and subscribers is refined in order to be stable. The proposal was tested in a simulated MATLAB environment considering 120 service subscribers, 4 data providers, and 20 Fog nodes.

Similarly, Zhang et al.[47] describe a game among service providers, Fog nodes and service subscribers. The mapping between the Fog resources and service subscribers is determined to solve a student-to-project allocation problem. During the game, subscribers consider revenues, data transmission costs, providers' costs and latency to evaluate possible assignments. On the other hand, providers consider revenue from subscribers minus the cost of service delay.

Deep Learning Reinforcement learning trains software agents (with reward mechanisms) so that they learn policies determining how to react properly under different conditions[69]. To the best of our knowledge, only Tang et al.[48] exploited recent reinforcement learning techniques to solve FAPP. After defining a multi-dimensional Markov Decision Process to minimise communication delay, power consumption and migration costs, a (deep) Q-learning algorithm is proposed to support migration of application components hosted in containers or VMs. The proposal took into account user mobility and was evaluated over a medium-sized infrastructure (i.e., $\simeq$ 70 nodes) using real data about users mobility taken from San Francisco taxi traces.

Dynamic Programming Differently from the others, Souza et al.[49] modelled FAPP as a 0-1 multidimensional knapsack problem[70] with the objective of minimising a given objective function. Limited simulation results on a medium-sized example (40 application components, 6 Fog nodes, 3 Clouds) are provided by the authors. However, no details are given on how the solution is computed, and the code to run the experiments is not available.

Also Rahbari et al.[50] modelled FAPP as a knapsack problem, by considering the allocation of application modules to running VMs in a Fog infrastructure. iFogSim was used to simulate the proposed symbiotic organisms search algorithm, showing some improvements in energy consumption and network usage with respect to a First Come First Served allocation policy and traditional knapsack solvers.

Complex networks To the best of our knowledge, only Filiposka et al.[51] relied on network science theory to model and study FAPP, employing on a community detection method to partition the available Fog nodes into a hierarchical dendogram[71]. The dendrogram was then used to analyse different community partitions so to find the most suitable set of nodes to support the VMs that encapsulate the applications. The proposal was validated with CloudSim[72] over a quite large use case (i.e., 80 Fog nodes, and from 150 to 250 application components). The proposed community-based extension to CloudSim is, however, not available.

3.2 | Modelling

3.2.1 | Considered Constraints

The set of surveyed papers has also been analysed in terms of their modelling perspective. In a first section, the features of the constraints and parameters of the model are explained. As we observed in Table 1 , we organised the constraints of the model in a two-level taxonomy, considering four main groups: network, energy, computing nodes and applications.

We defined four features for the second level of the network constraints. Latency is the network feature related with the transmission time of the requests and responses through the network links. This has an important influence on Fog environments since one of the objectives of these domains is to reduce the response time of the cloud-based applications. Consequently, this metric is commonly included in the optimisation works. Similarly, the bandwidth, quantity of data that the network link is able to transmit per unit of time, is also important to achieve this objective of reducing the user-perceived response time. In the field of communication networks, the availability and the influence of failures in the transmission is also important, and we additionally included the link reliability in our analysis. Finally, the topology was also included to highlight the articles that consider the topological distribution of the Fog devices, or the influence of the region of the network where the applications are deployed or users are connected, or even if the location of special IoT devices (such as, sensors or actuators) was considered in the analysis of network and in the application placement decision.

The constraints related with the Fog nodes are analysed from a hardware and software point of view. In the first case, the constraints refer to different types of resources (e.g., processors, memory, storage) or to a generic resource capacity that can represent any considered resource type. In the second case, the constraints are related to software dependencies that should be available at the node hosting a component (e.g., OS, libraries/frameworks, language support).

Energy constraints, commonly related to the power consumption of the hardware elements, are important in Fog domain for two main reasons. The first one is a constraint inherited from Cloud domains. Both Cloud and Fog need a high level of power consumption to give service to the users. Small improvements in the power consumption can result on important savings in energy. Additionally, in the case of Fog domains, mobile and battery-powered devices are also involved, taking the energy even more important in the optimisation of the architecture.

For the case of application constraints, we classified the articles by analysing the number of user request (workload), the interrelation between the modules of the applications (dependencies), or user-defined conditions or preferences for the application placement (user preferences).

Network Constraints

The first set of constraints and decision variables are related to the network. The most common metric in this set is the network latency, but others such as the bandwidth and the topology are also quite usual. On the contrary, link reliability is seldom considered in the research works. Various works solely considered the latency among the metrics related with the network[16,32,39,46,47,49]. But, latency was also usually considered along with the bandwidth[14,18,19,20,21,22,25,33,48]. Bandwidth was also considered in[26,28], but along with hardware constraints. On the contrary, the studies that included link reliability also included other network constraints, such as bandwidth[24,45], latency and topology[43], or latency, bandwidth and topology[33].

