electronics

MDPI

Article

Method for Dynamic Service Orchestration in Fog Computing

Nerijus Morkevicius *, Algimantas Venčkauskas, Nerijus Šatkauskas and Jevgenijus Toldinas

Department of Computers Science, Kaunas University of Technology, Studentų st. 50, LT-51368 Kaunas, Lithuania; algimantas.venckauskas@ktu.lt (A.V.); nerijus.satkauskas@ktu.lt (N.Š.); eugenijus.toldinas@ktu.lt (J.T.)
* Correspondence: nerijus.morkevicius@ktu.lt

Abstract: Fog computing is meant to deal with the problems which cloud computing cannot solve alone. As the fog is closer to a user, it can improve some very important QoS characteristics, such as a latency and availability. One of the challenges in the fog architecture is heterogeneous constrained devices and the dynamic nature of the end devices, which requires a dynamic service orchestration to provide an efficient service placement inside the fog nodes. An optimization method is needed to ensure the required level of QoS while requiring minimal resources from fog and end devices, thus ensuring the longest lifecycle of the whole IoT system. A two-stage multi-objective optimization method to find the best placement of services among available fog nodes is presented in this paper. A Pareto set of non-dominated possible service distributions is found using the integer multi-objective particle swarm optimization method. Then, the analytical hierarchy process is used to choose the best service distribution according to the application-specific judgment matrix. An illustrative scenario with experimental results is presented to demonstrate characteristics of the proposed method.

Keywords: fog computing; Internet of Things; service placement; fog service orchestration

check for updates

Citation: Morkevicius, N.; Venčkauskas, A.; Šatkauskas, N.; Toldinas, J. Method for Dynamic Service Orchestration in Fog Computing. *Electronics* **2021**, *10*, 1796. https://doi.org/10.3390/electronics10151796

Academic Editors: Kevin Lee and Ka Lok Man

Received: 29 June 2021
Accepted: 26 July 2021
Published: 27 July 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Fog computing acts as a missing link in the cloud-to-thing continuum. Services are provided closer to the edge of the network to enhance frequent services, latency, availability, and analysis. Fog computing places some computation resources in close proximity to a user where numerous heterogeneous end devices have to work in harmony. Control functions must work autonomously in such a heterogeneous and complex environment. Therefore, an orchestration is a centralized executable process to coordinate any interaction among any application or service [1]. Figure 1 shows the fog computing architecture.

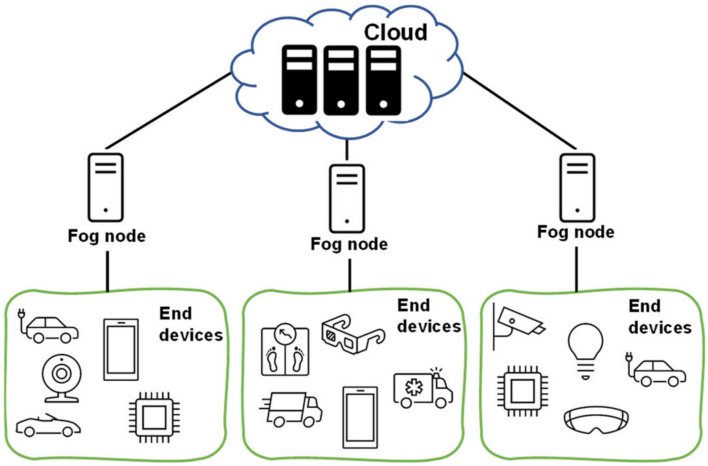

Figure 1. Fog computing architecture.

There is a wide variety of the application areas. As the review paper [2] classifies it in their fog computing application taxonomy, it is an application area which is made

of municipal services, smart citizens, smart education, smart healthcare, smart buildings, smart energy, smart governance, etc. The main concerns which were identified in the reviewed papers were the following: bandwidth management, power management, security, mobility, resource management, and latency.

In order to ensure any heterogeneous service provision infrastructure with a scalability, interoperability, and adaptability in mind, fog nodes have to be dynamically deployable [3]. These fog nodes also use constrained resources [1] if they are compared to a cloud infrastructure. Additionally, in order to give access to relevant services as well as to prevent any unauthorized access, access control or privacy control is required [4]. Having that in mind, security and privacy are a big concern [5,6]. Fog computing solutions frequently have insufficient security due to the fact that they rely on intensive communications with constrained devices [7] and constrained resources [8] located at the end device layer. If one of many end devices does not support a communication protocol of a sufficient strength, the security of the whole solution may be compromised. Moreover, roaming services are supported in some fog solutions, when the service follows a human, vehicle, etc., and travels from one fog node to another. In such cases, when any lower security service is brought to a secure fog node, the security of this fog node may be compromised. Similar problems may also occur with other QoS parameters such as a latency, bandwidth, range, etc. However, service orchestrators which are placed in the fog nodes may be used to monitor the whole situation in the fog node (including any communications with neighboring fog nodes) to take any required measures in the case of any potential violations of security and other QoS parameters.

After a dynamic service orchestrator deploys any relevant services within specific fog nodes, there is another hurdle to overcome, the optimization [9]. Fog computing keeps computing resources close to users and to the end nodes to reduce any delay for IoT services. It can also deal with the privacy, data locality, and bandwidth consumption. There are several objectives that can be enhanced by an optimization, such as a latency, cost, or energy management. It is a part of the quality of service (QoS) but it may come with a trade-off.

Any fog service orchestration can be challenging in such conditions. Cloud service orchestration may already be reliable enough at the moment [10], but the situation is different with fog computing. The complexity builds up due to the diversity of different services and resources. There are also concerns about interoperability, performance and service assurance, lifecycle management, scalability, security, and resilience, as identified in the review [11]. The paper [12] suggests that scalability, dynamics, and security are among the most common orchestration challenges which are specified in research papers.

