Optimization Problems in Fog and Edge Computing[1]

Zoltán Ádám Mann

1 Introduction

Fog / edge computing arises through the increasing convergence and integration of several – traditionally distinct – disciplines: cloud computing on one hand, mobile computing and the Internet of Things (IoT) on the other hand, and advanced networking technologies as a glue between them. The main idea is to combine the strengths of these technologies to provide the necessary compute power to end-user applications in a cost-effective and secure way, with low latencies. Thus, fog / edge computing brings significant benefits to all of the underlying fields.

The notions of fog computing and edge computing are somewhat vaguely defined in the literature and have largely overlapping meaning [1]. In this chapter, we use the terms "fog computing" and "edge computing" interchangeably to refer to an architecture combining cloud computing with resources on the network edge and end-user devices.

In cloud computing, there has been an evolution for several years from centralized architectures (one or few large data centers) towards increasing decentralization (several smaller data centers), which is still continuing, and fog / edge computing is a natural next step on this evolution trajectory [2]. Geographically distributed data centers lead to decreased latency for applications involving distributed data sources and sinks (e.g., users or sensors / actuators), since each data source / sink can be

[1] This article is Chapter 5 in the book "Fog and Edge Computing: Principles and Paradigms", edited by Rajkumar Buyya and Satish Narayana Srirama, published by John Wiley & Sons, 2019. ISBN: 9781119524984

served by a nearby data center. Other benefits include improved fault tolerance as well as access to green energy sources of limited capacity [3].

From the point of view of mobile computing and IoT, the devices' limited computational capacity and limited battery life span are major challenges [4]. By offloading resource-intensive compute tasks to more powerful nodes – such as servers in a data center or compute resources at the network edge – the range of possible applications can be widened significantly [5].

Optimization plays a crucial role in fog computing. For example, minimizing latency and energy consumption are just as important as maximizing security and reliability. Because of the high complexity of typical fog deployments (many different types of devices, with many different types of interactions) and their dynamic nature (mobile devices coming and going, devices or network connections failing permanently or temporarily etc.), it has become virtually impossible to ensure the best solution by design. Rather, the best solution should be determined using appropriate optimization techniques.

For this purpose, it is vital to define the relevant optimization problem(s) carefully and precisely. Indeed, the used problem formulation can have dramatic consequences on the practical applicability of the approach (e.g., omitting an important constraint may lead to solutions that cannot be applied in practice), as well as on its computational complexity.

Research on fog computing is still in its infancy. Some specific optimization problems have been defined, but in an ad hoc manner, independently from each other. As a result, it is difficult to compare or combine different approaches, because they usually address different variants or facets of the same problem and such subtle differences are often not apparent. (Earlier, we have witnessed a similar situation in cloud

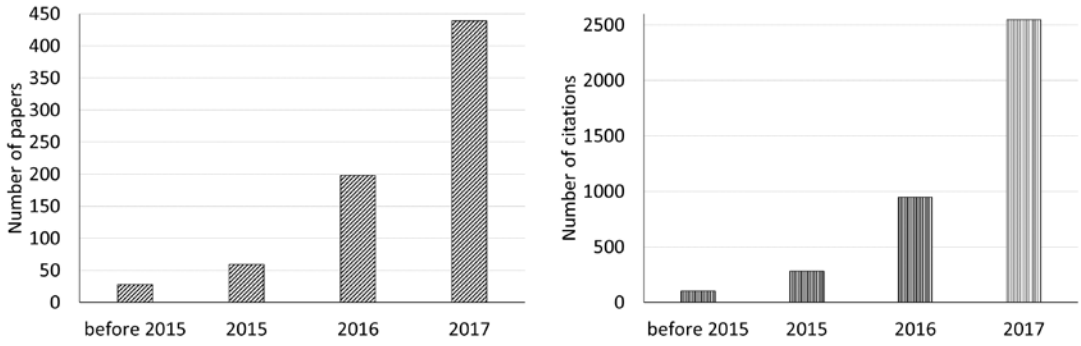
computing research as well [6].) Also, the quality and level of detail of existing problem formulations is quite heterogeneous.

Therefore, the aim of this chapter is to propose a generic conceptual framework for optimization problems in fog computing, based on consistent, well-defined, and formalized notation for constraints and optimization objectives. Using a taxonomy of problem formulations, their relationships will become clear, also highlighting the gaps that necessitate further research. With this standard reference, we hope to contribute significantly to the maturation of this field of research.

2 Background / Related Work

The concept of fog computing was introduced by Cisco in 2012 as a means to extend cloud computing capabilities to the network edge, thus enabling more advanced applications [7]. Since then, an increasing number of research papers have been published on fog computing. This is exemplified by Figure 1, which shows the development of the number of papers and number of citations in fog computing, available in the Scopus database[2] on 7th December 2017. The used search query was “TITLE-ABS-KEY ("fog computing")”, meaning that the phrase “fog computing” must occur in the title, the abstract, or the keywords of the paper.