Topology is the most diverse metric from the ones of the networks, and we give a more detailed attention to this case. Several works deal with the application placement by considering that exist statically defined Fog colonies[35,36,37], or sets of devices that are managed by a controller node. Thus, a twofold placement based on those colonies was proposed, with a previous mapping of applications in colonies and a second phase of mapping applications to the devices inside a colony. Venticinque and Amato[40] also considered Fog colonies, and they additionally included the network bandwidth constraint.

Yang et al.[29] considered that the devices with the capacity to place services are organised in cloudlets, and those cloudlets are assigned to cover a region of the network. A cloudlet is defined as an abstraction tier between the user and the cloud that provides processing capabilities. It can be a static infrastructure connected to the wireless access network, or augmented routers and switches in the wireless access network. Consequently, the use of a cloudlet would depend on the gateways where the users are connected to and in the topology of the network (reflected in the coverage definition). In this work, the authors also considered the latency of the network as a constraint.

Ottenwalder et al.[18] defined a federation of hierarchy brokers implemented with a combination of cloud data centres and Fog devices. They also studied the mobility pattern of the users and how they are connected to different nodes in the network topology. The study of the mobility of the users across different parts of the topology of the network is also included in the studies of Tang et al.[48] and Filiposka et al.[51]. Additionally, this last study used the topological structure of the network as the method to determine the mapping among applications and nodes. We have also considered that the number of node elements between the users and the applications or between nodes allocating modules of the same application is also a feature related to the topology of the network. Under these conditions, several works[14,17,38] took into account the hop distances of the Fog nodes as input variables of the optimisation process.

Finally, there is a small set of papers that also considered the interactions between applications and IoT devices[20,21], i.e., if the application modules need to be executed in devices with specific hardware requirements or with sensor or actuator elements. We have classified those works into the topology feature since the application placement depends on the characteristics of the network components (nodes).

Node Constraints

The analysis of Table 1 , clearly identified that most of the articles considered constraints about the hardware of the devices. On the contrary, software constraints are only included in a very small number of papers.

The most common feature related to the hardware was the resource capacity of the Fog devices as a constraint element for the number of services that can be placed into them. The resource capacity is usually modelled as a vector or set of elements, one for each of the hardware elements considered in the Fog devices or the network. Examples of those vectors for the case of the Fog devices are: considering CPU[14,48]; considering CPU and storage[28]; CPU, RAM and storage[30,36]; considering only storage[29,31,35]. Examples of resource capacity vectors both for Fog nodes and network are: considering CPU, RAM and network bandwidth[26,45]; considering CPU, RAM, storage and network bandwidth[23,25,40]; considering processing and transmission capacities[33].

Other papers defined a general resource model and they did not focus on the specific resource components[16,19,20,21,51]. Mahmud et al.[34] also considered a general resource value, but they additionally defined the service demand and device capacity in terms of the expected and offered processing times. On the contrary, this model was sometimes simplified to a scalar value which represents a general capacity unit[17,25,41,42,43], or with general resources slots[32,49]. Finally, some other papers defined the hardware resources of the Fog computing nodes, but they did not include this constraint in the optimisation process to simplify it[24].

In other papers, the hardware is considered as variables of a fitness function. The fitness function is used to measure the suitability of a device to place the application modules. Rahbari et al.[50] included the CPU and network usages in the fitness function. For example, Skarlat et al.[35] defined the type of service that determines the additional hardware that the device needs to include to be allocated in.

As we previously mentioned, constraints related to the software features of the Fog computing nodes are just considered in a small number of studies. For example, Brogi and Forti [19,20,21,22] characterised the Fog devices and applications with software capabilities (e.g., operating system, platforms, frameworks).

Energy Constraints

Energy is also considered as an input variable or constraint in some of the analysed works. For example, Barcelo et al. [33] characterised the fog devices with their energy resources, such as power grid, battery, or energy costs, and the network links with, also, the energy costs. The limitations of devices powered with batteries are taken into account to guarantee lifetime requirements. Souza et al. [49] proposed the concept of energy cells to measure the energy consumed by the underlying devices, and the optimisation took into account the number of available energy cells for the mapping of applications and devices. The objective was to minimise the excessive energy consumption in the most energy constrained devices.

Application Constraints

Three types of application constraints have been considered for the classification of the papers in the survey: constraints related to the dependencies between the modules or services of the applications; the workload generated over the applications; and if the users are able to define any kind of preference in the deployment of the services.

