Evaluation of fog application placement algorithms: a survey

Sven Smolka¹ · Zoltán Ádám Mann¹

Received: 21 May 2021 / Accepted: 5 November 2021 / Published online: 11 February 2022
© The Author(s) 2022

Abstract

Recently, the concept of cloud computing has been extended towards the network edge. Devices near the network edge, called fog nodes, offer computing capabilities with low latency to nearby end devices. In the resulting fog computing paradigm (also called edge computing), application components can be deployed to a distributed infrastructure, comprising both cloud data centers and fog nodes. The decision which infrastructure nodes should host which application components has a large impact on important system parameters like performance and energy consumption. Several algorithms have been proposed to find a good placement of applications on a fog infrastructure. In most cases, the proposed algorithms were evaluated experimentally by the respective authors. In the absence of a theoretical analysis, a thorough and systematic empirical evaluation is of key importance for being able to make sound conclusions about the suitability of the algorithms. The aim of this paper is to survey how application placement algorithms for fog computing are evaluated in the literature. In particular, we identify good and bad practices that should be utilized respectively avoided when evaluating such algorithms.

Keywords Fog computing · Edge computing · Application placement · Empirical evaluation · Simulation

1 Introduction

Cloud computing has been very successful in providing virtually unlimited processing and storage capacity, accessible from anywhere. However, with the rapid growth in the number of devices connected to the Internet and using cloud services, new

✉ Zoltán Ádám Mann
zoltan.mann@paluno.uni-due.de

Sven Smolka
sven.smolka@paluno.uni-due.de

¹ University of Duisburg-Essen, Essen, Germany

requirements have emerged [29]. Some applications require real-time processing of data from dispersed end devices, for which the delay of sending data to the cloud may be prohibitive. Also, if myriads of end devices send their data to the cloud, this may lead to a network overload.

To handle a massive number of dispersed end devices, new paradigms have been proposed to extend cloud computing to the network edge [13]. The main idea of these approaches is to provide, in addition to centralized cloud data centers, geographically distributed devices, called *fog nodes*. Fog nodes can be either existing networking equipment (e.g., routers or gateways) with spare compute capacity, or devices specifically deployed for this purpose. End devices can offload computing tasks to nearby fog nodes, ideally just 1–2 network hops from the end device, so that such offloading incurs low latency.

Fog computing (also called edge computing) is thus characterized by an infrastructure comprising cloud data centers, fog nodes, and end devices [63,97]. To take advantage of this heterogeneous infrastructure, *fog application placement algorithms* must decide for each component of an application on which node of the infrastructure the given component should be placed. The placement of an application can have considerable impact on key quality metrics, including performance, resource consumption, energy consumption, and costs [10,18,62]. Different algorithms may target specific types of applications (e.g., stream processing applications) or be generic.

Finding the best placement for a given application (or set of applications) and a given infrastructure is difficult. Several formulations of this problem have been proven to be NP-hard. Nevertheless, several algorithms have been proposed—mostly heuristics—to solve this problem [18,82].

The applicability, effectivity, and efficiency of proposed algorithms is in most cases demonstrated with an experimental evaluation. To allow well-founded statements about the performance of the algorithms, the experimental evaluation should follow a systematic methodology. The algorithm must be evaluated on a diverse set of relevant benchmark problem instances, with realistic and varying parameter values, also comparing against other state-of-the-art algorithms, and the results should be statistically analyzed. Following a sound evaluation methodology is especially critical for heuristics lacking rigorous quality guarantees, which is the case for most proposed approaches. Unfortunately, in several cases, the experimental evaluation suffers from critical issues, such as using only unrealistically small problem instances or comparing the proposed algorithm only to trivial competitors. In such cases, the evaluation results do not reveal how good the given algorithm really is. This heavily impedes further scientific progress on fog application placement algorithms.

This paper aims at giving an account of the practices in the relevant scientific literature concerning the evaluation of fog application placement algorithms. To this end, we survey roughly 100 papers proposing fog application placement algorithms, and analyze how they evaluate these algorithms. We categorize the evaluation in the papers according to 40 criteria, such as the used evaluation environment, the number of used problem instances, the baseline algorithms for comparison, and the used evaluation metrics. We analyze the findings, identifying good and bad practices, and also discussing the evolution over time (i.e., comparing newer to older papers).

Fig. 1 The fog application placement problem (FAPP) consists of mapping the application(s) on the infrastructure

Previous surveys on fog application placement [18,82] focus on problem variants and algorithms, only briefly mentioning how algorithms were evaluated. However, as the field is maturing, the solid evaluation of proposed algorithms is increasingly important. To our knowledge, our paper is the first comprehensive survey of the evaluation of fog application placement algorithms.

2 Preliminaries

Brogi et al. [18] define the fog application placement problem (FAPP) as follows. Given is the description of an infrastructure I and an application A . I contains information about the nodes (cloud data centers, fog nodes, end devices), including their capacity (CPU, RAM etc.), and about the links between nodes, with information such as latency and bandwidth of each link. Similarly, A contains information about the components of the application(s)¹ and the connectors between components. A “connector” is a relationship between two components, allowing them to invoke each other and exchange data.

The aim of the FAPP is to find a mapping (placement) of application components on infrastructure nodes (see Fig. 1). A set R of requirements must be fulfilled, including resource requirements of the components (in terms of CPU, RAM etc.), and possibly of the connectors (e.g., in terms of bandwidth). There can also be other requirements, e.g., placing some application components on certain infrastructure nodes may be prohibited for security reasons.

There can be multiple placements of the given applications on the given infrastructure that respect all requirements. From these placements, the best one should be chosen, with respect to a given set O of objectives. Objectives may include, for example, minimizing application response time or overall energy consumption [10]. One *problem instance* of the FAPP consists of finding a solution for a given input (i.e., application and infrastructure description), which satisfies the requirements and preferably optimizes the given objectives.

There are many different versions of the FAPP, depending on the specific types of requirements and objectives considered, as well as on further assumptions about the

¹ While [18] talks about a single application, some works (e.g., [64]) address versions of the FAPP with multiple applications. We adopt here the more general assumption that A consists of a set of applications, each of which may consist of several components.

Fig. 2 Overview of the used survey methodology

infrastructure and the applications. The following are some examples for the differentiation between different FAPP variants:

- Beyond connectors between a pair of components, some authors also consider connectors between a component and an end device. Such connectors represent the fact that the component exchanges data with the end device.
- In some FAPP variants, each connector (i.e., edge in the application graph) is mapped to one link (i.e., edge in the infrastructure graph). In other formulations, each connector is mapped to one path or even a set of paths in the infrastructure graph.
- Some authors deal with the initial placement of new applications, whereas others address the re-optimization of already placed applications with the aim of reacting to changes.
- In some variants of the problem, end devices cannot host components, while in other variants this is possible.

There are also other optimization problems in fog computing that are related to the FAPP. For example, some authors investigated the offloading of specific computations from an end device to a fog node, or the distribution of load among fog nodes. Such problems are different from the FAPP, since they do not involve deciding the placement of application components.