Our goal in this research is to offer an effective orchestrator working in the fog layer of the fog computing architecture by providing effective means to solve the QoS- and security-related problems of the orchestration in heterogeneous fog layer devices and services. The idea is to check the placement of fog devices and services for any potential QoS and security issues in order to find any non-optimal distribution of services among fog nodes.

This paper includes three main contributions aimed at the fog service orchestration problem of an optimal placement of services inside the available fog nodes. First, it presents a detailed review of the fog service orchestration challenges and solutions proposed by other authors. The review clearly demonstrates the most promising mechanisms to be used for a fog service orchestration and it defines the problem more formally, which is addressed in this paper. Second, a two-stage optimal service placement algorithm based on integer multi-objective particle swarm optimization (IMOPSO) and the analytical hierarchy process (AHP) is proposed and formally described. The first stage of the proposed method finds a Pareto set of non-dominated potential placements of services, then the AHP is used to choose one best solution according to the application-specific judgment matrix provided by a user. Third, the proposed method is experimentally evaluated using an illustrative scenario, showing the performance of the algorithm in some likely situations.

The rest of this paper is structured in the following way: related publications are presented in Section 2, following by Section 3, which presents a more formal definition of the service orchestration problem. We describe our proposed method in Section 4. Evaluation and experimental results are summarized in Section 5, and, finally, Section 6 is dedicated to conclusions and a discussion.

2. Related Work Review

Fog orchestrator components, as concluded in the paper [5], can generally be divided into three main groups: fog orchestrator, fog node which can function as a fog orchestrator agent (FOA), and end devices. A fog orchestrator needs to consult its catalogs and certain monitoring data to make an orchestration plan. A fog orchestrator can start its orchestration manually or after reaching a benchmark, such as the availability of other nodes. FOA, in turn, can handle only local resources which are within that particular node.

The main research challenges in fog orchestration identified by Velasquez et al. [13] are the following: resource management, performance, and security management. Resource and service allocation optimization techniques are used, among others, to address these challenges. The problem is that an allocation procedure is a non-trivial problem, because essentially it is a multi-objective optimization problem.

In order to address the issues in the perspective of the fog computing, the authors of the paper [14] suggest using four of their proposed algorithms for their identified construction phase and maintenance phase. The construction phase aims to find some probable candidate locations to place the gateways while using the candidate location identification (CLI) algorithm. A Hungarian method-based topology construction (HTC) algorithm is used to select the optimal gateway locations. Meanwhile, the maintenance phase increases the processing resources in the gateways by intelligent sleep scheduling with the help of the vacation-based resource allocation (VRA) algorithm. Their processing and storage resources in the gateways are further improved based on the tracked data arrival rates with the help of the dynamic resource allocation (DRA) algorithm. Another option which can be beneficial to improving the performance of a network in terms of reliability and response is caching, as was noted in the publication [15]. The caching at the fog nodes can reduce computational complexity and network load. Even though the computing power is the most critical aspect in the fog node to complete specific tasks as the paper [16] suggests, an effective allocation of resources can vary due to limitations. These limitations may include the hierarchy of the fog network, network communication resources, and storage resources.

Yang et al. [17,18] confirm that the orchestration has to deal with a number of factors such as resource filtering and assignment, component selection and placement, dynamics with the runtime QoS, systematic data-driven optimization, or machine learning for orchestration. They implemented a novel parallel genetic algorithm-based framework (GA-Par) on Spark. They normalized the utility of security and network QoS into an objective fitness function within GA-Par. It reduces any security risks and performance deterioration. As their experiments later demonstrated, GA-Par outperforms a standalone genetic algorithm (SGA). Skarlat et al. [19] proposed to solve the fog service placement problem (FSPP) by using orchestration control nodes which place each service either in the fog cells or in the fog orchestration control nodes. The goal of optimization is to maximize the number of service placements in the fog nodes (rather than in cloud ones), while satisfying the requirements of each application. The authors used a genetic algorithm to find the optimal FSPP.

The authors of another paper identified resource allocation and provisioning as a challenging task considering dynamic changes of user requirements and limited resources [20]. They proposed their resource allocation and provisioning algorithms based on the resource ranking. They evaluated their algorithms in a simulation environment after extending their CloudSim toolkit. There are mainly two steps which are used to solve a deadline-based user dynamic behavior problem. First, they ranked resources based on processing power,

bandwidth, and response time. Later, they provided resources by prioritizing processing application requests.

As the deployment infrastructure has to adapt itself to extremely dynamic requirements, the fog layer may not provide enough resources and, meanwhile, the cloud layer can fail due to latency requirements [21]. The paper presents a rewriting-based approach to design and verify a self-adaptation and orchestration process in order to achieve a low latency and the right quantity of resources. An executable solution is provided based on Maude, the formal specification language. Properties are expressed using linear temporal logic (LTL). Their proposed cloud–fog orchestrator works as a self-adaptation controller. It is deployed in the fog layer as a fog node master for low latency requirements. The orchestrator triggers the right actions after a decision is made.

Smart service placement and management of services in big fog platforms can be challenging due to a dynamic nature of the workload applications and user requirements for low energy consumption and good response time. Container orchestration platforms are to help with this issue [22]. These solutions either use heuristics for their timely decisions or AI methods such as reinforcement learning and evolutionary approaches for dynamic scenarios. Heuristics cannot quickly adapt to extremely dynamic environments, while the second option can negatively impact response time. The authors also noted that they need scheduling policies which are efficient in volatile environments. They offer a gradient based optimization strategy using back-propagation of gradients with respect to the input (GOBI). They also developed a coupled simulation and container orchestration framework (COSCO) that enabled the creation of a hybrid simulation decision approach (GOBI*) which they used to optimize their quality of service (QoS) parameters.