These constraints were not usually considered together in the papers of the survey. There were only four papers [17,18,28,29] that included more than one of those application constraints, more concretely, the workload and dependency constraints. Guerrero et al. [17] considered both, the request rate of the applications to prioritise the placement of some application, and the interrelation between the services. They considered that the interrelated services of an application should be allocated in devices in the network shortest path between the user and the Cloud provider, and the placement order was determined by the topological order of the services, placing the initial services closer to the users. Yang et al. [29] analysed the workload generated from each region of the Fog domain and the dependencies between the application modules. Ottenwalder et al. [18] defined the dependencies of the applications as an operator graph and also considered the load over the system to find a migration plan for the operators (services) across the devices. Arkian et al. [28] considered the request rates of the applications between the fog devices and the association between the consumers and the data. Mahmud et al. [34] defined the user expectation metric that includes metrics such as the service access rate, demanded resources, and expected processing time, and this metrics is used to prioritise the placement of the applications. Additionally, the interrelation of the devices was also included in the status metric (proximity, resource availability and processing speed).

The constraint about the dependency of the application models is the most common one. In the work of Skarlat et al. [36], the interactions between the application services were considered to calculate the theoretical response time of applications. Since they organised the devices in colonies, if two interrelated services are allocated in different colonies, the response time would be increased. Consequently, the optimisation algorithm should co-locate the services of an application in the same colony. Zeng et al. [31] considered the interrelation between the storage (data placement) and the execution (scheduling) of the applications to minimise the application completion time by optimising the influence of the I/O time and the task completion time. Wang et al. [24] proposed two algorithms which were defined by considering the interrelations between the modules of the applications, represented by a graph. The first algorithm was defined for linear-like applications and the second one for tree-like applications. Other application shapes were not considered. Rahbari et al. [50] considered a symbiotic organisms search that used the relationships between the virtual machines (VM) to decide the allocation of the services on those VMs. Mahmud et al. [16] presented a decentralized policy for the inter-dependent application modules that simultaneously considered the service access delay, service delivery time and device communication delays.

User preferences were only included in three papers. Brogi et al. [20] created an algorithm that suggests to the user several alternatives for the deployment among a set of eligible candidates. But they left to the users to choose how to trade-off the QoS metrics and the Fog resources consumptions, or even taking into account other types of considerations. In the work of Skarlat et al. [35], the users are allowed to define a deadline for the applications to warranty a level of QoS. Finally, Cardellini et al. [43] stated that their solution could easily include user-related constraints, such as service co-location, bandwidth limitation, even tag-based constraints, but they didn't implement them.

3.2.2 | Optimised & Comparison Metrics

The ultimate objective of the FAPP is to optimise one or several metrics of the Fog domain. Usually, in a complex system such as a Fog architecture, some of the most common optimisation objectives are opposite, i.e., they cannot be both optimised and a trade-off between them needs to be fixed. For example, if resource usage is optimised, probably the QoS of the application would be damaged. Some of the papers also studied the influence of their proposals in additional metrics to the ones of their optimisation metrics. By this, a general view of the effects of the FAPP proposal in the system is provided. Thus, in this section, apart from the optimised metrics, we have also included all those metrics that have been studied, evaluated, and analysed in the results of the papers in the survey.

From the analysis of the papers, we defined a taxonomy of eight elements to classify the articles in terms of the optimised/studied metrics, as it can be seen in Table 1 . Despite this, the papers are presented in only six groups to avoid repetition of references. In some cases, these groups are created by the union of related metrics, such as the network delay and the node execution time, that are commonly studied together. For other metrics, such as the network bandwidth and the hardware of the nodes, they resulted in not being always related to the same other metrics, and the papers that included them were already explained in other groups of metrics. If we had created a group for bandwidth or node hardware, it would have resulted in repeating the papers in several groups.

Network Delay and Execution Time

Probably, the most important contribution of the Fog computing paradigm is to reduce the latency of cloud-based applications by placing them closer to the users. This latency includes the communication time and the node execution time.

In a first set, we present the article that included both communication and node execution time. Skarlat et al.[37] obtained a decrease of the network delay and the execution time up to 39% with regard to a baseline policy. Zeng et al.[31] also considered both network and execution times. But for the latter, they studied the computation and input/output operations separately. Cardellini et al.[43] also studied other metrics, such as availability or network traffic, apart from the network delay and service time. The work proposed by Xia et al.[23] minimised the application response time to improve the number of requests that are served before a fixed application deadline. Gupta et al.[14] and Mahmud et al.[15] presented some baseline policies to validate their Fog simulator, and they studied the total execution time of the user requests together with the network usages and the power consumption. They compared their policies with the case of requesting services only from the Cloud provider.

In the second set, we present the papers including the network delay but that did not consider the node execution time. There are two papers that solely considered the network delay[30,32]. Souza et al.[32] reduced the execution time of the application service by measuring the allocated time slots, and the results proved the reduction of the high delays of requesting the services to the Cloud provider. In some other cases, the network latency is not measured directly, and indicators such as the hop count are considered[27]. Guerrero et al.[17] also minimised the hop count between the users and the placement of the services with the objective to reduce the application latency, and the network usage. The number of migrations was also included in the metrics evaluated in the experimental phase.