3 Methodology

To ensure a good coverage of the relevant literature, we performed a systematic literature search [15]. Figure 2 shows an overview of the process. We started from the survey of Brogi et al. [18] and the 38 papers analyzed in that survey. We repeated the same search as Brogi et al. to also find papers that were not available at the time when that survey was written. The query taken from Brogi et al. was “fog computing $\wedge$ (service $\vee$ application) $\wedge$ (placement $\vee$ deployment)”. To ensure higher coverage, we used further queries on Google Scholar with related terms, including “fog application placement problem” and “application placement in fog computing” (both without quotation marks).

Beside keyword search, we also applied snowball search, where papers citing or cited by the already found relevant papers are analyzed as well. This way, also papers that use different terminology (edge instead of fog, service instead of application etc.) can be found. In particular, we thus found the survey [82], from which the continued snowball search led to several further related papers.

Fig. 3 Overview of the used criteria

The relevance of each found paper was assessed manually. Altogether, we identified 99 relevant papers that present algorithms for the FAPP. From the 99 papers, 98 performed a quantitative evaluation. The cutoff date for the literature search was 25 October 2020.

We categorized the papers based on the properties of the evaluation of the proposed algorithms, using the 40 criteria of Fig. 3. (We count the leaves of the tree in Fig. 3 as criteria.) The following types of criteria were used:

- General properties (7 criteria) capture the scale of the performed experiments, the environment used for the evaluation, and the baseline algorithms used for comparison.
- Assessment properties (2 criteria) relate to how the results of the experiments were processed.
- Infrastructure properties (15 criteria) relate to the infrastructure that was used (in most cases simulated) for the experiments in the given paper. This includes information about the size of the infrastructure, such as the number of fog nodes. We also looked at where the test data used in the experiments (such as the capacity

Fig. 4 Frequency of different types of used evaluation environments (shown for each year)

of nodes or the bandwidth of links) came from, to understand how realistic the used parameter values are.

- Similarly to infrastructure properties, application properties (16 criteria) relate to the size and structure of the applications used in the experiments, as well as the origin of the values for component and connector parameters.

After categorizing the papers, we analyzed the results to identify typical patterns and trends, which are described in the next sections.

4 Results

To identify trends, we could look at the yearly evolution of the number of papers with given characteristics. However, the quantities in individual years are often small, so that trends may be lost in the noise, and no statistically significant statements can be made. For example, Fig. 4 shows the frequency with which different types of evaluation environments were used per year. The frequency of papers using a “testbed” jumps back and forth (0, 0, 2, 3, 0, 2), and it is unclear how to interpret this. With such a low number of papers, random events (e.g., a conference proceedings only published in the following year) can largely distort the results. Furthermore, the interpretation of the diagram is complicated by the variance of the total number of papers per year. For example, it could be concluded that “unnamed simulators” suddenly became unpopular from 2019 to 2020, but the same seems to be true for the other types of environments. We can avoid these problems by aggregating the papers into two larger periods with roughly equal size. Hence, we split the set of 99 found papers into two buckets of similar size: older papers from 2013 to 2018, and newer papers from 2019 to 2020, as shown in Table 1.

Table 1 Overview of the found relevant papers

2013–2018	[45] [58] [74] [16] [19] [20] [105] [103] [47] [93] [98] [109] [43] [92] [9] [89] [48] [24] [26] [25] [88] [68] [91] [77] [37] [52] [80] [106] [42] [104] [40] [56] [21] [84] [12] [96] [102] [110] [8] [57] [33] [3] [87] [85] [11] [55] [69] [75] [95] [5] [112]
2019–2020	[44] [22] [59] [100] [94] [70] [76] [60] [83] [49] [50] [67] [32] [72] [64] [30] [81] [1] [66] [71] [61] [65] [7] [34] [107] [14] [28] [27] [46] [53] [108] [39] [99] [101] [2] [35] [4] [41] [78] [54] [31] [6] [111] [90] [79] [17] [36] [38]

Fig. 5 Frequency of different types of used evaluation environments

In the following, we present the main results of our analysis according to the criteria of Fig. 3. Further details, including diagrams with the yearly evolution of the numbers (both absolute and relative to the total number of papers in the given year) are available online.²

4.1 General properties

4.1.1 Evaluation environment

Figure 5 shows the frequency with which different types of evaluation environments were used. From the 99 investigated papers, 52 used an existing fog computing simulator (“named simulators”), 35 used a proprietary simulation environment (“unnamed simulators”), and 19 used a testbed, i.e., a small-scale deployment for test purposes with real hardware and software. No paper reported a real-world deployment, i.e., a production-scale deployment in a real setting, with real hardware and software. The sum of the number of papers in all categories slightly exceeds the number of papers investigated; this is because some papers used both a simulation and a testbed. From older to newer papers, there is an increase in the usage of testbeds.

Table 2 shows in more detail by how many papers the different named simulators were used. Many different simulators have been used, but only iFogSim was used by

² https://drive.google.com/uc?export=download&id=14XaGjTHR3o_LeObt-biPeCnmSEpNI-5.

Table 2 Frequency of the different types of the used named simulators

Simulator	2013–2018	2019–2020	Total
iFogSim	10	16	26
FogTorch/II	3	3	6
YAFS	1	3	4
OMNET++	2	1	3
MyiFogSim	2	0	2
SimGrid	2	0	2
ECSNeT++	0	1	1
ECSSim++	1	0	1
EdgeCloudSim	1	0	1
Extended CloudSim	1	0	1
FAAP-Simulator	0	1	1
FogAtlas	0	1	1
FogTorch	1	0	1
MobFogSim	0	1	1
ONE simulator	1	0	1

Table 3 Frequency of the different types of trivial algorithms

Algorithm	2013–2018	2019–2020	Total
Cloud-only	12	11	23
Greedy	10	12	22
Never migrate	6	3	9
Random	2	4	6
Always migrate	5	0	5
FCFS	2	1	3
Brute force	1	1	2
End-devices-only	1	0	1

several different authors. (For example, FogTorchII was used in 6 papers, but all of them were written by the same research group.)

4.1.2 Baseline algorithms

Figure 6 shows the frequency of different types of algorithms used as a baseline to which proposed algorithms were compared. 10 papers used no baseline algorithm for comparison. One paper did not present an own algorithm, but compared existing algorithms with each other. We divided the used baseline algorithms into five categories: “Trivial algorithms”, “Nontrivial algorithms”, “Algorithms from literature”, “Own algorithms” (i.e., variants of the authors’ own algorithm) and “Algorithms in real products”.

Trivial algorithms are simple heuristics that obviously do not capture the full complexity of the FAPP. This includes algorithms like Cloud-Only (place each component

Fig. 6 Frequency of different types of baseline algorithms

Table 4 Frequency of the different types of nontrivial algorithms

Algorithm	2013–2018	2019–2020	Total
Optimal	14	10	24
Genetic	3	3	6
Backtrack	1	1	2
Machine learning	1	1	2
Breadth first search	0	1	1
Particle swarm optimization	0	1	1
Symbiotic organisms search	1	0	1

in the cloud), Greedy (place each component in the first node where it fits), and Random (place each component on a random node). Such algorithms were used by 50 papers as comparison baseline. As can be seen in Table 3, Cloud-Only and Greedy algorithms were used the most. The sum of the trivial algorithms from Table 3 does not equal the number of papers that used trivial algorithms, since some papers used more than one of the trivial algorithms from Table 3 as a baseline algorithm.