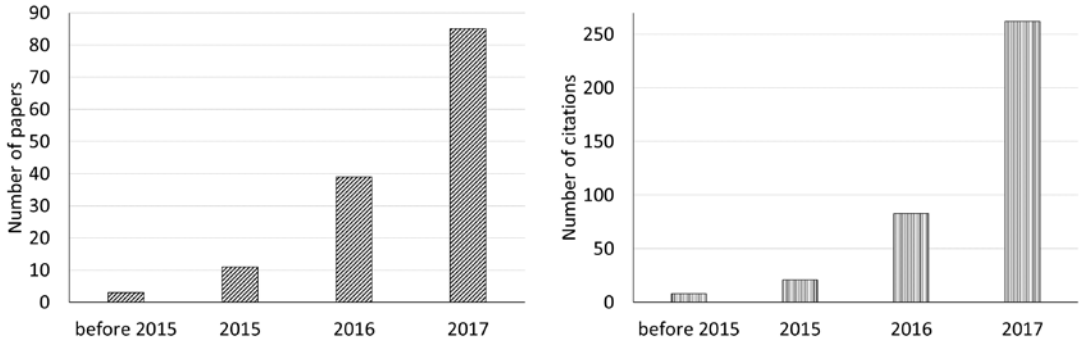
[2] https://www.scopus.com



(a) (b)

Figure 1: Number of (a) papers and (b) citations in fog computing

Several of those papers describe technologies, architectures, and applications in a fog computing setting. However, the number of papers that deal with optimization in fog computing is also quickly rising. This is demonstrated by Figure 2, which shows the number of papers and number of citations obtained from the Scopus database on 7th December 2017, with the search query “TITLE-ABS-KEY ("fog computing") AND TITLE-ABS-KEY (optim*)”, meaning that both the phrase “fog computing” and a word starting with “optim” (like optimal, optimized, or optimization) must occur in the title, the abstract, or the keywords of the paper.



(a) (b)

Figure 2: Number of (a) papers and (b) citations about optimization in fog computing

Later in Section 9, when the essential characteristics of optimization problems in fog computing have already been defined, we will show how existing literature on optimization in fog computing can be classified.

3 Preliminaries

Before delving into optimization problems and optimization approaches in fog computing, we describe some essential properties and notions of optimization in general.

An optimization problem is generally defined by the following [8]:

- a list of variables $\bar{x} = (x_1, \dots, x_n)$
- the domain – i.e., the set of valid values – of each variable; the domain of variable x_i is denoted by D_i
- a list of constraints $(C_1, \dots, C_m)$; constraint C_j relates to some variables $x_{j_1}, \dots, x_{j_k}$ and defines the valid tuples for those variables in the form of a set $R_j \subseteq D_{j_1} \times \cdots \times D_{j_k}$
- an objective function $f: D_1 \times \cdots \times D_n \to \mathbb{R}$

The problem then consists of finding appropriate values $v_1, \dots, v_n$ for the variables, such that all of the following holds:

(1) $v_i \in D_i$ for each $i = 1, \dots, n$

(2) for any constraint C_j relating to variables $x_{j_1}, \dots, x_{j_k}$, it holds that $(v_{j_1}, \dots, v_{j_k}) \in R_j$

(3) $f(v_1, \dots, v_n)$ is maximum among all $(v_1, \dots, v_n)$ tuples that satisfy (1) and (2)

A tuple $(v_1, \dots, v_n)$ that satisfies (1) and (2) is called a solution of the problem. Thus, the goal is to find the solution with highest f value. At least, this is the case for maximization problems (as defined above). For a minimization problem, the goal is to find

the solution with lowest f value, which is equivalent to finding the solution that maximizes the objective function $f' = -f$. In case of minimization problems, the objective function is often called cost function because it represents some – real or fictive – cost that needs to be minimized.

It is important to differentiate between a practical problem in engineering – e.g., minimization of power consumption in fog computing – and a formally defined optimization problem as outlined above. Deriving a formalized optimization problem from a practical problem is a non-trivial process, in which the variables, their domains, the constraints, and the objective function have to be defined. In particular, there are usually many different ways to formalize a practical problem, leading to different formal optimization problems. Formalizing the problem is also a process of abstraction, in which some non-essential details are suppressed or some simplifying assumptions are made. Different formalizations of the same practical problem may exhibit different characteristics for example in terms of computational complexity. Therefore, the decisions made during problem formalization have high impact. Problem formalization implies finding the most appropriate trade-off between the generality and applicability of the formalized problem on one hand and its simplicity, clarity, and computational tractability on the other hand. This requires expertise and an iterative approach in which different ways of formalizing the problem are evaluated.