Finally, the number of papers considering the execution time but without including the network delay is very reduced. For example, Wen et al.[44] solely considered the execution time, showing improvement around the 30%. Additionally, Mennes et al.[45] had the objective of maximising the number of applications deployed on Fog devices, but they also measured the execution time of the application in the results of their experiments.

QoS-assurance

Some approaches, instead of minimising network or execution times, aimed at increasing the Quality of Service (QoS) satisfaction. QoS is directly related to those times, but its improvement does not necessarily result in a reduction of the times. For example, the QoS can be measured as the percentage of requests that are executed before a time deadline. The improvement of the QoS involves then to keep the execution times below this threshold, but the minimisation of the times is not required.

Brogi et al.[19,20,21,22] studied the QoS in terms of latency and network bandwidth. They also extended the work and considered resource consumption of the Fog devices and they proposed a novel cost model for Fog devices. In the work of Mahmud et al.[16], the objective of the optimisation was to reduce the number of active Fog devices. But this optimisation was constrained with the warranty of satisfying the QoS level, i.e., shorter execution times than the application deadlines. Consequently, results about the percentage of deadline satisfaction were presented, showing important improvements. Brogi et al.[35,36] maximised the resource usage of the Fog devices to maximise the number of applications deployed on the Fog layer, while the latency and QoS are not damaged. The results showed that the Fog landscape was used for the 70% of the services, reducing 30% of the execution cost, without affecting the QoS.

Venticinque and Amato[40] considered the QoS in terms of the number of request and transactions processed per unit of time. The authors also included the results of the execution time and the resource usages of the devices. Cardellini et al.[41,42] also studied the QoS measured with data obtained from each node, such as utilization, availability, and network metrics. All the nodes are informed of the QoS of other nodes with the use of a gossip-based dissemination schema. The authors presented the experiment results in terms of application availability, application latency, network traffic and node utilization.

Mahmud et al.[34] defined three metrics to study the QoS of their proposal: network relaxation ratio, processing time reduction ratio and resource gain. Additionally, they also presented results about deadlines, costs and packet losses. Finally, Zhang et al.[47] optimised the QoS by providing a suitable utilization of the nodes. The algorithm ensures an optimal amount of hardware resources for the allocated applications.

Migrations

An important consequence of bringing the applications near to the users is the increase of the network traffic due to the application migrations. Consequently, some of the studies have dealt with the minimization of the number of migrations or their effects on the system.

Apart from the already cited work of Velasquez et al.[27], where the number of migrations was minimised along with the network delay, migrations were also optimised by Ottenwalder et al.[18], by minimising the network utilization without damaging the network latency, and by Yang et al.[29], where the migration was reduced along with the latency, the resource usage, and the provider cost.

Finally, Filiposka et al.[51] studied the cost of migrations to just migrate the applications if the obtained benefits are bigger than the overhead generated in the system by the movement of the application between nodes. They also studied the network latency in terms of hop counts.

Cost

Cost is a common minimisation objective in resource management proposals in computation-as-a-service domains. In Fog domain the evaluation of the cost is still in a very initial phase, but there are some preliminary works dealing with its optimisation.

Apart from the already cited works[21,22,37], there are other four papers that included the cost in their evaluation and optimisation. Zhang et al.[46] studied the cost of the transmission delay and the service execution. Arkian et al.[28] optimised the overall cost of the deployment of the applications, while the QoS is guaranteed, by allocating them in the devices with the smallest costs. They additionally studied the power consumption and the service latency. Wang et al.[24] addressed the minimisation of the cost of each physical node and link, ensuring that the devices would not be overloaded.

Finally, Hong et al.[25] optimised the cost of the application deployment by selection of the provider from a federated pool of Cloud providers. Additionally, other three objectives were also minimised: the available resources on the devices, the network distance between the users and the services, and between the nodes which allocate interrelated services.

Energy

The energy and the power consumption are also one of the most important concerns among the authors of the analysed papers. The energy optimisation has been addressed from different point of views. For example, Barcelo et al.[33] defined a linear characterised function of the energy cost, and they minimised it. Their proposal was able to reduce the overall power consumption by more than an 80%.

Huang et al.[38] were more focused on the reduction of the communication energy cost, by placing in the same device interrelated services and, consequently, reducing the number of communications between devices and their hop count. The experiments showed an improvement of 10% energy savings. In other cases, the energy was optimised along with other metrics: (a) the trade-off between energy consumption and end-user delay[39], (b) optimisation of the energy consumption with the network usage and the execution cost[50], resulting in improvements of 18% for the energy consumption, 1.17% for the network usage and 15% for the execution cost, (c) balancing the energy consumption and the resource usage in the fog devices[49], (d) or reducing the application execution time, the power consumption of the devices, and the cost of the services migrations[48].