Nontrivial algorithms are algorithms that are more sophisticated and lead to good results for the FAPP, but were still mainly created to serve as comparison baseline, and not as competitive algorithms for the FAPP themselves. This includes algorithms such as Optimal (computing an optimal placement with, e.g., integer programming) and Genetic (using a genetic algorithm). This category of algorithms was used by 29 papers. As can be seen in Table 4, Optimal was the most frequently used nontrivial algorithm. Again, the sum of the nontrivial algorithms from Table 4 does not equal the number of papers that used nontrivial algorithms because some papers used more than one of the nontrivial algorithms from Table 4 as a baseline algorithm.

35 papers used algorithms from the literature as comparison baseline. Comparing older and newer papers, there is a clear increase in the usage of algorithms from the

Fig. 7 Histogram of the number of used problem instances

Fig. 8 Histogram of the number of runs per problem instance

literature as baseline. Papers [45,93] stand out, as 11 and 6 papers, respectively, used their algorithms as comparison baseline.

30 papers used own algorithms as baseline algorithms. These are usually weaker versions of the proposed algorithm that, for example, do not take some aspects into account that the proposed algorithm does.

Only 3 papers used algorithms in real products as comparison baseline. These algorithms were Storm's EventScheduler and Kubernetes' scheduler.

4.1.3 Scale of the performed experiments

Figure 7 shows a histogram of the number of problem instances used in the experiments. We divided the set of positive integers into the intervals $[1, 20]$, $[21, 40]$, and $[41, \infty)$. For each interval, the figure shows how often the number of used problem instances falls into the given interval. If a paper reports multiple experiments, we summed the number of problem instances of the experiments. The majority of papers (63) used between 1 and 20 problem instances. The number of problem instances was in the interval $[21, 40]$ and $[41, \infty)$ only 19 and 16 times, respectively.

Figure 8 shows a histogram of the number of runs per problem instance, i.e., the number of times an experiment was repeated on the same problem instance. Here, we divided the set of positive integers into the intervals $\{1\}$, $[2, 10]$, $[11, 100]$, and

Fig. 9 Frequency of the different evaluation metrics

[101, ∞). For each interval, the figure shows how often the number of runs per problem instance falls into the given interval. If a paper reported multiple different experiments, we take the maximum number of runs per problem instance in any experiment. In roughly half of the papers (49 out of 99), only one run per problem instance was performed. 10 papers performed 2–10, 19 papers performed 11–100, and 6 performed more than 100 runs per problem instance. From the 6 papers with more than 100 runs per problem instance, 5 performed even more than 1000 runs per problem instance. It should be noted that 14 papers did not clearly indicate the number of runs per problem instance, so we could not assign them to any interval.

4.2 Assessment properties

4.2.1 Evaluation metrics

Figure 9 shows which metrics were used and how often to measure the quality of fog application placement algorithms. As can be seen, latency was the most widely used metric. A more detailed analysis shows that the term “latency” was used with different meanings (in line with previous findings [10]). In particular, 52 papers reported the end-to-end latency of an application (or the total for a set of applications), consisting of the time for both computation and data transfer. 19 papers measured not-end-to-end latency, i.e., either only network latency or only computation time. Of the 52 papers that considered end-to-end latency, 4 also considered network latency and computation time separately, so in the end 23 papers considered not-end-to-end latency. Some papers, such as [105], combined different (partial) latencies into one metric using their weighted sum.

Beside latency, also resource utilization, network usage, and QoS violations were often used as evaluation metrics. Monetary costs and power consumption had the greatest increase over time. Only 12 papers consider the number of migrations as a metric. (It should be noted that only 32 papers use migrations.)

Most papers used characteristics of the resulting placement (e.g., latency) as evaluation metrics. Only 38 of the 99 papers considered the execution time of the placement

Fig. 10 Frequency of the different statistical assessment methods

algorithm. From those 38, 30 also considered scalability, i.e., how the execution time of the algorithm grows with increasing problem size.

4.2.2 Statistical assessment

Roughly half of the investigated papers (49 out of 99) used some kind of statistical assessment of the results of their experiments. As Fig. 10 shows, by far the most popular statistical assessment method (used in 33 papers) is computing the average of the results from multiple runs. Also, several other methods were used, although by a minority of papers, to give more details about the distribution of the results. Boxplots and confidence intervals were used by 9 papers each, with a clearly increasing trend. A CDF (cumulative distribution function) was used by 6 papers. Statistical hypothesis testing was used by only 1 paper (it was a Wilcoxon signed rank test).

4.3 Application properties

4.3.1 Origin of the application structure

Figure 11 shows the frequency of different origins of the application structure. By origin of the application structure, we mean the source from which the structure of the used test application(s) stems. In 64 of the 99 papers, applications were created by the authors themselves, without providing further details of possible origins (“Self chosen: Not specified”). The ratio of such papers decreased significantly from older to newer papers. 26 papers took application structures from the literature (“Literature”). Of these 26 papers, the most used application structures were from [45] (9 times) and [86] (4 times). The number of papers that took application structures from the literature increased in the newer papers. 17 papers randomly generated the application structure (“Self chosen: Generated”). Different approaches were used for this purpose; e.g., some papers used a “growing network” structure, in which each new component is connected to a randomly chosen one of the previous components. Several papers only mentioned that the application structure was randomly generated, without specifying

Fig. 11 Frequency of different origins of the application structure

Fig. 12 Frequency of different origins of application parameter values

the exact procedure. One paper used a dataset from reality to create an application structure (“Reality”). It should be noted that several authors used multiple applications, sometimes from different origins. For this reason, the same paper may be found in more than one category, so that the sum of the number of papers from the categories exceeds the total number of papers.

4.3.2 Origin of application parameter values

Figure 12 shows how often the parameter values of the applications (such as components’ CPU requirements and connectors’ bandwidth requirements) are self chosen without further explanation (“Self chosen: Not specified”), randomly generated based on self-chosen intervals (“Self chosen: Generated”), taken from a real system (“Reality”) or from the literature (“Literature”). 49 papers used self-chosen values without any justification. 28 papers generated parameter values randomly, based on self chosen intervals. 14 papers took the values from the literature, with [45] being the most frequently used source (used 4 times). The remaining 10 of these 14 papers took the values from different sources, with 4 of them generating values based on the values they took from the literature. Only 9 papers took values from reality. Of these 9 papers, 5 used default values for VM instances provided by OpenStack.³ 13 papers did not provide any information about the values used and their origin. In the newer papers,

³ <https://www.openstack.org/>.

Fig. 13 Histogram of the number of components in the test applications

using the literature as the source for application parameters greatly increased, while the number of papers using self chosen and not further specified values decreased. Some papers used multiple sources for application parameters; thus, the sum of the number of papers from all categories exceeds the total number of papers.

4.3.3 Number of components

Figure 13 shows a histogram of the number of components used in the experiments. If a paper used multiple applications with different numbers of components, then we chose the maximum of those numbers. The number of components is between 1 and 10 for 27 papers, between 11 and 100 for 35 papers, and above 100 for 29 papers. In 7 papers, the number of components was not clear from the paper.