It should be mentioned that some papers jump from an informal problem description directly to devising some algorithm, without formally defining the problem first. This, however, has the disadvantage of prohibiting precise reasoning about the problem itself, e.g., about its computational complexity or its similarity with known other problems that could lead to the adoption of existing algorithms.

In the above definition of a general optimization problem, it was assumed that there is a single real-valued objective function. However, in several practical problems, there are multiple objectives and the difficulty of the problem often lies in balancing between conflicting objectives. Let the objective functions be $f_1, \dots, f_q$, where the aim is to maximize all of them. Since there is generally no solution that maximizes all of the objective functions simultaneously, some modification is necessary to obtain a well-defined optimization problem. The most common approaches for that are the following [9]:

- Adding lower bounds to all but one of the objective functions and maximizing the last one. That means adding constraints of the form $f_s(v_1, \dots, v_n) \geq l_s$, where l_s is an appropriate constant, for all $s = 1, \dots, q-1$, and maximizing $f_q(v_1, \dots, v_n)$.
- Scalarizing all objective functions into a single combined objective function $f_{combined}(v_1, \dots, v_n) = F\big(f_1(v_1, \dots, v_n), \dots, f_q(v_1, \dots, v_n)\big)$. Common choices for the function F are product and weighted sum.
- Looking for Pareto-optimal solutions. A solution $(v_1, \dots, v_n)$ dominates another solution $(v'_1, \dots, v'_n)$, if $f_s(v_1, \dots, v_n) \geq f_s(v'_1, \dots, v'_n)$ holds for all $s = 1, \dots, q$, and $f_s(v_1, \dots, v_n) > f_s(v'_1, \dots, v'_n)$ holds for at least one value of s, i.e., $(v_1, \dots, v_n)$ is at least as good as $(v'_1, \dots, v'_n)$ regarding each objective and it is strictly better regarding at least one objective. A solution is called Pareto-optimal, if it is not dominated by any other solution. In other words, a Pareto-optimal solution can only be improved with regard to an objective if it is wors-

ened regarding some other objective. Different Pareto-optimal solutions of a problem represent different trade-offs between the objectives, but all of them are optimal in the above sense.

4 The Case for Optimization in Fog Computing

The fundamental motivation for the developments leading to fog computing are strongly related to some important quality attributes that should be improved. As explained earlier, fog computing can be seen as an extension of cloud computing towards the network edge, with the aim of providing lower latencies for latency-critical applications within end devices. In other words, the optimization objective of minimizing latency is a major driving force behind fog computing [10].

On the other hand, from the point of view of end devices, fog computing promises significantly increased compute capabilities, enabling the execution of compute-intensive tasks quickly and without major impact on energy consumption of the device. Therefore, optimization relating to execution time and energy consumption are also fundamental aspects of fog computing.

As we will see shorty in Section 6, several other optimization objectives are relevant to fog computing as well. Moreover, there are non-trivial interactions, sometimes also conflicts, among the different objectives. Hence it is important to systematically study the different aspects of optimization in fog computing.

5 Formal Modelling Framework for Fog Computing

Before discussing individual optimization objectives, it is useful to define a generic framework for modeling – different variants of – the problem.

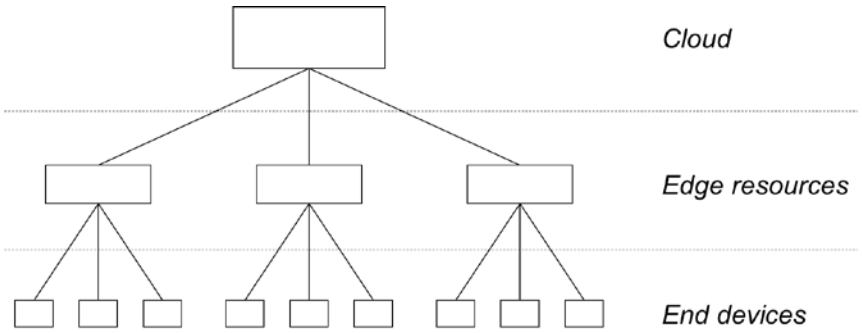


Figure 3: Three-layer model of fog computing

As shown in Figure 3, fog computing can be represented by a hierarchical three-layer model [11]. Higher layers represent higher computational capacity, but at the same time also higher distance – and thus higher latency – from the end devices. On the highest layer is the cloud with its virtually unlimited, high-performance, and cost- and energy-efficient resources. The middle layer consists of a set of edge resources: machines offering compute services near the network edge, e.g. in base stations, routers, or small, geographically distributed data centers of telecommunication providers. The edge resources are all connected to the cloud. Finally, the lowest layer contains the end devices like mobile phones or IoT devices. Each end device is connected to one of the edge resources.