In a few cases, the energy was not the optimisation objective, but it was analysed to validate the benefits of the proposals. For example, Taneja et al.[26] mainly addressed the minimisation of the application execution time, but they also analysed other common metrics such as network usage and energy consumption.

4 | CONCLUSIONS: OPEN PROBLEMS AND RESEARCH CHALLENGES

In this section, we conclude by pointing to some open challenges and future directions that can be explored to better approach FAPP.

First of all, whilst search and mathematical programming have been thoroughly investigated in the literature, more modern techniques are to be applied to FAPP. Particularly, based on the first promising results the attained on FAPP, it would be interesting to study further the applicability of other genetic or evolutionary algorithms, swarm optimisation techniques, deep learning, network science and game theoretical approaches to FAPP. Additionally, very few approaches available nowadays are distributed, whilst those solutions might scale better and show stronger resilience in highly dynamic Fog infrastructures. In these regards, devising decentralised algorithms to be applied to solve FAPP online and without relying on control nodes would be important and crucial to the success of the Fog.

Naturally, whilst exploring new proposals, it is important to compare them to the related state-of-the-art techniques with respect to the execution time (utterly important in Fog scenarios) and – possibly – to the achieved results (in terms of a standard set of common metrics). To this end, the design of a set of benchmark examples (based on established standards like TOSCA Yaml[73]) would make it possible to systematically compare and contrast different approaches to FAPP as well as to quantify performance improvements or degradation. Such examples should possibly consider all the attributes that have been modelled in the literature (e.g., hardware, software, IoT, QoS, energy) and should come with an optimal candidate solution, determined by exhaustive techniques. Another possibility towards this direction is to exploit the solid theory of complex networks both to generate test topologies and to analyse the obtained results.

From a modelling perspective, none of the surveyed studies considered security aspects when determining optimal application placements. Security will play a crucial role in the success of the Fog paradigm and it represents a concern that should be addressed *by-design* at all architectural levels[9]. Therefore, there is a clear need to (quantitatively) evaluate whether an application will have its security requirements fulfilled by the (Cloud and Fog) nodes chosen for the deployment of its components. Furthermore, due to the mission-critical nature of many Fog applications (e.g., e-health, disaster recovery), it is important that the techniques employed to perform security analyses to solve FAPP are well-founded and, possibly, explainable.

Similarly, mobility of Fog and IoT nodes was considered in very few of the reviewed works, even though it is a parameter that cannot be neglected in Fog scenarios. Indeed, many Fog verticals (e.g., autonomous vehicles, flying drones) include nodes that move, and that continuously and opportunistically connect/disconnect from other devices. In general, few authors considered infrastructure variations due for instance to changing topologies (i.e., available nodes), network traffic (i.e., latency, bandwidth), or workload conditions (i.e., available node hardware). Future research in the field of FAPP should, therefore, devise and tweak novel models that account for these typical traits of Fog computing, so to understand how application placement can be adaptively adjusted to this phenomenon.

Analogously, Fog computing exists in continuity with both the IoT and the Cloud. Hence, more effort should be made to consider the integration and simultaneous management of these three entities which was neglected in many works. Particularly, QoS attributes that define the reachability of IoT and Cloud nodes from different Fogs showed to be key in leading the search towards better placements. In line with this effort of considering the Cloud to Things continuum as a unique system, the possibility of some application components to be deployed in different flavours depending on the resources of target deployment nodes (like in *Osmotic Computing*[74]) is to be studied yet. Overall, understanding how the proposed methodologies could be applied exploited to work with production-ready tools for Fog application management (e.g., CISCO FogDirector) would be surely of interest.

To conclude, most of the experiments were carried out in small to medium scale simulated environments, often without disclosing the codebase to repeat them. Future research in this field should prototype more versatile and well-documented simulators that can be used to experiment with different strategies or algorithms and that permit evaluating proposals over large-scale, lifelike, examples. Last but not least, real Fog testbeds could be realised with the help of those industries that are currently shaping Fog computing, so to actually test and asses the proposed solutions.

ACKNOWLEDGEMENTS

This research was partially supported by the University of Pisa project "*DECLWARE: Declarative methodologies of application design and deployment*" (PRA_2018_66) and by the Spanish Government (Agencia Estatal de Investigación) and the European Commission (Fondo Europeo de Desarrollo Regional) through grant number TIN2017-88547-P (MINECO/AEI/FEDER, UE).

Author contributions

The authors are ordered alphabetically. All the authors contributed equally to this work.