4.4 Infrastructure properties

4.4.1 Origin of the infrastructure topology

Figure 14 shows the frequency of different origins of the infrastructure topology. Again, some authors performed experiments using infrastructure topologies from different origins; thus, the sum of all categories is higher than the number of papers investigated. The infrastructure topologies of 73 papers were self chosen with no further justification (“Self chosen: Not specified”). Infrastructure topologies were randomly generated in 27 papers, based on self-chosen data (“Self chosen: Generated”). In these 27 papers, an Erdős-Rényi graph model was used in 3 papers, a random Barabasi-Albert network in 3 papers, and a random Euclidean graph and a random Waxman graph in one paper each, to generate the topologies. The remaining 19 papers provided no details on the random generation of infrastructure topologies. 2 papers used graphs from reality, namely from the Internet Topology Zoo⁴ (“Reality”). Only one paper used infrastructure topologies from the literature, namely from [45,93] (“Literature”).

⁴ <http://www.topology-zoo.org/>.

Fig. 14 Frequency of different origins of the infrastructure topology

Fig. 15 Frequency of different origins of infrastructure parameter values

4.4.2 Origin of the infrastructure parameter values

Figure 15 shows the frequency of different origins for the infrastructure parameters (such as nodes' RAM capacity and links' bandwidth). The same categories are used as previously for the origin of application parameters. Again, some papers used different origins for different infrastructure parameters; thus, the sum of the number of papers from all categories exceeds the total number of papers. As with the application parameters, also here most papers, in this case 61, used self-chosen values without further justification ("Self chosen: Not specified"). In 29 papers, the authors generated the infrastructure parameters ("Self chosen: Generated"). Unlike for application parameters, values for the infrastructure parameters were taken from reality quite frequently, namely in 30 papers ("Reality"). 16 papers used values from the literature ("Literature") and 6 papers did not provide any information about the values used and their origin ("No information"). Comparing old and new papers, the literature as source for

Fig. 16 Histogram of the number of cloud data centers

Fig. 17 Histogram of the number of fog nodes

infrastructure parameter values has become more frequent (as for application parameter values), with [45] as the most frequent source (used 4 times).

4.4.3 Number of nodes

We evaluated the number of cloud data centers, fog nodes, and end devices in the used test infrastructures. When authors performed multiple experiments with different number of nodes, we selected the highest number from the experiments. To create histograms, we formed intervals for each type of nodes. For end devices and fog nodes, these are $\{0\}$, $[1, 10]$, $[11, 100]$, and $[101, \infty)$. Since in many cases only one cloud was used, we limited the intervals for cloud data centers to $\{0\}$, $\{1\}$, and $[2, \infty)$.

Figure 16 shows the histogram of the number of cloud data centers. 28 papers did not use cloud data centers in their experiments. About half of the papers (49 out of 99) used only one cloud data center, while 19 papers used more than 1. In 2 papers, it was not clear how many cloud data centers were used.

Figure 17 shows the histogram of the number of fog nodes. It can be seen that 2 papers did not use fog nodes. (These papers used end devices, on which the deployment of components is allowed.) Most papers used between 1 and 100 fog nodes in their experiments, while 19 papers used more than 100 fog nodes. Of these 19 papers, 5

Fig. 18 Histogram of the number of end devices

used more than 1000 fog nodes. 4 papers did not specify how many fog nodes they used.

As can be seen in Fig. 18, roughly one-third of the papers (37 out of 99) did not use any end devices in their experiments. Only 13 papers used a number of end devices between 1 and 10, while 22 papers used a number of end devices between 11 and 100. Of the 99 papers, 18 used more than 100 end devices and of these, 6 papers used more than 1000. 8 papers did not provide any information about the number of end devices used. A total of 38 papers used end devices on which deployment of components was allowed.

5 Discussion

In this section, we analyze the results to identify good and bad practices and trends. We focus on two key aspects of the validity of empirical studies: internal validity and external validity [23]. Internal validity means that the experiments allow sound and meaningful conclusions to be drawn for the cases covered by the experiments. External validity means that the conclusions can be generalized to other situations that were not tested explicitly.

5.1 Internal validity of empirical evaluations

Reviewing the results of the previous section reveals some threats to the internal validity of empirical studies on fog application placement algorithms.

5.1.1 Baseline algorithms

As shown in Sect. 4.1.2, many papers use trivial algorithms or weakened versions of their own algorithms as comparison baseline, or no comparison with other algorithms at all. Such an experiment design does not allow to assess whether the proposed algorithm advances the state of the art. However, Sect. 4.1.2 also revealed a promising trend: the number of papers using baseline algorithms from the literature is increasing. The community needs to reinforce this trend. Newly proposed algorithms should be

compared to current high-performing algorithms to show that they improve the state of the art. This also implies that algorithm implementations should be made publicly available so that other researchers can use them for comparison.

As mentioned in Sect. 4.1.2, algorithms from [45,93] were often used as comparison baseline. This does not mean, however, that these algorithms should be also used for comparison in the future. These algorithms were published in 2017. Since then, numerous algorithms have been shown to outperform them. Thus, they do not represent the state of the art anymore.

5.1.2 Number of runs and statistical assessment

For randomized algorithms, a sufficient number of runs should be performed on each problem instance to minimize the effect of chance. Even for deterministic algorithms, randomization and performing multiple runs may offer more insights into the algorithm's performance. As Sect. 4.1.3 reveals, the ratio of papers with only a single run per problem instance is decreasing, which is commendable.

Results from several runs can be assessed statistically. As Sect. 4.2.2 showed, the majority of papers with multiple runs per problem instance only compute the average of the results. Few papers use more advanced statistical methods, but the ratio of such papers is increasing. The question whether the experimental results support that one algorithm performs better than another could be rigorously answered by statistical hypothesis testing. The field of statistics offers many useful methods for such purposes which are now largely unused.

5.1.3 Algorithm execution time

Section 4.2.1 shows that less than 40% of the investigated papers assess the execution time of their algorithms. However, the execution time of fog application placement algorithms is critical. In most FAPP formulations, the set of all possible placements of an application is finite, so the best placement can be found by checking each possible placement, and taking a placement that satisfies the requirements and is best in terms of the objectives. The only reason why such an algorithm is not feasible is its exponential execution time. Thus, the evaluation of an algorithm should include its execution time.

5.1.4 Reproducibility

As we have seen throughout Sect. 4, some papers could not be categorized in some of the investigated dimensions, because the relevant information was not clear from those papers. Authors should describe their experimental setup in such detail and clarity that others can reproduce the experiments. This helps to minimize the likelihood of errors, and if errors are made, they are easier to spot (e.g., by reviewers) and can be corrected.

5.2 External validity of empirical evaluations

5.2.1 Evaluation environments

A production-scale real-world deployment would be the ideal evaluation environment. However, as Sect. 4.1.1 shows, all papers use a simulation or testbed instead. This is understandable, since the use of a production-scale real-world deployment incurs huge costs. Also, simulations have the advantage of easy experimentation with many different settings. Not using a production-scale real-world deployment is not necessarily a problem, if the used environment behaves similarly to real production environments. It is important to validate that the used environment behaves similarly to real production environments. Unfortunately, this rarely happens. Using a non-realistic evaluation environment can heavily degrade the transferability of the results to practical settings. The community should work on the creation and thorough validation of evaluation environments for fog application placement algorithms, and then consistently use those environments [51].