More formally, let c denote the cloud, E the set of edge resources, D_e the set of end devices connected to edge resource $e \in E$, and $D = \bigcup_{e \in E} D_e$ the set of all end devices. The set of all resources is $R = \{c\} \cup E \cup D$. Each resource $r \in R$ is associated with a compute capacity $a(r) \in \mathbb{R}^+$ and a compute speed $s(r) \in \mathbb{R}^+$. Moreover, each resource has some power consumption, which depends on its computational load. Specifically, the power consumption of resource r increases by $w(r) \in \mathbb{R}^+$ for every instruction to be carried out by r.

The set of links between resources is $L = \{ce: e \in E\} \cup \{ed: e \in E, d \in D_e\}$. Each link $l \in L$ is associated with a latency $t(l) \in \mathbb{R}^+$ and a bandwidth $b(l) \in \mathbb{R}^+$.

Moreover, transmitting one more byte of data over link l increases power consumption by $w(l) \in \mathbb{R}^+$. Table 1 gives an overview of the used notation.

Table 1: Notation overview

Notation	Explanation
c	cloud
E	set of edge resources
D_e	set of end devices connected to edge resource $e \in E$
R	set of all resources
$a(r)$	compute capacity of resource $r \in R$
$s(r)$	compute speed of resource $r \in R$
$w(r)$	marginal energy consumption of resource $r \in R$
L	set of all links between resources
$t(l)$	latency of link $l \in L$
$b(l)$	bandwidth of link $l \in L$
$w(l)$	marginal energy consumption of link $l \in L$

6 Metrics

As already mentioned, there are several metrics that need to be optimized in a fog computing system. Depending on the specific optimization problem variant, these metrics may indeed be optimization objectives, but they can also be used as constraints. For example, one problem variant may look at a real-time application, in which overall execution time needs to be constrained by an upper bound, while en-

ergy consumption should be minimized. In another application, the finite battery capacity of a mobile device may be the bottleneck, so that energy consumption should be constrained by an upper bound, while execution time should be minimized.

Independently from the specific application – and hence, problem variant – there are some metrics that play an important role in fog computing. These metrics are reviewed next.

6.1 Performance

There are several performance-related metrics, like execution time, latency, and throughput. Generally, performance is related to the amount of time needed to accomplish a certain task. In a fog computing setting it is important to note that accomplishing a task usually involves multiple resources, often on different levels of the reference model of Figure 3. Hence, the completion time of the task may depend on the computation time of multiple resources, plus the time for data transfer between the resources. Some of these steps might be made in parallel (e.g., multiple devices can perform computations in parallel), whereas others must be made one after the other (e.g., the results of a computation can only be transferred once they have been computed). The total execution time depends on the critical path of compute and transfer steps. For instance, if a computation is partly done in an end device and partly offloaded from the end device to an edge resource, this may lead to a situation such as the one depicted in Figure 4, in which the total execution time is determined by the sum of multiple computation and data transfer steps.

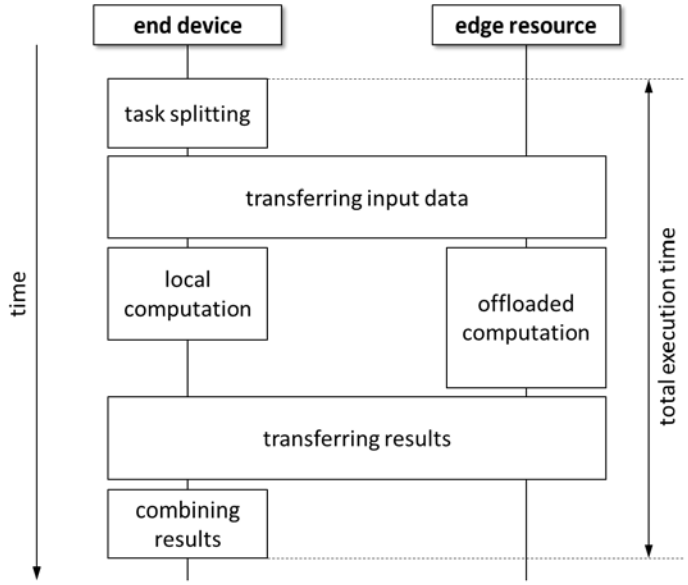


Figure 4: Total execution time of an example computation offloading scenario

6.2 Resource usage

Especially in the lower layers of the reference model of Figure 3, the economical use of the scarce resources is vital. This particularly applies to end devices which typically have very limited CPU and memory capacity. Edge resources typically offer higher capacities, but also those capacities can be limited, given that edge resources also may include machines like routers that do not offer exhaustive computational capabilities. To some extent, CPU usage can be traded off with execution time, i.e., overbooking the CPU may lead to a situation where the application is still running, but more slowly. This may be acceptable for some applications, but not for time-critical ones. Moreover, memory poses a harder constraint on resource consumption, since overbooking the memory may lead to more serious problems like application failure. Beyond CPU and memory, also network bandwidth can be a scarce resource, both between end devices and edge resources and between edge resources and the cloud. Hence, also the use of network bandwidth may have to be either minimized or constrained by an upper bound. It is important to note that, in contrast to performance,

which is a global metric spanning multiple resources, resource consumption needs to be considered at each network node and link separately.