References

1. CISCO . Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf[Online; accessed 27-July-2018]; 2015.

2. CISCO . Cisco Global Cloud Index: Forecast and Methodology, 20162021 https://www.cisco.com/c/en/us/solutions/collateral/service-p accessed 27-July-2018]; 2018.

3. Manyika James, Chui Michael, Brown Brad, et al. *Big data: The next frontier for innovation, competition, and productivity.* 2014.

4. Hashem Ibrahim Abaker Targio, Yaqoob Ibrar, Anuar Nor Badrul, Mokhtar Salimah, Gani Abdullah, Khan Samee Ullah. The rise of Big Data on cloud computing: Review and open research issues. *Information Systems.* 2015;47:98–115.

5. Shi Weisong, Dustdar Schahram. The Promise of Edge Computing. *Computer.* 2016;49(5):78–81.

6. Dastjerdi A. V., Buyya R.. Fog Computing: Helping the Internet of Things Realize Its Potential. *Computer.* 2016;49(8):112-116.

7. Bonomi Flavio, Milito Rodolfo A., Natarajan Preethi, Zhu Jiang. Fog Computing: A Platform for Internet of Things and Analytics.. in *Big Data and Internet of Things* (Bessis Nik, Dobre Ciprian, eds.) Studies in Computational Intelligence, vol. 546: Springer 2014 (pp. 169-186).

8. IEEE . IEEE 1934-2018 - IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing http://standards.ieee.org/findstds/standard/1934-2018.html[Online; accessed 27-July-2018]; 2018.

9. Consortium OpenFog. OpenFog Reference Architecture for Fog Computing https://www.openfogconsortium.org/wp-content/uploads/C accessed 27-July-2018]; 2017.

10. Iorga Michaela, Feldman Larry, Barton Robert, Martin Michael J., Goren Nedim S., Mahmoudi Charif. *Fog Computing Conceptual Model.* 500-325: National Institute of Standards and TechnologyGaithersburg; 2018.

11. Brogi Antonio, Forti Stefano, Ibrahim Ahmad, Rinaldi Luca. Bonsai in the Fog: An active learning lab with Fog computing. in *Fog and Mobile Edge Computing (FMEC), 2018 Third International Conference on*:79–86IEEE; 2018.

12. Aazam Mohammad, Zeadally Sherali, Harras Khaled A.. Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities. *Future Generation Computer Systems.* 2018;.

13. Xu D., Li Y., Chen X., et al. A Survey of Opportunistic Offloading. *IEEE Communications Surveys Tutorials.* 2018;:1-1.

14. Gupta Harshit, Vahid Dastjerdi Amir, Ghosh Soumya K, Buyya Rajkumar. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience.* 2017;47(9):1275–1296.

15. Mahmud Redowan, Buyya Rajkumar. Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit. in *Fog and Edge Computing: Principles and Paradigms* (Buyya Rajkumar, Srirama Satish N., eds.)Wiley; 2018. In press.

16. Mahmud Redowan, Ramamohanarao Kotagiri, Buyya Rajkumar. Latency-aware Application Module Management for Fog Computing Environments. *ACM Transactions on Internet Technology (TOIT).* 2018;.

17. Guerrero Carlos, Lera Isaac, Juiz Carlos. A lightweight decentralized service placement policy for performance optimization in fog computing. *Journal of Ambient Intelligence and Humanized Computing.* 2018;.

18. Ottenwälder Beate, Koldehofe Boris, Rothermel Kurt, Ramachandran Umakishore. MigCEP: Operator Migration for Mobility Driven Distributed Complex Event Processing. in *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*DEBS '13:183–194ACM; 2013; New York, NY, USA.

19. Brogi A., Forti S.. QoS-Aware Deployment of IoT Applications Through the Fog. *IEEE Internet of Things Journal.* 2017;4(5):1185-1192.

20. Brogi Antonio, Forti Stefano, Ibrahim Ahmad. How to best deploy your Fog applications, probably. in *Proceedings of 1st IEEE International Conference on Fog and Edge Computing* (Rana O., Buyya R., Anjum A., eds.); 2017.

21. Brogi Antonio, Forti Stefano, Ibrahim Ahmad. Deploying Fog Applications: How Much Does It Cost, By the Way?. in *Proceedings of the 8th International Conference on Cloud Computing and Services Science, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018.*:68–77; 2018.

22. Brogi Antonio, Forti Stefano, Ibrahim Ahmad. Predictive Analysis to Support Fog Application Deployment. in *Fog and Edge Computing: Principles and Paradigms* (Buyya Rajkumar, Srirama Satish N., eds.)Wiley; 2018. In press.

23. Xia Ye, Etchevers Xavier, Letondeur Loïc, Coupaye Thierry, Desprez Frédéric. Combining hardware nodes and software components ordering-based heuristics applications in the fog. in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*:751–760ACM; 2018.