5.2.2 Problem instances

The experimental evaluation of algorithms is always limited to a finite set of problem instances, and we cannot be sure whether the results can be generalized to other problem instances. Nevertheless, authors should aim at using a representative set of problem instances, to foster generalization as much as possible. This requires a substantial number of problem instances, but, as Sect. 4.1.3 showed, most papers actually use very few. The ratio of papers using at most 20 problem instances is decreasing, while the ratio of papers using more than 40 is increasing, but the trend is slow.

The next question is how realistic the used problem instances are. As an indicator for this, we investigated in Sects. 4.3.1–4.4.2 the sources for the structure and the parameter values for the applications and infrastructures. In all four cases we found that most papers use self-chosen structures and parameter values without any justification, which raises the question of how realistic those structures and values are. Concerning the applications, the ratio of papers with self-chosen structures and parameter values is decreasing, while the ratio of papers taking the structures and the parameter values from the literature is increasing, which seems to be a positive trend. However, the number of papers using structures and parameter values from real applications is very low. Infrastructure parameter values are taken much more frequently from real systems. On the other hand, the infrastructure topology is rarely taken from real systems, and also reusing topologies from the literature is rare.

The third question regarding problem instances is how big they are. As Sects. 4.3.3 and 4.4.3 show, the majority of papers use less than 100 components, less than 100 fog nodes, and less than 100 end devices. While these problem instance sizes might well represent small-scale fog deployments, they do not seem to be sufficient to test how the algorithms scale to large fog deployments. Several authors motivate fog computing by quoting studies predicting billions of end devices, but no paper actually performs experiments on this scale.

5.2.3 Literature

The reuse of infrastructure and application specifications from the literature is increasing, which is a positive trend. However, it also matters which previous papers are used as sources. The main aim should be to reuse test data that is realistic. Several authors using the iFogSim simulator reuse test data from [45]. It is not clear if test data from [45] is reused because it is considered realistic, or only because it is readily available in iFogSim.

5.3 Positive examples

In the following, we give more details of selected papers exhibiting some good practices. For example, Gavaber et al. [39] carried out a simulation in iFogSim, with two experiments using almost all the test data and the baseline algorithms from [45,93]. For the statistical evaluation, the authors performed a Wilcoxon signed-rank test. *Good practice*: use of statistical hypothesis testing.

Xia et al. [105] carried out a simulation in SimGrid. They developed a scaling approach for the test infrastructure and the test application, which consists of three phases and generates problem instances with up to approximately 10,000 fog nodes and 4000 components. They considered 10 problem instances and performed 10 runs per problem instance, randomly generating the parameter values of the infrastructure. In the statistical evaluation, not only the average, but also the variance was considered. Xia et al. also investigated how long their algorithms take on such large problem instances. *Good practice*: use of large-scale test infrastructures and large test applications.

Djemai et al. [30] carried out a simulation in iFogSim. The parameter values for the infrastructure were taken from existing hardware (for example, the hardware configuration of a Samsung smartphone) and various access network technologies, such as LTE and WiFi. The application structure and the parameter values of the application were taken from the literature. A total of 8 problem instances were examined. 50 runs were performed per problem instance and then the average was calculated. The presented algorithm was compared to the algorithm from [93]. The execution time required by the algorithms to find a feasible solution was also investigated. *Good practice*: use of parameter values from real systems.

5.4 Summary of recommendations

We now summarize our recommendations for the experimental evaluation of fog application placement algorithms:

- Use an evaluation environment that has been shown to deliver results in tolerable error bounds compared with the targeted production system.
- Use a sufficient number of different problem instances. The problem instances used for the evaluation should be representative of the types of problem instances that could arise in the targeted production system. This will usually require at least some dozens of problem instances.

- Determine the structure and parameter values for the infrastructure and the applications from realistic sources (from real systems or the literature).
- Ensure that the size of the test problem instances is in a similar range as the size of the target systems.
- Use state-of-the-art algorithms as comparison baseline.
- Perform a sufficient number of runs per problem instance to minimize the effects of chance. The necessary number of runs can be estimated using statistical methods from the targeted confidence level and precision [73]. As a rule of thumb, if performing the experiments multiple times leads to significantly different distributions, then the number of runs is not enough.
- Use solid statistical methods, like hypothesis testing, to assess the results.

To foster the meaningful evaluation of new fog application placement algorithms, the community should address some general challenges:

- Creation and detailed experimental validation of appropriate evaluation environments (simulators, testbeds).
- Definition of standard variants of the FAPP that allow the direct comparison of different algorithms solving the same problem variant.
- Creating a repository of publicly available and readily applicable fog application placement algorithm implementations.
- Definition of standard benchmark problem instances or generators for such problem instances.

6 Conclusions

We have presented the results of a literature survey on fog application placement algorithms. Our focus was on how newly proposed algorithms are evaluated. We investigated 99 papers and categorized them based on a taxonomy of 40 categories, from which we presented here the most interesting results.

The results of our analysis show that the validity of the findings from existing empirical evaluations is often threatened. Typical shortcomings of such evaluations include the lack of comparison to state-of-the-art algorithms, lack of statistical assessment, unrealistic evaluation environments, too few or too small problem instances, and unjustified test data. Although some of these aspects seem to improve over time, there is still much to do to further improve the validity of such evaluations. We presented several recommendations on how to avoid the identified pitfalls.

Future work should focus on creating sound groundwork for the empirical evaluation of fog application placement algorithms. This includes the creation and validation of realistic evaluation environments (such as simulators), establishing standard variants of the fog application placement problem along with (generators of) corresponding benchmark problem instances, and creating a repository of interoperable algorithm implementations. Organizing regular competitions for FAPP algorithms, such as the ones for satisfiability (<http://www.satcompetition.org/>) or software verification (<https://sv-comp.sosy-lab.org/>), could also be beneficial.

Acknowledgements This work was partially supported by the European Union's Horizon 2020 research and innovation programme under Grant 871525 (FogProtect).