6.3 Energy consumption

Energy can also be seen as a scarce resource, but it is quite different from the other resource types considered above. Energy is consumed by all resources as well as the network. Even idle resources and unused network elements consume energy, but their energy consumption increases with usage. Generally, assuming that the power consumption of a resource depends linearly on its CPU load is a good approximation [12]. It is important though to highlight the difference between power consumption and energy consumption, since energy consumption also depends on the amount of time during which power is consumed. Thus, it is for instance beneficial in terms of overall energy consumption to move a compute task from one resource to a significantly faster one, even if the faster machine has slightly higher power consumption.

Energy consumption is important on each layer of the fog, but in different ways. For end devices, battery power is often a bottleneck and thus preserving it as much as possible is a primary concern. Edge resources are typically not battery-powered; hence, their energy consumption is less important. For the cloud, energy consumption is again very important, but because of its financial implications: electric power is a major cost driver in cloud data centers. Finally, also the overall energy consumption of the whole fog system is important because of its environmental impact.

6.4 Financial costs

As already mentioned, energy consumption has implications on financial costs. But also other aspects influence costs. For example, the use of the cloud or edge infrastructure may incur costs. These costs can be fixed or usage-based, or some combination thereof. Similarly, also the use of the network for transferring data may incur costs.

6.5 Quality attributes

All aspects covered so far are easily quantifiable. However, they are not sufficient to guarantee a high quality of experience for users. For this, also quality attributes like reliability [13], security [11], and privacy [14] need to be taken into account, which are harder to quantify.

Traditionally, such quality attributes are not captured by optimization problems, but rather addressed with appropriate architectural or technical solutions. For instance, reliability may be achieved by creating redundancy in the architecture, security may be achieved by using appropriate cryptographic techniques for encryption, while privacy may be achieved by applying anonymization of personal data. Nevertheless, there are several ways to address also quality attributes during optimization of a fog system, as shown by the following representative examples:

- To increase reliability, it is beneficial to let multiple resources perform the same critical computations in parallel, so that the result is available even if some of the resources stop working or become unreachable, and also to compare the results with each other to filter out flawed results. The higher the number of resources used in parallel, the higher level of reliability can be achieved this way. Therefore, the number of

resources used in parallel is an important optimization objective that should be maximized.

- Both security and privacy concerns may be mitigated by preferring trusted resources. Using existing techniques to quantify trust, for instance based on reputation scores [14], the usage of trusted resources becomes an optimization objective, in which trust levels of the used resources should be maximized.
- Co-location of computational tasks belonging to different users / tenants may increase the likelihood of tenant-on-tenant attacks. Therefore, minimizing the number of tenants whose tasks are co-located is an optimization objective that helps to keep security and privacy risks at an acceptably low level.
- Co-location of tasks belonging to the same user decreases the need for exchanging data over the network, which in turn decreases the likelihood of eavesdropping, man-in-the-middle, and other network-based attacks. Hence, minimizing the number of resources used by a user also helps in decreasing risks related to information security.

It is important to note that the above optimization objectives relating to quality attributes typically conflict with other optimization objectives relating to costs, performance, etc. For example, increasing redundancy may be beneficial for improving reliability but at the same time it can lead to higher costs. Similarly, preferring service providers with high reputation is advantageous from the point of view of security, but may also lead to higher costs. Constraining co-location options may improve privacy, but may lead to worse performance or higher energy consumption, and so on. This is

one of the main reasons why it is beneficial to include also quality attributes in optimization problems, because this enables explicit reasoning about the optimal trade-off between the conflicting objectives.

7 Optimization opportunities along the fog architecture

Optimization problems in fog computing can be classified according to which layer(s) of the three-layer fog model (cf. Figure 3) is/are involved.

In principle, it is possible that only one layer is involved. This, however, is typically not regarded as fog computing. For example, if only the cloud layer is involved, then we have a pure cloud optimization problem. Likewise, if only end devices are involved, then the problem would not be in the realm of fog computing, but rather – depending on the kinds of devices and their interconnections – in mobile computing, IoT, wireless sensor networks etc.

Therefore, real fog computing problems involve at least two layers. This consideration leads to the following classification of optimization problems in fog computing:

- Problems involving the cloud and the edge resources. This is a meaningful setting, which allows for example to optimize overall energy consumption of cloud and edge resources, subject to capacity and latency constraints [15]. This setup shows some similarity to distributed cloud computing; a potential difference is that the number of edge resources can be several orders of magnitude higher than the number of data centers in a distributed cloud.