24. Wang S., Zafer M., Leung K. K.. Online Placement of Multi-Component Applications in Edge Computing Environments. *IEEE Access.* 2017;5:2514-2533.

25. 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)*:1-6; 2016.

26. 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)*:1222-1228; 2017.

27. Velasquez Karima, Abreu David Perez, Curado Marilia, Monteiro Edmundo. Service placement for latency reduction in the internet of things. *Annals of Telecommunications.* 2017;72(1-2):105–115.

28. Arkian Hamid Reza, Diyanat Abolfazl, Pourkhalili Atefe. MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications. *Journal of Network and Computer Applications.* 2017;82:152 - 165.

29. Yang L., Cao J., Liang G., Han X.. Cost Aware Service Placement and Load Dispatching in Mobile Cloud Systems. *IEEE Transactions on Computers.* 2016;65(5):1440-1452.

30. Gu L., Zeng D., Guo S., Barnawi A., Xiang Y.. Cost Efficient Resource Management in Fog Computing Supported Medical Cyber-Physical System. *IEEE Transactions on Emerging Topics in Computing.* 2017;5(1):108-119.

31. Zeng D., Gu L., Guo S., Cheng Z., Yu S.. Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System. *IEEE Transactions on Computers.* 2016;65(12):3702-3712.

32. Souza V. B. C., RamÃŋrez W., Masip-Bruin X., MarÃŋn-Tordera E., Ren G., Tashakor G.. Handling service allocation in combined Fog-cloud scenarios. in *2016 IEEE International Conference on Communications (ICC)*:1-5; 2016.

33. Barcelo M., Correa A., Llorca J., Tulino A. M., Vicario J. L., Morell A.. IoT-Cloud Service Optimization in Next Generation Smart Environments. *IEEE Journal on Selected Areas in Communications.* 2016;34(12):4077-4090.

34. Mahmud Redowan, Srirama Satish Narayana, Ramamohanarao Kotagiri, Buyya Rajkumar. Quality of Experience (QoE)-aware placement of applications in Fog computing environments. *Journal of Parallel and Distributed Computing.* 2018;.

35. Skarlat O., Nardelli M., Schulte S., Dustdar S.. Towards QoS-Aware Fog Service Placement. in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*:89-96; 2017.

36. Skarlat Olena, Nardelli Matteo, Schulte Stefan, Borkowski Michael, Leitner Philipp. Optimized IoT service placement in the fog. *Service Oriented Computing and Applications.* 2017;11(4):427–443.

37. Skarlat O., Schulte S., Borkowski M., Leitner P.. Resource Provisioning for IoT Services in the Fog. in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*:32-39; 2016.

38. Huang Zhenqiu, Lin Kwei-Jay, Yu Shih-Yuan, Hsu Jane Yung-jen. Co-locating services in IoT systems to minimize the communication energy cost. *Journal of Innovation in Digital Ecosystems.* 2014;1(1-2):47–57.

39. Deng R., Lu R., Lai C., Luan T. H.. Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. in *2015 IEEE International Conference on Communications (ICC)*:3909-3914; 2015.

40. Venticinque Salvatore, Amato Alba. A methodology for deployment of IoT application in fog. *Journal of Ambient Intelligence and Humanized Computing.* 2018;.

41. Cardellini Valeria, Grassi Vincenzo, Lo Presti Francesco, Nardelli Matteo. Distributed QoS-aware scheduling in Storm. in *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems*:344–347ACM; 2015.

42. Cardellini Valeria, Grassi Vincenzo, Presti Francesco Lo, Nardelli Matteo. On QoS-aware scheduling of data stream applications over fog computing infrastructures. in *Computers and Communication (ISCC), 2015 IEEE Symposium on*:271–276IEEE; 2015.

43. Cardellini Valeria, Grassi Vincenzo, Lo Presti Francesco, Nardelli Matteo. Optimal Operator Placement for Distributed Stream Processing Applications. in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*DEBS '16:69–80ACM; 2016; New York, NY, USA.

44. Wen Zhenyu, Yang Renyu, Garraghan Peter, Lin Tao, Xu Jie, Rovatsos Michael. Fog orchestration for internet of things services. *IEEE Internet Computing.* 2017;21(2):16–24.

45. Mennes R., Spinnewyn B., LatrÃľ S., Botero J. F.. GRECO: A Distributed Genetic Algorithm for Reliable Application Placement in Hybrid Clouds. in *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*:14-20; 2016.

46. Zhang Huaqing, Xiao Yong, Bu Shengrong, Niyato Dusit, Yu F Richard, Han Zhu. Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining Stackelberg game and matching. *IEEE Internet Things J..* 2017;4(5):1204–1215.

47. Zhang Huaqing, Zhang Yanru, Gu Yunan, Niyato Dusit, Han Zhu. A hierarchical game framework for resource management in fog computing. *IEEE Communications Magazine.* 2017;55(8):52–57.