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Afrasiabi SN, Kianpisheh S, Mouradian C, Glitho RH, Moghe A (2019) Application components migration in NFV-based hybrid cloud/fog systems. In: IEEE international symposium on local and metropolitan area networks (LANMAN), pp 1–6
2. Al-Tarawneh MA (2020) Mobility-aware container migration in cloudlet-enabled IoT systems using integrated multicriteria decision making. *Environments* 11(9):1–8
3. Amarasinghe G, de Assunção MD, Harwood A, Karunasekera S (2018) A data stream processing optimisation framework for edge computing applications. In: IEEE 21st international symposium on real-time distributed computing (ISORC), pp 91–98
4. Amarasinghe G, de Assunção MD, Harwood A, Karunasekera S (2020) ECSNeT++: a simulator for distributed stream processing on edge and cloud environments. *Future Gener Comput Syst* 111:401–418
5. Aryal RG, Altmann J (2018) Dynamic application deployment in federations of clouds and edge resources using a multiobjective optimization AI algorithm. In: 3rd international conference on fog and mobile edge computing (FMEC), pp 147–154. IEEE
6. Azizi S, Khosroabadi F (2019) A QoS-aware service placement algorithm for fog-cloud computing environments. In: 4th international conference neural sciences (ICNS). Mathematics and Computer Science
7. Azizi S, Khosroabadi F, Shojafar M (2019) A priority-based service placement policy for fog-cloud computing systems. *Comput Methods Differ Equ* 7(4 (Special Issue)):521–534
8. Bahreini T, Grosu D (2017) Efficient placement of multi-component applications in edge computing systems. In: 2nd ACM/IEEE symposium on edge computing, pp 1–11
9. Barcelo M, Correa A, Llorca J, Tulino AM, Vicario JL, Morell A (2016) IoT-cloud service optimization in next generation smart environments. *IEEE JSAC* 34(12):4077–4090
10. Bellendorf J, Mann ZÁ (2020) Classification of optimization problems in fog computing. *Future Gener Comput Syst* 107:158–176
11. Benamer AR, Teyeb H, Hadj-Alouane NB (2018) Latency-aware placement heuristic in fog computing environment. In: OTM confederated international conference on “On the Move to Meaningful Internet Systems”, pp 241–257. Springer
12. Bittencourt LF, Diaz-Montes J, Buyya R, Rana OF, Parashar M (2017) Mobility-aware application scheduling in fog computing. *IEEE Cloud Comput* 4(2):26–35
13. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the Internet of Things. In: 1st MCC workshop on mobile cloud computing, pp 13–16
14. Bourhim EH, Elbiaze H, Dieye M (2019) Inter-container communication aware container placement in fog computing. In: 15th international conference on network and service management (CNSM), pp 1–6. IEEE
15. Brereton P, Kitchenham BA, Budgen D, Turner M, Khalil M (2007) Lessons from applying the systematic literature review process within the software engineering domain. *J Syst Softw* 80(4):571–583

16. Brogi A, Forti S (2017) QoS-aware deployment of IoT applications through the fog. *IEEE Internet Things J* 4(5):1185–1192
17. Brogi A, Forti S, Guerrero C, Lera I (2019) Meet genetic algorithms in Monte Carlo: Optimised placement of multi-service applications in the fog. In: *IEEE international conference on edge computing (EDGE)*, pp 13–17
18. Brogi A, Forti S, Guerrero C, Lera I (2020) How to place your apps in the fog: state of the art and open challenges. *Softw Pract Exp* 50(5):719–740
19. Brogi A, Forti S, Ibrahim A (2017) How to best deploy your fog applications, probably. In: *IEEE 1st international conference fog and edge computing (ICFEC)*, pp 105–114
20. Brogi A, Forti S, Ibrahim A (2018) Deploying fog applications: how much does it cost, by the way? In: *CLOSER*, pp 68–77
21. Brogi A, Forti S, Ibrahim A (2018) Optimising QoS-assurance, resource usage and cost of fog application deployments. In: *International conference on cloud computing and services science*, pp 168–189. Springer
22. Brogi A, Forti S, Ibrahim A (2019) Predictive analysis to support fog application deployment. *Fog and edge computing: principles and paradigms*, pp 191–222
23. Campbell DT, Stanley JC (2015) *Experimental and quasi-experimental designs for research*. Ravenio Books
24. Cardellini V, Grassi V, Lo Presti F, Nardelli M (2015) Distributed QoS-aware scheduling in storm. In: *9th ACM international conference on distributed event-based system*, pp 344–347
25. Cardellini V, Grassi V, Lo Presti F, Nardelli M (2016) Optimal operator placement for distributed stream processing applications. In: *10th ACM international conference on distributed and event-based systems*, pp 69–80
26. Cardellini V, Grassi V, Presti FL, Nardelli M (2015) On QoS-aware scheduling of data stream applications over fog computing infrastructures. In: *IEEE symposium on computers and communication (ISCC)*, pp 271–276
27. Charântola D, Mestre AC, Zane R, Bittencourt LF (2019) Component-based scheduling for fog computing. In: *12th IEEE/ACM international conference on utility and cloud computing companion*, pp 3–8
28. Chiti F, Fantacci R, Paganelli F, Picano B (2019) Virtual functions placement with time constraints in fog computing: a matching theory perspective. *IEEE Trans Netw Serv Manag* 16(3):980–989
29. Dastjerdi AV, Buyya R (2016) Fog computing: helping the Internet of Things realize its potential. *Computer* 49(8):112–116
30. Djemai T, Stolf P, Monteil T, Pierson JM (2019) A discrete particle swarm optimization approach for energy-efficient IoT services placement over fog infrastructures. In: *18th international symposium on parallel and distributed computing (ISPD)*, pp 32–40. IEEE
31. Donassolo B, Fajjari I, Legrand A, Mertikopoulos P (2019) Fog based framework for IoT service provisioning. In: *16th IEEE annual consumer communications and networking conference (CCNC)*, pp 1–6
32. Eidenbenz R, Pignolet YA, Ryser A (2020) Latency-aware industrial fog application orchestration with Kubernetes. In: *5th international conference on fog and mobile edge computing (FMEC)*, pp 164–171. IEEE
33. Elgamal T, Sandur A, Nguyen P, Nahrstedt K, Agha G (2018) DROPLET: Distributed operator placement for IoT applications spanning edge and cloud resources. In: *IEEE 11th international conference on cloud computing (CLOUD)*, pp 1–8
34. Faticanti F, De Pellegrini F, Siracusa D, Santoro D, Cretti S (2019) Cutting throughput with the edge: app-aware placement in fog computing. In: *6th IEEE international conference on cyber security and cloud computing (CSCloud)/5th IEEE international conference on edge computing and scalable cloud (EdgeCom)*, pp 196–203
35. Faticanti F, De Pellegrini F, Siracusa D, Santoro D, Cretti S (2020) Throughput-aware partitioning and placement of applications in fog computing. *IEEE Trans Netw Serv Manage* 17:2436–2450
36. Faticanti F, Savi M, Pellegrini FD, Kochovski P, Stankovski V, Siracusa D (2020) Deployment of application microservices in multi-domain federated fog environments. In: *International conference on omni-layer intelligent systems (COINS)*, pp 1–6
37. Filiposka S, Mishev A, Gilly K (2018) Community-based allocation and migration strategies for fog computing. In: *IEEE wireless communications and network conference (WCNC)*