As the service offloading is relevant enough in the perspective of time and energy, selection of the best fog node can be a serious challenge [23]. The researchers presented in their paper a module placement method by classification and regression tree algorithm (MPCA). Decision parameters select the best fog node, including authentication, confidentiality, integrity, availability, capacity, speed, and cost. They later analyzed and applied the probability of network resource utilization in the module offloading to optimize the MPCA.

Linear programming is another very popular optimization method used for resource allocation and service placement in fog nodes. Arkian et al. [24] linearized a mixed-integer non-linear program (MINLP) into the mixed-integer linear program (MILP) for optimal task distribution and virtual machine placement by using the minimization of cost. Velasquez et al. [25] proposed the service orchestrator which tries to minimize the latency of services using integer linear programming (ILP) to minimize the hop count between communicating nodes.

The authors of [26] present a method used to help deployments of composite applications in fog infrastructures, which have to satisfy software, hardware, and QoS requirements. The developed prototype (FogTorch) uses the Monte Carlo method to find the best deployment which ensures the lowest fog resource consumption—the aggregated averaged percentage of consumed RAM and storage in all the fog nodes.

A sequential decision-making Markov decision problem (MDP) enhanced by the technique of Lyapunov optimization is used by the authors of [27] to minimize operational costs of an IoT system while providing rigorous performance guarantees. The proposed method is intended to be used for a general problem of resource allocation and workload scheduling in cloud computing, but it may also be applied to a service placement problem in fog nodes.

As fog computing has a number of challenges to deal with, optimization is vital, and the classification of optimization problems can play an important role [28]. A service placement problem, in general, has been shown to be NP-complete by the authors of [29]. An optimization is typically made up of [30] (a) a set of variables to encode decisions, (b) a set of possible values for each variable, (c) a set of constraints which the variables are to satisfy, and (d) an objective function. Optimization solutions involving end devices and fog nodes differ based on their application area.

Our analyses of the methods used by other authors for service placement problem optimization, as well as findings of other researchers [31], show that various well-established optimization methods are used for this task, including integer linear programming, genetic algorithms, the Markov decision process, gradient based optimization, the Monte Carlo method, reinforcement learning, etc. The objective functions used by the authors of these methods vary from an overall cost minimization [24,27], to network latency [25], hop and service migration count [25], and response time and latency of the IoT system [26]. The literature review allows us to conclude that the most optimization methods tend to seek for an optimal placement of the services based on the most important parameter of the IoT system, which is represented by the objective function used in an optimization process. Other important parameters of IoT systems in such cases are used as restrictions, and usually include latency, power, bandwidth and QoS [24,26], CPU, RAM, and storage demands [19]. This kind of optimization problem formulation allows one to avoid the challenges of multi-objective optimization, but it may not be used in situations where more than one objective function is required. Some other approaches tend to evaluate several characteristics by combining them into one composite criterion, such as cost [24,27] or fog resource consumption [26] composed from an average RAM and storage usage in the fog nodes. The composite criteria calculation equations usually are provided by the authors of the proposed algorithms, and they use some predefined coefficients which are difficult to justify and validate. One very important challenge remains in this area in that case—how to find the best placement of the services according to several different heterogeneous criteria, with different origins and different units of a measurement, when they often contradict each other. The usage of composite criteria is not always the best answer to this.

The service placement optimization method proposed in this paper tries to address these challenges by using a multi-objective optimization method to find all non-dominated placements of the services and then to select one best placement using an analytical hierarchy process which simplifies the process of the criterion comparison performed by the experts of the application area. In this way, any number of objective functions (optimization criteria) may be used in the optimization process as long as experts are able to provide a consistent pair-to-pair comparison of their priority in the context of a concrete area of application.

3. Orchestrator Components and Architecture

In this paper, we consider the fog orchestration architecture and components presented in Figure 2. We have a service orchestrator in the cloud layer which is used to optimally distribute the services between several orchestrated fog nodes. The orchestrated fog nodes host some services which communicate with end devices, collect and process data, and make some local decisions on the control of actuators located in the end device layer. Special services (orchestrator agents) are physically located in each fog node and they communicate with the orchestrator to provide it with all the necessary information needed to make any decisions on service placement.

Orchestrator agents locally monitor the hardware and software environment of the fog nodes. They are aware of the current CPU and RAM usage, power requirement and energy levels, available communication protocols and bandwidth, security capabilities, state of the hosted services, etc. They summarize all the collected information to provide it to the orchestrator in the cloud layer. The orchestrator is aware of the current situation in all the fog nodes and, additionally, it has security and QoS requirements imposed by the application area of the IoT solution, and it makes decisions on starting, stopping, or moving particular services among the orchestrated fog nodes. The decisions made by the orchestrator are communicated down to the orchestrator agents inside the fog nodes, then the orchestrator agents initialize the required actions on the services. A control cycle performed inside the orchestrator is illustrated in Figure 3.