- Problems involving edge resources and end devices. The collaboration of end devices with edge resources (e.g., offloading computations) is a typical fog computing problem, and because of the limited resources of end devices, optimization plays a vital role in such cases. An often studied special case of this problem setup is when a single edge resource is considered together with the end devices that it serves [16]. However, the more general case in which multiple edge resources – together with the end devices that they serve – are considered has also received attention [17]. The latter leads to more complex optimization problems, but has the advantage to balance computational load among multiple edge resources.
- In principle, all three layers can be optimized together. This, however, is seldom studied, probably because of the difficulties of such optimization. The difficulties relate on one hand to the computational complexity of large-scale optimization problems involving decision variables for all fog resources. On the other hand, many different technical issues would have to be integrated into a single optimization problem to capture the different optimization concerns of the cloud, the edge resources, and the end devices, which is challenging in itself. In addition, changes to the cloud, the edge resources, and the end devices are typically made by different stakeholders on different time scales, which is also a rationale for independent optimization of the different fog layers.

In each of the fog layers, optimization may target the distribution of data, code, tasks, or a combination of these. In data-related optimization, decisions have to

be made about which pieces of data are stored and processed where in the fog architecture. In code-related optimization, program code can be deployed on multiple resources and the goal is to find the optimal placement of the program code. Finally, in task-related optimization, the aim is to find the optimal split of tasks among multiple resources.

Finally, it should be noted that the distributed nature of fog computing systems may make it necessary to perform optimization also in a distributed fashion. Ideally, the locally optimal decisions of the participating autonomous resources should lead to a globally optimal behavior [18].

8 Optimization opportunities along the service lifecycle

Just like cloud computing, fog computing is also characterized by the provision and consumption of services. By looking at the different optimization opportunities at the different stages of the service lifecycle, one can differentiate between the following options:

- **Design-time optimization**. When a fog service is designed, exact information about the end devices to be served is typically not available. Hence, optimization will be constrained mostly to the cloud and edge layers of the architecture, where more information may be available already at design time. Concerning the end devices, optimization is constrained to questions dealing with types of devices (as opposed to device instances which will be known only during run time).

- **Deployment-time optimization**. When the deployment of the service on specific resources is planned, the available information of the resources can be used to make further optimization decisions. For example, the exact capacity of the edge resources to be used may become available at this time, so that the split of tasks between the cloud and the edge resources can be (re-)optimized.
- **Run-time optimization**. Although some aspects of a fog system may be optimized in advance (i.e., during design time or deployment time), many important aspects become clear only when the system is running and used. Examples include the specific end devices with their parameters (e.g., compute capacity) and the compute tasks that the end devices want to offload to the edge resources. These aspects are vital for making sound optimization decisions. Moreover, these aspects keep changing during the operation of the system. As a consequence, much of the system operation needs to be optimized during run time. This requires continuous monitoring of important system parameters, analysis of whether the system still operates with acceptable effectiveness and efficiency, and re-optimization whenever necessary [18].

As can be seen, run-time optimization plays a very important role in the optimization of fog computing systems. This has some important consequences. First, the time available for executing an optimization algorithm during run time is seriously limited, thus the adopted optimization algorithms have to be fast. Second, run-time optimization is usually not about laying out a system from scratch, but rather about adapting an existing setup. This implies in particular that the costs associated with changes to the system have to be taken into account.

9 Towards a taxonomy of optimization problems in fog computing

The different aspects of optimization covered so far can form the basis to devise a taxonomy of optimization problems in fog computing. In the following, we illustrate this by means of classifying some representative publications taken from the literature along the presented dimensions.

Table 2: Classification of the work of Do et al. [19] according to the presented dimensions

Paper:	Do et al.: A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing [19]
Context / domain:	Video streaming service with a central cloud serving distributed edge resources which in turn serve end devices
Considered metrics:	• "Utility" (weighted data rate of the edge resources) • Compute capacity of the cloud data center • Energy consumption of the cloud data center
Considered layer / resources:	• Cloud • Edge resources
Phase in lifecycle:	Design / deployment time
Optimization algorithm:	Distributed iterative improvement

As a first example, Table 2 shows the classification of the work of Do et al. [19]. This paper considers a video streaming service, consisting of a central cloud data center

and a huge number of geographically distributed edge resources (called fog computing nodes or FCNs in the paper) which are to provide end devices with streaming video. The aim is to determine the data rate of video streaming for each edge resource, taking into account the different utility provided by different data rates at different edge resources, data center energy consumption, and the workload capacity of the data center. The paper proposes a distributed iterative improvement algorithm inspired by the ADMM (Alternating Direction Method of Multipliers) method.