38. Forti S, Ferrari GL, Brogi A (2020) Secure cloud-edge deployments, with trust. *Future Gener Comput Syst* 102:775–788
39. Gavaber MD, Rajabzadeh A (2020) MFP: An approach to delay and energy-efficient module placement in IoT applications based on multi-fog. *J Ambient Intell Hum Comput* 12:7965–7981
40. Gedeon J, Stein M, Wang L, Muehlhaeuser M (2018) On scalable in-network operator placement for edge computing. In: 27th international conference on computer communication and networks (ICCCN), pp 1–9. IEEE
41. Godinho N, Curado M, Paquete L (2019) Optimization of service placement with fairness. In: IEEE symposium computers and communications (ISCC), pp 1–6
42. Gonçalves D, Velasquez K, Curado M, Bittencourt L, Madeira E (2018) Proactive virtual machine migration in fog environments. In: IEEE symposium on computers and communications (ISCC), pp 00742–00745
43. Gu L, Zeng D, Guo S, Barnawi A, Xiang Y (2015) Cost efficient resource management in fog computing supported medical cyber-physical system. *IEEE Trans Emerg Top Comput* 5(1):108–119
44. Guerrero C, Lera I, Juiz C (2019) A lightweight decentralized service placement policy for performance optimization in fog computing. *J Ambient Intell Humaniz Comput* 10(6):2435–2452
45. Gupta H, Dastjerdi AV, Ghosh SK, Buyya R (2017) iFogSim: a toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments. *Softw Pract Exp* 47(9):1275–1296
46. Hiessl T, Karagiannis V, Hochreiner C, Schulte S, Nardelli M (2019) Optimal placement of stream processing operators in the fog. In: IEEE 3rd international conference on fog and edge computing (ICFEC), pp 1–10
47. Hong HJ, Tsai PH, Hsu CH (2016) Dynamic module deployment in a fog computing platform. In: 18th Asia-Pacific networking operations and management symposium (APNOMS)
48. Huang Z, Lin KJ, Yu SY, Hsu JY (2014) Co-locating services in IoT systems to minimize the communication energy cost. *J Innov Digit Ecosyst* 1(1–2):47–57
49. Kayal P, Liebeherr J (2019) Autonomic service placement in fog computing. In: IEEE 20th international symposium on world of wireless, mobile and multimedia networks (WoWMoM)
50. Kayal P, Liebeherr J (2019) Distributed service placement in fog computing: an iterative combinatorial auction approach. In: IEEE 39th international conference on distributed computing systems (ICDCS), pp 2145–2156
51. Kunde C, Mann ZÁ (2020) Comparison of simulators for fog computing. In: Proceedings of the 35th annual ACM symposium on applied computing, pp 1792–1795
52. Lera I, Guerrero C, Juiz C (2018) Availability-aware service placement policy in fog computing based on graph partitions. *IEEE Int Things J* 6(2):3641–3651
53. Lera I, Guerrero C, Juiz C (2019) Analyzing the applicability of a multi-criteria decision method in fog computing placement problem. In: 4th international conference on fog and mobile edge computing (FMED), pp 13–20. IEEE
54. Lera I, Guerrero C, Juiz C (2019) YAFS: a simulator for IoT scenarios in fog computing. *IEEE Access* 7:91745–91758
55. Li K, Nabrzyski J (2017) Traffic-aware virtual machine placement in cloudlet mesh with adaptive bandwidth. In: IEEE international conference on cloud computing technology and science (Cloud-Com), pp 49–56
56. Lopes MM, Higashino WA, Capretz MA, Bittencourt LF (2017) MyiFogSim: a simulator for virtual machine migration in fog computing. In: 10th international conference on utility and cloud computing companion, pp 47–52
57. Loukopoulous T, Tziritas N, Koziri M, Stamoulis G, Khan SU, Xu CZ, Zomaya AY (2018) Data stream processing at network edges. In: IEEE international on parallel and distributed processing symposium workshops (IPDPSW), pp 657–665
58. Mahmud R, Ramamohanarao K, Buyya R (2018) Latency-aware application module management for fog computing environments. *ACM Trans Int Technol* 19(1):1–21
59. Mahmud R, Srirama SN, Ramamohanarao K, Buyya R (2019) Quality of Experience (QoE)-aware placement of applications in fog computing environments. *J Parallel Distrib Comput* 132:190–203
60. Mahmud R, Toosi AN, Rao K, Buyya R (2020) Context-aware placement of industry 4.0 applications in fog computing environments. *IEEE Trans Ind Inform* 16(11):7004–7013

61. Maia AM, Ghamri-Doudane Y, Vieira D, de Castro MF (2019) Optimized placement of scalable IoT services in edge computing. In: IFIP/IEEE symposium on integrated network and service management (IM), pp 189–197
62. Mann ZÁ (2019) Optimization problems in fog and edge computing. In: Fog and edge computing: principles and paradigms, pp.103–121. Wiley
63. Mann ZÁ (2021) Notions of architecture in fog computing. *Computing* 103(1):51–73
64. Mann ZÁ, Metzger A, Prade J, Seidl R (2019) Optimized application deployment in the fog. In: International conference on service-oriented computing, pp 283–298. Springer
65. Martin JP, Kandasamy A, Chandrasekaran K (2020) CREW: Cost and reliability aware eagle-whale optimiser for service placement in fog. *Pract Exp Softw* 50:2337–2360
66. Martin JP, Kandasamy A, Chandrasekaran K (2020) Mobility aware autonomic approach for the migration of application modules in fog computing environment. *J Ambient Intell Hum Comput* 1–20
67. Mehran N, Kimovski D, Prodan R (2019) MAPO: a multi-objective model for IoT application placement in a fog environment. In: 9th international conference on Internet of Things
68. Mennes R, Spinnewyn B, Latré S, Botero JF (2016) GRECO: A distributed genetic algorithm for reliable application placement in hybrid clouds. In: 5th IEEE international conference on cloud networking (Cloudnet), pp 14–20
69. Minh QT, Nguyen DT, Van Le A, Nguyen HD, Truong A (2017) Toward service placement on fog computing landscape. In: 4th NAFOSTED conference on information and computer science, pp 291–296. IEEE
70. Mouradian C, Kianpisheh S, Abu-Lebdeh M, Ebrahimnezhad F, Jahromi NT, Glitho RH (2019) Application component placement in NFV-based hybrid cloud/fog systems with mobile fog nodes. *IEEE JSAC* 37(5):1130–1143
71. Nardelli M, Cardellini V, Grassi V, Presti FL (2019) Efficient operator placement for distributed data stream processing applications. *IEEE Trans Parallel Distrib Syst* 30(8):1753–1767
72. Nashaat H, Ahmed E, Rizk R (2020) IoT application placement algorithm based on multi-dimensional QoE prioritization model in fog computing environment. *IEEE Access* 8:111253–111264
73. NIST (2012) NIST/SEMATECH e-handbook of statistical methods. <http://www.itl.nist.gov/div898/handbook>
74. Ottenwälder B, Koldehofe B, Rothermel K, Ramachandran U (2013) MigCEP: Operator migration for mobility driven distributed complex event processing. In: 7th ACM international conference on distributed event-based systems, pp 183–194
75. Ouyang T, Zhou Z, Chen X (2018) Follow me at the edge: mobility-aware dynamic service placement for mobile edge computing. *IEEE JSAC* 36(10):2333–2345
76. Pallewatta S, Kostakos V, Buyya R (2019) Microservices-based IoT application placement within heterogeneous and resource constrained fog computing environments. In: 12th IEEE/ACM international conference on utility and cloud computing, pp 71–81
77. Rahbari D, Nickray M (2017) Scheduling of fog networks with optimized knapsack by symbiotic organisms search. In: 21st conference of open innovations association (FRUCT), pp 278–283. IEEE
78. Rahbari D, Nickray M (2019) Low-latency and energy-efficient scheduling in fog-based IoT applications. *Turk J Electr Eng Comput Sci* 27(2):1406–1427
79. Renart EG, Veith ADS, Balouek-Thomert D, De Assuncao MD, Lefevre L, Parashar M (2019) Distributed operator placement for IoT data analytics across edge and cloud resources. In: 19th IEEE/ACM international symposium on cluster, cloud, and grid computing (CCGrid), pp 1–10
80. Rezazadeh Z, Rahbari D, Nickray M (2018) Optimized module placement in IoT applications based on fog computing. In: Iranian conference on electrical engineering (ICEE), pp 1553–1558. IEEE
81. Rezazadeh Z, Rezaei M, Nickray M (2019) LAMP: A hybrid fog-cloud latency-aware module placement algorithm for IoT applications. In: 5th conference on knowledge based engineering and innovation (KBEI), pp 845–850. IEEE
82. Salaht FA, Desprez F, Lebre A (2020) An overview of service placement problem in fog and edge computing. *ACM Comput Surv* 53(3):art. 65
83. Salaht FA, Desprez F, Lebre A, Prud'Homme C, Abderrahim M (2019) Service placement in fog computing using constraint programming. In: IEEE international conference on services computing (SCC), pp 19–27