48. Tang Z., Zhou X., Zhang F., Jia W., Zhao W.. Migration Modeling and Learning Algorithms for Containers in Fog Computing. *IEEE Transactions on Services Computing.* 2018;:1-1.

49. Souza V. B., Masip-Bruin X., Marin-Tordera E., Ramirez W., Sanchez S.. Towards Distributed Service Allocation in Fog-to-Cloud (F2C) Scenarios. in *2016 IEEE Global Communications Conference (GLOBECOM)*:1-6; 2016.

50. Rahbari D., Nickray M.. Scheduling of fog networks with optimized knapsack by symbiotic organisms search. in *2017 21st Conference of Open Innovations Association (FRUCT)*:278-283; 2017.

51. Filiposka S., Mishev A., Gilly K.. Community-based allocation and migration strategies for fog computing. in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*:1-6; 2018.

52. Russell Stuart J, Norvig Peter. *Artificial intelligence: a modern approach.* Pearson Education Limited,; 2010.

53. Weaveworks , ContainerSolutions . Socks Shop - A Microservices Demo Application https://microservices-demo.github.io/[Online; accessed 27-July-2018]; 2016.

54. Varga András, Hornig Rudolf. An overview of the OMNeT++ simulation environment. in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*:60ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering); 2008.

55. Maio V. De, Brandic I.. First Hop Mobile Offloading of DAG Computations. in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*:83-92; 2018.

56. Letondeur Loïc, Ottogalli François-Gaël, Coupaye Thierry. A demo of application lifecycle management for IoT collaborative neighborhood in the Fog: Practical experiments and lessons learned around docker. in *Fog World Congress (FWC), 2017 IEEE*:1–6IEEE; 2017.

57. Chowdhury Mosharaf, Rahman Muntasir Raihan, Boutaba Raouf. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking (TON).* 2012;20(1):206–219.

58. Sinha SM. *Mathematical Programming: Theory and Methods.* Elsevier; 2005.

59. Gurobi Optimization LLC. Gurobi Optimizer Reference Manual http://www.gurobi.com[Online; accessed 27-July-2018]; 2018.

60. Mitchell Stuart, Consulting Stuart Mitchell, OâĂŹSullivan Michael, Dunning Iain. *PuLP: A Linear Programming Toolkit for Python.* : Department of Engineering Science, The University of Auckland, Auckland, New Zealand; 2011.

61. Guéret Christelle, Prins Christian, Sevaux Marc, Heipcke Susanne. *Applications of optimization with Xpress-MP.* Dash optimization London, England.; 2002.

62. MathWorks . MATLAB http://www.mathworks.com/products/matlab.html[Online; accessed 27-July-2018]; .

63. CoSSMiC – Collaborating Smart Solar-powered Microgrids http://cossmic.eu/[Online; accessed 27-July-2018]; .

64. Jiang S., Venticinque S., Horn G., Hallsteinsen S., Noebels M.. A distributed agent-based system for coordinating smart solar-powered microgrids. in *2016 SAI Computing Conference (SAI)*:71-79; 2016.

65. IBM ILOG CPLEX Optimizer http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/[Online; accessed 27-July-2018]; 2010.

66. Xu J., Chen Z., Tang J., Su S.. T-Storm: Traffic-Aware Online Scheduling in Storm. in *2014 IEEE 34th International Conference on Distributed Computing Systems*:535-544; 2014.

67. Bakirtzis Anastasios, Kazarlis Spyros. Genetic algorithms. *Advanced Solutions in Power Systems: HVDC, FACTS, and Artificial Intelligence: HVDC, FACTS, and Artificial Intelligence.* 2016;:845–902.

68. Tadelis Steven. *Game theory: an introduction.* Princeton University Press; 2013.

69. Goodfellow Ian, Bengio Yoshua, Courville Aaron, Bengio Yoshua. *Deep learning.* MIT press Cambridge; 2016.

70. Cormen Thomas H, Leiserson Charles E, Rivest Ronald L, Stein Clifford. *Introduction to algorithms.* MIT press; 2009.

71. Barabási Albert-László, Pósfai Márton. *Network science.* Cambridge university press; 2016.

72. Calheiros Rodrigo N, Ranjan Rajiv, Beloglazov Anton, De Rose César AF, Buyya Rajkumar. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience.* 2011;41(1):23–50.

73. Palma Derek, Rutkowski M, Spatzier T. TOSCA Simple Profile in YAML Version 1.0. *OASIS Committee Specification.* 2016;.

74. Villari Massimo, Fazio Maria, Dustdar Schahram, Rana Omer, Ranjan Rajiv. Osmotic computing: A new paradigm for edge/cloud integration. *IEEE Cloud Computing*. 2016;3(6):76–83.

□