Table 3: Classification of the work of Sardellitti et al. [20] according to the presented dimensions

Paper:	Sardellitti et al.: Joint optimization of radio and computational resources for multicell mobile-edge computing [20]
Context / domain:	Computation offloading from mobile end devices to an edge resource
Considered metrics:	• Energy consumption of the end devices • Total time to transfer and execute offloaded tasks • Amount of compute power of edge resource occupied by offloaded tasks of the devices
Considered layer / resources:	• Edge resource • End devices
Phase in lifecycle:	Run time
Optimization algorithm:	Iterative heuristic using successive convex approximation

As another example, Table 3 shows the classification of the work of Sardellitti et al. [20] according to the presented dimensions. That paper considers the computation offloading problem in a mobile edge computing setting, where some mobile end devices offload some compute tasks to a nearby edge resource. For each compute task of each end device, it can be decided whether or not it should be offloaded, and in case of offloading which radio channel should be used for the communication. The optimization problem is formed in terms of energy consumption and latency. The paper first formulates the problem for a single end device, which can be solved explicitly in closed form. However, for several end devices with potentially interfering communication, the problem becomes much tougher (in particular, non-convex), which the authors solved by means of an appropriate heuristic.

Table 4: Classification of the work of Mushunuri et al. [21] according to the presented dimensions

Paper:	Mushunuri et al.: Resource optimization in fog enabled IoT deployments [21]
Context / domain:	Cooperating mobile robots sharing compute tasks
Considered metrics:	• Communication cost between end devices and edge resource • Battery power of end devices • CPU capacity of end devices and edge resource
Considered layer / resources:	• Edge resource • End devices
Phase in lifecycle:	Run time

Optimization algorithm:	Non-linear optimization with the COBYLA (Constrained Optimization By Linear Approximations) algorithm within the NLOpt library

Finally, Table 4 describes the work of Mushunuri et al. [21], which addresses the problem of finding the optimal work distribution among cooperating robots. The robots (end devices) offload their compute tasks to a server (edge resource), which distributes it among the end devices and itself. It is assumed that compute tasks can be split arbitrarily. The optimization, carried out at run time by the edge resource, takes into account the communication costs, battery status, and compute capacities of the devices, and uses an off-the-shelf non-linear optimization package.

As can be seen from these three examples that cover different optimization problems within fog computing, the presented aspects can be applied successfully to classify the approaches from the literature and capture their characteristics which are relevant for optimization.

10 Optimization Techniques

Already the three examples presented in Section 9 show that the optimization techniques adopted in fog computing optimization problems are quite heterogeneous. The following characteristics seem to be quite common though:

- Adoption of non-linear, sometimes even non-convex optimization techniques.
- Usage of heuristics (as opposed to exact algorithms) to derive – potentially suboptimal – results to hard problems with limited computational effort.

- Usage of distributed algorithms, accounting for the distributed resources and the distributed knowledge in fog computing.

In the future, with the maturation of the field, a consolidation of the used methods may take place. However, since the considered problem variants are also manifold, we expect the field to continue to require several different types of algorithmic techniques.

11 Future Research Directions

Fog computing is still in its early days, with optimization taking an ever more important role in it. Accordingly, there are several areas where significant future research is needed:

- **Co-optimization**. One of the key challenges in optimizing fog computing systems is that several different technical systems and sub-systems must be tuned to achieve an overall optimal, or at least good enough configuration. This includes on one hand the different devices making up a fog system and on the other hand the different technical aspects like networking, computation, volatile memory and persistent storage, sensors and actuators etc. Optimizing all those aspects together, or finding good ways decompose this huge optimization problem into sub-problems that can be solved mostly independently remains an important challenge for future research.
- **Balancing multiple optimization objectives**. Another important characteristic of optimization in fog computing is that multiple, often conflicting optimization objectives must be considered simultaneously.

Current practices to handle multi-criteria optimization in fog computing – e.g., using the weighted sum of the different optimization objectives – are simple and may lead to good results in several cases, but may lead to implausible solutions in extreme situations, hindering the practical adoption of such approaches. Finding more robust ways of incorporating multiple optimization objectives thus remains an important future research direction.

- **Algorithmic techniques**. So far, optimization algorithms have been selected largely arbitrarily, often based primarily on authors' previous experience with different techniques. With the maturation of the field, the community should develop a better understanding of which algorithmic approaches work well for which problem variants.
- **Evaluation of optimization algorithms**. Existing approaches were evaluated in rather ad hoc ways. Before methods can be transferred from research into practice, it is vital to evaluate the applicability of the proposed algorithms in a sound, thorough, and repeatable manner. This requires the definition of benchmark problems with publicly available problem instances, consensus in the community on evaluation methodologies and test environments, development of reliable and realistic simulators, and unbiased comparison of competing approaches under realistic – also including extreme – situations. Also theoretical methods to prove algorithm properties in a rigorous way will be necessary.