84. Saurez E, Hong K, Lillethun D, Ramachandran U, Ottenwälder B (2016) Incremental deployment and migration of geo-distributed situation awareness applications in the fog. In: 10th international conference on distributed and event-based systems, pp 258–269
85. Scoca V, Aral A, Brandic I, De Nicola R, Uriarte RB (2018) Scheduling latency-sensitive applications in edge computing. In: CLOSER, pp 158–168
86. Shukla A, Chaturvedi S, Simmhan Y (2017) Riotbench: A real-time IoT benchmark for distributed stream processing platforms. arXiv preprint [arXiv:1701.08530](https://arxiv.org/abs/1701.08530)
87. da Silva Veith A, de Assuncao MD, Lefevre L (2018) Latency-aware placement of data stream analytics on edge computing. In: International conference on service-oriented computing, pp 215–229. Springer
88. Skarlat O, Nardelli M, Schulte S, Borkowski M, Leitner P (2017) Optimized IoT service placement in the fog. *Serv Orient Comput Appl* 11(4):427–443
89. Skarlat O, Nardelli M, Schulte S, Dustdar S (2017) Towards QoS-aware fog service placement. In: IEEE 1st international conference on fog and edge computing (ICFEC), pp 89–96
90. de Souza FR, de Assuncao MD, Caron E, da Silva Veith A (2020) An optimal model for optimizing the placement and parallelism of data stream processing applications on cloud-edge computing. In: IEEE 32nd international symposium on computer architecture and high performance computing (SBAC-PAD), pp 59–66
91. Souza VB, Masip-Bruin X, Marín-Tordera E, Ramírez W, Sanchez S (2016) Towards distributed service allocation in fog-to-cloud (f2c) scenarios. In: IEEE global communications conference (GLOBECOM), pp 1–6
92. Souza VBC, Ramírez W, Masip-Bruin X, Marín-Tordera E, Ren G, Tashakor G (2016) Handling service allocation in combined fog-cloud scenarios. In: IEEE international conference on communications (ICC), pp 1–5
93. Taneja M, Davy A (2017) Resource aware placement of IoT application modules in fog-cloud computing paradigm. In: IFIP/IEEE symposium on integrated network and service management (IM), pp 1222–1228. IEEE
94. Tang Z, Zhou X, Zhang F, Jia W, Zhao W (2019) Migration modeling and learning algorithms for containers in fog computing. *IEEE Comput Archit Lett* 12(05):712–725
95. Tinini RI, Reis LC, Batista DM, Figueiredo GB, Tornatore M, Mukherjee B (2017) Optimal placement of virtualized BBU processing in hybrid cloud-fog RAN over TWDM-PON. In: IEEE global communications conference (GLOBECOM), pp 1–6
96. Urgaonkar R, Wang S, He T, Zafer M, Chan K, Leung KK (2015) Dynamic service migration and workload scheduling in edge-clouds. *Perf Eval* 91:205–228
97. Vaquero LM, Roderio-Merino L (2014) Finding your way in the fog: towards a comprehensive definition of fog computing. *ACM SIGCOMM Comput Commun Rev* 44(5):27–32
98. Velasquez K, Abreu DP, Curado M, Monteiro E (2017) Service placement for latency reduction in the Internet of Things. *Ann Telecommun* 72(1–2):105–115
99. Velasquez K, Abreu DP, Paquete L, Curado M, Monteiro E (2020) A rank-based mechanism for service placement in the fog. In: IFIP networking, pp 64–72
100. Venticinque S, Amato A (2019) A methodology for deployment of IoT application in fog. *J Ambient Intell Human Comput* 10(5):1955–1976
101. Wang N, Varghese B (2020) Context-aware distribution of fog applications using deep reinforcement learning. arXiv preprint [arXiv:2001.09228](https://arxiv.org/abs/2001.09228)
102. Wang S, Urgaonkar R, He T, Chan K, Zafer M, Leung KK (2016) Dynamic service placement for mobile micro-clouds with predicted future costs. *IEEE Trans Parallel Distrib Syst* 28(4):1002–1016
103. Wang S, Zafer M, Leung KK (2017) Online placement of multi-component applications in edge computing environments. *IEEE Access* 5:2514–2533
104. Wöbker C, Seitz A, Mueller H, Bruegge B (2018) Fogernetes: Deployment and management of fog computing applications. In: IEEE/IFIP network operations and management symposium (NOMS), pp 1–7
105. Xia Y, Etchevers X, Letondeur L, Coupaye T, Desprez F (2018) Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed IoT applications in the fog. In: 33rd annual ACM symposium on applied computing, pp 751–760
106. Xia Y, Etchevers X, Letondeur L, Lebre A, Coupaye T, Desprez F (2018) Combining heuristics to optimize and scale the placement of IoT applications in the fog. In: IEEE/ACM 11th international conference Utility and cloud computing (UCC), pp 153–163

107. Yadav R, Baranwal G (2019) Trust-aware framework for application placement in fog computing. In: IEEE international conference on advanced networks and telecommunication systems (ANTS)
108. Yadav V, Natesha B, Guddeti RMR (2019) GA-PSO: Service allocation in fog computing environment using hybrid bio-inspired algorithm. In: IEEE region 10 conference (TENCON), pp 1280–1285
109. Yang L, Cao J, Liang G, Han X (2015) Cost aware service placement and load dispatching in mobile cloud systems. *IEEE Trans Comput* 65(5):1440–1452
110. Yao H, Bai C, Zeng D, Liang Q, Fan Y (2015) Migrate or not? Exploring virtual machine migration in roadside cloudlet-based vehicular cloud. *Concurr Comput Pract Exp* 27(18):5780–5792
111. Yousefpour A, Patil A, Ishigaki G, Kim I, Wang X, Cankaya HC, Zhang Q, Xie W, Jue JP (2019) Fog-Plan: a lightweight QoS-aware dynamic fog service provisioning framework. *IEEE Internet Things J* 6(3):5080–5096
112. Zhu H, Huang C (2017) Availability-aware mobile edge application placement in 5g networks. In: IEEE global communications conference (GLOBECOM), pp 1–6

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.