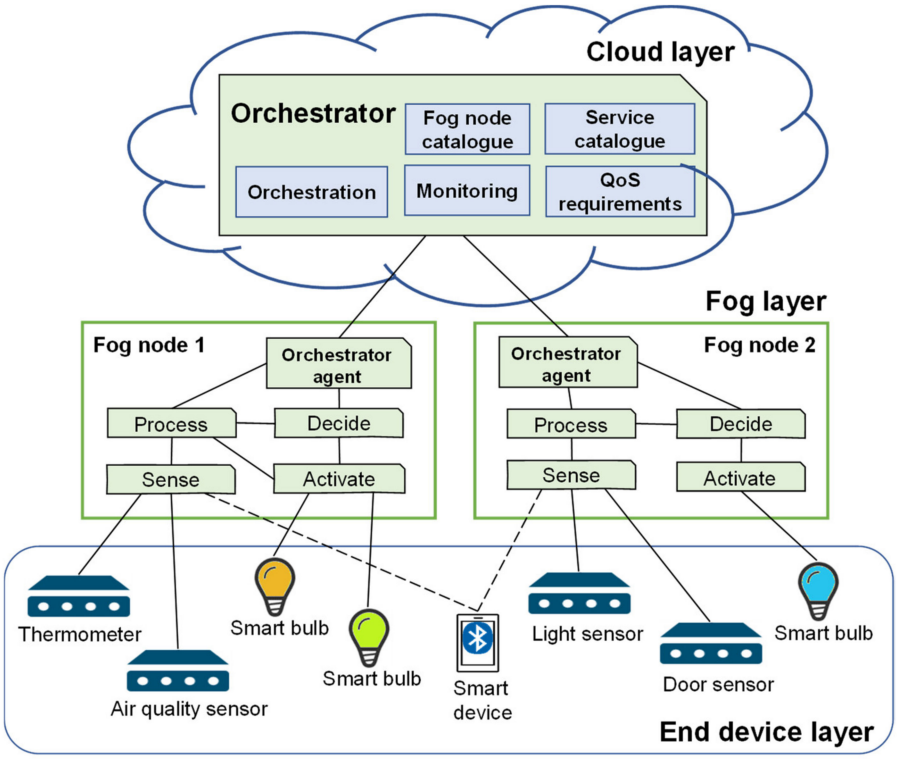

Figure 2. Fog orchestrator architecture and components.

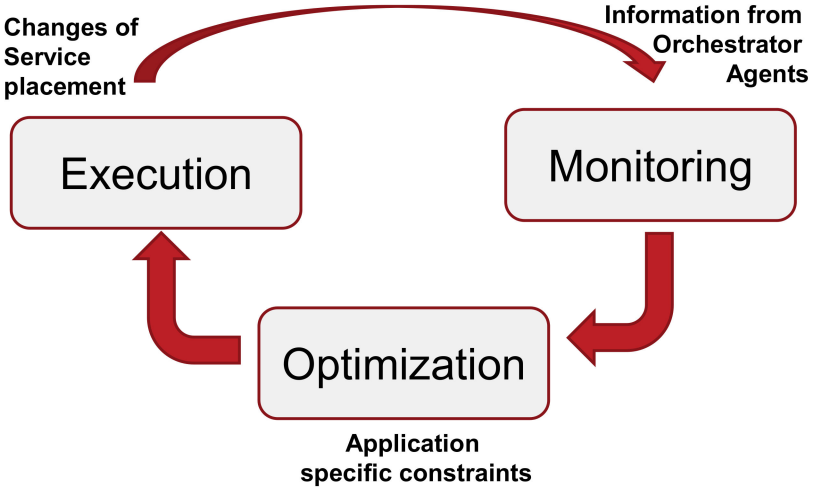

Figure 3. Control cycle inside the orchestrator.

The method of the fog service orchestration presented in this paper is intended to be used inside the orchestrator. The main task of the proposed method is to optimally distribute n services among k fog nodes according to the information collected from the corresponding fog nodes and the requirements imposed by the area of an application of the IoT system. This task of a service distribution is not trivial since several different optimization criteria which contradict each other must usually be considered (i.e., security level, energy consumption, bandwidth capabilities, latency, etc.). The number of possible different distributions of services among fog nodes increases rapidly with the increase in the number of available fog nodes and services. Any evaluation of all the possible placements of the services is infeasible, therefore, more sophisticated methods are needed. Moreover, the situation and the evaluation criteria can change dynamically due to the dynamic environment of the fog architecture. Some currently available fog nodes as well

as end devices may change their location or new fog nodes may even emerge while, on the other hand, some currently running services may become unused and some new services may occur.

4. Method for Fog Service Orchestration

We propose to use multi-objective optimization to decide which placement of n available services in k fog nodes is the best according to given constraints and conditions. The overall flow chart of the proposed two-stage optimization method is presented in Figure 4.

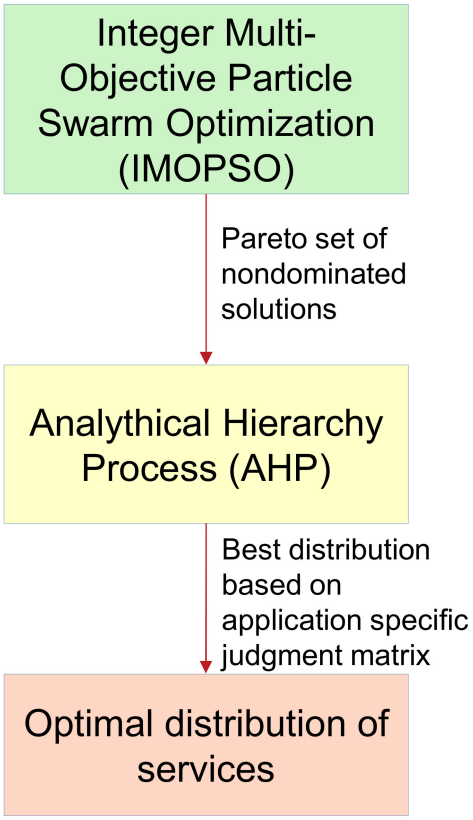

Figure 4. Flow chart of the proposed service distribution optimization process.

The optimization process has two main steps—multi-objective optimization and a multi-objective decision, but the problem must be expressed as a formal mathematical model before using any formal optimization methods. The following subsections describe the optimization process in detail. We summarize the key notations used in this paper in Table 1 in order to give a description of the optimization process.