12 Conclusion

In this chapter, we have presented a review of optimization problems in fog computing. In particular, we have explained why optimization plays a vital role in fog computing and why it is important to define optimization problems unambiguously, preferably using a formal problem model. The most important aspects of optimization in fog computing have been reviewed according to multiple dimensions: the metrics that serve as optimization objectives or as constraints, the considered layers within the fog architecture, and the relevant phase in the service lifecycle. These dimensions also lend themselves to form a taxonomy, which can be used to classify existing or future problem variants.

We have also argued that there are several important directions for future research, including the improved handling of multiple optimization objectives, the co-optimization of multiple technical aspects, better understanding of which algorithmic techniques work best for which problem variant, and devising disciplined evaluation methodologies.

Acknowledgements

The work of Z. Á. Mann has been supported by the Hungarian Scientific Research Fund (Grant Nr. OTKA 108947) and the European Union's Horizon 2020 research and innovation program under grant 731678 (RestAssured).

References

[1] L. M. Vaquero, L. Rodero-Merino, Finding your way in the fog: Towards a comprehensive definition of fog computing, *ACM SIGCOMM Computer Communication Review*, 44(5):27-32 (October 2014).

[2] B. Varghese, R. Buyya, Next generation cloud computing: New trends and research directions, *Future Generation Computer Systems*, 79(3):849-861 (February 2018).

[3] E. Ahvar, S. Ahvar, Z. Á. Mann, N. Crespi, J. Garcia-Alfaro, R. Glitho, CACEV: A cost and carbon emission-efficient virtual machine placement method for green distributed clouds, in *IEEE International Conference on Services Computing*, pp. 275-282, IEEE, 2016.

[4] A. V. Dastjerdi, R. Buyya, Fog computing: Helping the Internet of Things realize its potential, *Computer*, 49(8):112-116 (2016).

[5] K. Kumar, Y.-H. Lu, Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43(4):51-56 (April 2010).

[6] Z. Á. Mann, Allocation of virtual machines in cloud data centers – A survey of problem models and optimization algorithms, *ACM Computing Surveys*, 48(1): article 11 (September 2015).

[7] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, In *Proceedings of the 1st ACM Mobile Cloud Computing Workshop*, pp. 13-15 (2012).

[8] Z. Á. Mann, *Optimization in computer engineering – Theory and applications*, Scientific Research Publishing (2011).

[9] R. T. Marler, J. S. Arora, Survey of multi-objective optimization methods for engineering, *Structural and Multidisciplinary Optimization*, 26(6):369-395 (April 2004).

[10] S. Soo, C. Chang, S. W. Loke, S. N. Srirama, Proactive mobile fog computing using work stealing: Data processing at the edge, *International Journal of Mobile Computing and Multimedia Communications*, 8(4):1-19 (2017).

[11] I. Stojmenovic, S. Wen, The fog computing paradigm: Scenarios and security issues, In *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 1-8 (2014).

[12] S. Rivoire, P. Ranganathan, C. Kozyrakis, A comparison of high-level full-system power models, In *Proceedings of the 2008 Conference on Power Aware Computing and Systems (HotPower '08)*, article 3 (2008).

[13] H. Madsen, B. Burtschy, G. Albeanu, F. Popentiu-Vladicescu, Reliability in the utility computing era: Towards reliable fog computing, *20th International Conference on Systems, Signals and Image Processing*, pp. 43-46 (2013).

[14] S. Yi, Z. Qin, Q. Li, Security and privacy issues of fog computing: A survey, *International Conference on Wireless Algorithms, Systems, and Applications*, pp. 685-695 (2015).

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

[16] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, *IEEE/ACM Transactions on Networking*, 24(5):2795-2808 (2016).

[17] J. Oueis, E. C. Strinati, S. Barbarossa, The fog balancing: Load distribution for small cell cloud computing, *81st IEEE Vehicular Technology Conference* (2015).

[18] J. O. Kephart, D. M. Chess, The vision of autonomic computing, *Computer*, 36(1):41-50 (2003).

[19] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, C. S. Hong, A proximal algorithm for joint resource allocation and minimizing carbon footprint in

geo-distributed fog computing, *International Conference on Information Networking*, pp. 324-329, IEEE (2015).

[20] S. Sardellitti, G. Scutari, S. Barbarossa, Joint optimization of radio and computational resources for multicell mobile-edge computing, *IEEE Transactions on Signal and Information Processing over Networks*, 1(2):89-103 (2015).

[21] V. Mushunuri, A. Kattepur, H. K. Rath, A. Simha, Resource optimization in fog enabled IoT deployments, *2nd International Conference on Fog and Mobile Edge Computing*, pp. 6-13 (2017).