Table 1. Key notations used in this paper.

Notation	Description
n	Total number of services
k	Total number of available fog nodes for hosting the services
$X_i = (x_1, x_2, \ldots, x_n)^T$, $x_j \in \{1, 2, \ldots, k\}$, $j = 1, 2, \ldots, n$	i-th possible distribution of services among the fog nodes, also the position of the i-th particle in n-dimensional definition area
m	Total number of evaluation criteria, also the number of objective functions
$f_j(x)$, $j = 1, 2, \ldots, m$	j-th evaluation criterion, objective function
$F_i = (f_1(X_i), f_2(X_i), \ldots, f_m(X_i))^T$	Score vector of the i-th particle
V_i, $V_i \in R^n$	Velocity of the i-th particle
$pBest_i$	The best score of the i-th particle
$pBPos_i$	The best position of the i-th particle
$gBest$	The best global score of all the particles
$gBPos$	Position of the particle with the best global score
S	Set of particles, swarm
R	External repository of particles, a set of Pareto optimal solutions
X_{opt}	The best service distribution among all the available fog nodes, particle with the best score

4.1. The Optimization Model of a Service Distribution Problem

The main task of this optimization procedure is to find an optimal distribution of n services among possible k fog nodes. Each fog node may have slightly different characteristics, but we assume that all the nodes are capable of running all the services. The goal of optimization is to distribute all the services in such a way that a set of important characteristics is optimal. Characteristics of the i-th possible service distribution X_i are expressed by the values of the objective functions $f_j(X_i)$, $j = 1, 2, \ldots, m$ and constraint conditions. The objective of the optimization process is to find the best service distribution X_{opt} which minimizes all the objective functions f_j, i.e., we have a multi-objective optimization problem:

$$X_{opt} = \arg\min_i F(X_i) \tag{1}$$

where $F(x) = \{f_1(x),\ f_2(x),\ \ldots,\ f_m(x)\}$ is a set of the objective functions, and $x \in \{X_i\}$ is a member of the set with all the possible service distributions.

Constraint conditions are expressed by the following equations:

$$\begin{cases} g_j(X_i) \geq 0, j = 1, 2, \ldots, n_g \\ h_K(X_i) = 0, k = 1, 2, \ldots, n_h \end{cases} \tag{2}$$

4.2. Objective Functions

Different fog nodes have different performance, network bandwidth, and security characteristics. Different distributions of services among the fog nodes may produce a working system with slightly different characteristics. For example, if one fog node supports a lower level of a security (due to limited hardware capabilities), and an important service is placed in this node, then the overall security of the whole system is reduced to the security of the least secure fog node. We consider multiple objective functions ($f_j(.),\ j = 1, 2, \ldots, m$) to evaluate all such situations, which include: overall security of the system, CPU utilization, RAM utilization, power utilization, range, etc. Some objective functions which were used in our experiments are provided in the following paragraphs.

A security of the whole system while using the *i*-th service distribution $f_{sec}(X_i)$ is defined by the lowest security of all the services. We assign security levels (expressed in security bits, according to the NIST publication [32]) to fog nodes based on their capabilities to support corresponding security protocols. We assume that services are capable working on all the fog nodes, then a value of the security criteria function $f_{sec}(X_i)$ for the service distribution X_i is the lowest security level of all the fog nodes in which at least one service is hosted. For example, if we have a situation where three fog nodes are able to provide 128 bits of security and one fog node is constrained to support only 86 bits of security, and if at least one service is hosted by it, then the overall security of the service distribution is equal to 86 bits, i.e., the value of the objective function $f_{sec}(X_i) = 86$. If we use our services in the application area which requires a specific level of security, then such a requirement is expressed as a constraint condition, i.e., if some application area requires at least 128 bits of security, then we have a corresponding constraint $g_{sec}(X_i) \geq 128$.

The criterion of CPU usage $f_{CPU}(.)$ evaluates how evenly, CPU utilization-wise, the services are distributed among the fog nodes. The main idea here is to try to decrease the overall CPU utilization of the system to allow hosting of additional services more easily in the case they occur during the runtime of the system. Each fog node has its CPU capabilities expressed in MIPS, which depend on HW capabilities of the corresponding fog node. All services are also evaluated for required CPU resources. To calculate the value of CPU usage of the whole system while using the *i*-th service distribution $f_{CPU}(X_i)$, we first calculate a relative CPU usage for each fog node (dividing the sum of CPU resources required by all the services hosted in each fog node by the capabilities of the corresponding fog nodes) and we find afterwards the maximum CPU utilization among all the fog nodes. The lower the maximum CPU utilization is, the better service distribution we have. We can obtain this situation while using this method of calculation, when some service distributions make up a CPU allocation greater than 100% in some fog nodes and, therefore, corresponding constraints are added to the optimization problem. The usage of this criterion automatically solves some frequent restrictions and incompatibilities, i.e., situations when some services require CPU resources which may not be provided by some fog nodes.

The criterion of RAM usage $f_{RAM}(.)$ which evaluates how evenly RAM utilization is distributed among any fog nodes hosting the services is very similar to CPU usage. The calculation of this criterion is the same as the calculation of CPU usage. A constraint which does not allow exceeding 100% of the RAM utilization in each fog node is also added.

A criterion of the power usage $f_{pw}(X_i)$ of possible service distribution X_i is evaluated using the average power requirements of each service (expressed in mW) and the available power of fog nodes (expressed in mW). The main objective of this evaluation is to maximize the overall runtime of the system. A calculation is performed by dividing the sum of power requirements of all the services hosted in each fog node by the available power of a corresponding fog node to find the maximum among all the fog nodes. A distribution of services is better in such a case when all the fog nodes are evenly loaded power-wise, i.e., the maximum power utilization is minimized.

The communication of fog devices with sensors and actuators is affected by the physical range between devices in some cases. Some communications protocols add strict requirements for the range as some of them may be less efficient if the communication range is increased. A criterion of the maximum range $f_{rng}(.)$ may be used to assess these properties. In this study, a criterion of the range is calculated by averaging the range of each fog node location with respect to all the devices the particular fog node is communicating with to find the maximum of these ranges among all the fog nodes hosting at least one service which requires communication with end devices. The main idea of this criterion is to prefer a shorter communication path as it ensures better performance in most cases. Any corresponding constraints on the range may be also added if a communication protocol induces such restrictions.

Other application-specific criteria such as local storage capabilities, communication latency, bandwidth, etc. may also be evaluated, defining corresponding objective functions

representing system characteristics which are important in a particular application scenario. All specific implementations of the criteria evaluation functions $f_i(.)$ are implementation specific and are out of the scope of this paper. The proposed optimization procedure is not limited to any specific amount or nature of the objective functions as long as they follow a few common criteria:

- A return value of the objective function must be a positive real number.
- Better values of the criteria must be expressed by smaller numbers (this is because the particle swarm optimization method searches for a minimum of the function).

Generally, one common feature of all these objective functions is that they are mutually exclusive. Any optimization of one objective will often be at the expense of affecting the other one. For example, we may consider moving all the services to more secure fog nodes to increase security, but such a service distribution will likely cause reduced power efficiency, excessive load on some of the nodes, and a lower overall runtime of the system. Moreover, different objectives have different measurement units, e.g., security may be evaluated in bits while the power requirement of the services is measured in Watts, any available network bandwidth is measured in kbps, etc. Even if all the measurements are converted to real positive numbers, it is still very difficult to objectively compare them. There is no single solution to a multi-objective optimization problem that optimizes all the objectives at the same time. The objective functions are contradictory in this situation, therefore a set of non-dominated (Pareto optimal) solutions can be found. We propose to use a two-stage optimization procedure (see Figure 4) in order to deal with this situation, where the first step will use a multi-objective optimization to find a set of solutions (possible distributions of services), the elements of which are a Pareto optimal. We propose to use for this the integer multi-objective particle swarm optimization (IMOPSO) method described in the next paragraph. A choice of the particle swarm optimization method is based on the research of other authors [33–35] which shows that this method is suitable for a similar class of problems, and it demonstrates good results. We will use the analytical hierarchy process (AHP) in the second step to choose the best solution from a Pareto optimal set.

4.3. IMOPSO for Finding a Pareto Set of Possible Service Distributions

The original particle swarm optimization (PSO) algorithm is best suited for an optimization of continuous problems, but several modifications [36,37] exist, which enable it to be used for discrete problems. In the case of multiple objectives which contradict each other, the PSO algorithm may be adapted to find a Pareto optimal set of solutions [38,39]. We used the Multi-objective particle swarm optimization (MOPSO) method proposed by Coello et. all in [39] to find a Pareto set of the possible service distributions among fog nodes. In order to use this method, we had to slightly adapt it for it to work in the constrained integer n-dimensional space of possible distributions of services represented as the particles of a swarm (the original method uses a continuous real number space).

We used the vector $X_i = (x_1, x_2, \ldots, x_n)^T$, $x_j \in \{1, 2, \ldots, k\}$, $j = 1, 2, \ldots, n$ to encode the i-th distribution of services, where n is the number of services which have to be distributed among k fog nodes. The meaning of the vector element $x_j = l$ is that the j-th service must be placed in the l-th fog node.

A flow diagram of the integer multi-objective particle swarm optimization (IMOPSO) algorithm is shown in Figure 5.

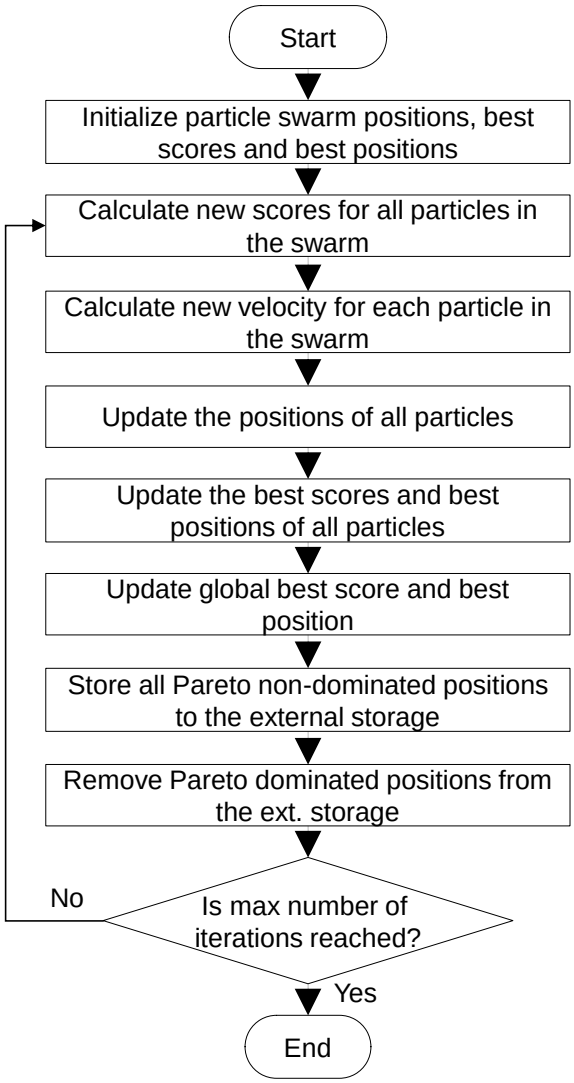

Figure 5. Flow chart of IMOPSO algorithm.

The main steps of the IMOPSO algorithm are the following:

1. Initialize the particle swarm S by randomly generating an initial set of positions of the particles (possible service distributions) X_i, $i = 1, 2, \ldots, |S|$, where $|S|$ is the initial size of a particle swarm.
2. Initialize the velocities $V_i = \vec{0}$, the best scores $pBest_i = \vec{inf}$, and the best positions $pBPos_i = X_i$ of all the particles in the swarm S. Initialize the global best position $gBPos = \vec{0}$ and the global best score $gBPos = \vec{inf}$.
3. Repeat it until a maximum number of iterations is reached:
   - Evaluate the new scores F_i of all the particles in the swarm S using all the objective functions: $F_i = (f_1(X_i), f_2(X_i), \ldots, f_m(X_i))^T$, $i = 1, 2, \ldots, |S|$.
   - Calculate new velocities of each particle using the expression $V_i = wV_i + r_1(pBPos_i - X_i) + r_2(gBPos - X_i)$, where w is an inertia weight (initially a real value around 0.4); r_1 and r_2 are random numbers in the range of [0..1]; V_i is the velocity of the i-th particle; $pBPos_i$ is the position of the i-th particle with the best score; X_i is the current position of the i-th particle; and $gBPos$ is the position of the particle with the best global score.
   - Update positions of all the particles in the swarm: $X_i = round(X_i + V_i)$, $i = 1, 2, \ldots, |S|$. The position is approximated to the nearest integer value.

If the particle is out of the range, give it the opposite direction of the speed ($V_i = -V_i$), and set the position X_i to the edge of the range of its definition.
- Update all the best scores $pBest_i$ and the best positions $pBPos_i$ of all the particles in the swarm $i = 1, 2, \ldots, |S|$. If the new score F_i dominates the current best score $pBest_i$, then update the best position and the best score of the i-th particle. If the new score neither dominates nor is dominated by the current best score of the i-th particle, then set the best position (and the best score) of the particle to a new position with a probability of 0.5.
- Update the global best score $gBest$ and the global best position $gBPos$ of the particles using the same algorithm used for updating the best scores of the individual particles.
- Store the positions and scores of the particles that are non-dominated in the external repository (set R). Use all the available particles in the sets S and R during any dominance comparison.
- Analyze the repository R and remove all the duplicated and dominated scores.

4. The external repository R is a set of Pareto optimal solutions.

4.4. Finding an Optimal Service Distribution Using the AHP

We used the analytical hierarchy process (AHP) [40,41] to choose the best solution from a Pareto optimal set by using pairwise comparisons of all non-dominated distributions of services using all the available objective functions. The AHP is usually used in situations where a decision must be made using a small amount of quantitative data, using a deep analysis performed by several decision-making parties, by applying a pair-to-pair comparison of possible solutions. The AHP may be adapted to be used by machine-based decision making in the scenarios where complex multiple criteria problems are evaluated [42–44]. The choice of the AHP instead of other more formal multi-criteria decision-making algorithms is based on the following reasons [40].

The AHP allows one to automatically check the consistency of the evaluations provided by decision makers. The AHP uses normalized values of criteria, so it allows one to use heterogeneous measurement scales for different criteria. For example, one can use a purely qualitative scale for the security (high, low, medium) and use inconsistent numeric scales for any power and CPU requirements at the same moment. The AHP uses pairwise comparisons of the alternatives only, which eases multi-objective decision making to obtain improved reliability of the results. The importance of the criteria used in the AHP is also evaluated using the same methodology, which allows one to skip the most controversial step of a manual weight assignment to different criteria.

A three-level hierarchical structure of the AHP is generalized in Figure 6. Level one is an objective of the process which in our case is to choose an optimal distribution of all the available services among fog nodes. The second level is the criteria, which are the same as the objective functions used in the IMOPSO part of the optimization process. An important step in this level is to use the same AHP to find the weight of all the criteria by using a pairwise comparison of the criteria. This step should be done manually before putting an automatic service allocation algorithm into production. Moreover, a step of the evaluation of criterion importance should be different based on the application area in which a service orchestrator is applied. For example, security may be evaluated as more important than power efficiency in a healthcare application compared to a home automation application. We assume in our algorithm that the step of the evaluation of criterion importance is already performed, and the decision-making system already has its judgment matrix with all the required weights of all the criteria in level 2.

The third level is alternatives. These are filled with all the Pareto optimal solutions from a previous step of the optimization process using the IMOPSO method. Then, the AHP is started to choose the best alternative. The whole process is summarized in Figure 7.

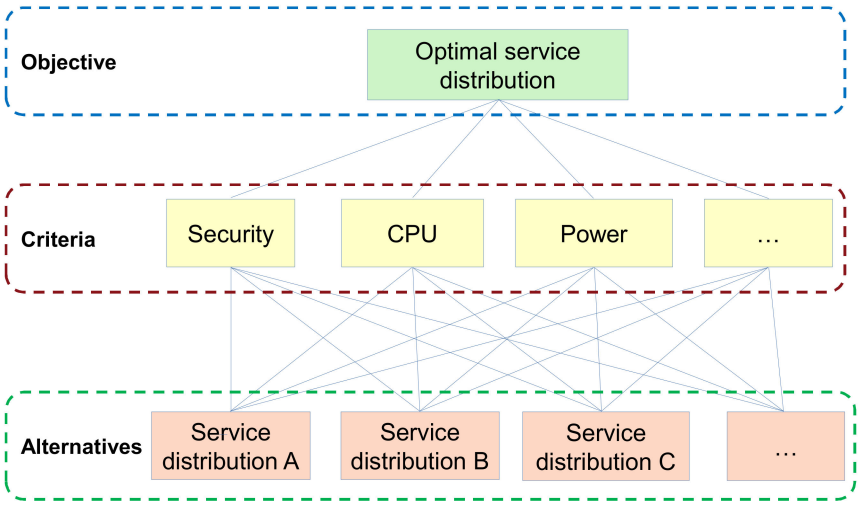

Figure 6. Hierarchical framework for AHP.

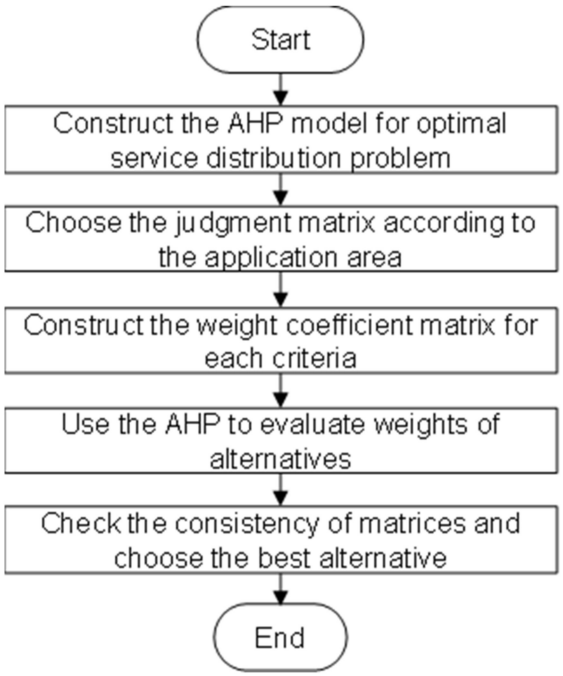

Figure 7. Process of AHP decision making.

The main steps of the AHP are the following:

1. Construct a corresponding AHP framework using a Pareto optimal solution set to prepare all the data structures for a comparison of alternatives using all the available criteria.
2. Load a judgment matrix with the results of pairwise comparisons of criteria prepared to be used in the current application area.
3. Repeat the following step for each criterion (objective function) $f_k(.)$, $k = 1, 2, \ldots, m$:
   - Construct the weight coefficient matrix $M_k = (m_{i,j})$ using all the alternatives in the Pareto optimal solution set R. The size of the matrix M_k is $s \times s$, where $s = |R|$; $m_{i,j} \in (0, 9]$; $m_{i,j} = \frac{1}{m_{j,i}}$; $m_{i,i} = 1$; $i, j = 1, 2, \ldots, s$. The matrix M_k elements are calculated using special comparison functions $m_{i,j} = comp_k(X_i, X_j)$ which use a corresponding objective function $f_k(.)$, which calculates two objective function values $f_k(X_i)$ and $f_k(X_j)$, and compares them with each other to transform the result into the required real number from the interval $(0, 9]$. A comparison

function heavily depends on the meaning of the criteria and the corresponding real number represents a preference of one alternative over another [35].

4. Provide all the created matrices to the standard AHP decision-making method to obtain any estimated weights of all the alternatives.
5. Check the consistency of the provided matrices using consistency indicators provided by the AHP. Choose the best alternative as the final solution of an optimization process.

5. Implementation and Evaluation

Implementation results of our method are summarized in this section with a discussion on each result. The implementation of a real fog computing environment with a measurement of all the parameters used in the service placement decisions is out of the scope of this paper, and it also makes it difficult to scale the solution and reproduce the results, therefore, we used a simulation. The main objective is to show how the proposed method performs in different situations as well as to test the feasibility of the proposed service placement method.

We implemented the proposed optimal service placement-finding method using Matlab. The implementation uses as an input some basic performance information on the fog nodes and services, a set of the objective functions, and an application area-specific judgment matrix J. The method performs integer multi-objective particle swarm optimization, finds a Pareto optimal set of solutions, automatically performs an AHP using a provided judgment matrix, and finds the best placement of the services in the fog nodes.

5.1. Illustrative Scenario

We used an illustrative scenario to evaluate the characteristics of the proposed method. We have 4 fog nodes and 13 services in this scenario, and they must be optimally placed in those fog nodes. Capabilities of the fog nodes and requirements of the services are chosen to show how the proposed method performs in different situations. We used several papers [19,45,46] analyzing various requirements of real hardware and software IoT systems to provide realistic numbers. A summary of the fog node parameters and requirements of the services are presented in Tables 2 and 3.

A security level of any fog device is determined by the hardware and software capabilities as well as by the availability of corresponding libraries, and it is expressed in bits according to the NIST guidelines [32].

All services are divided into three main groups. Sense1, Sense2, and Sense3 services are primarily used to communicate with any corresponding sensor devices, collect the measurement data, and provide it to the other services for processing. On the other hand, services Actuate1, Actuate2, and Actuate3 are mainly used to communicate with the actuator devices. The rest of the services are primarily used to collect data, perform calculations, and make decisions. Resource requirements of the services from different classes are very different.

Table 2. Resources available in the fog nodes.

	Power (mW)	CPU (MIPS)	RAM (MB)	Security (bits)
Fog1	1000	2000	512	256
Fog2	2000	1000	256	112
Fog3	1000	1000	256	128
Fog4	2000	500	512	86

Table 3. Resources required by the services.

Service	Processing Power (mW)	Transfer Power (mW)	CPU (MIPS)	RAM (MB)
Sense1	5	20	50	10
Sense2	5	25	60	15
Sense3	5	20	50	20
Process1	100	0	200	60
Process2	150	0	250	75
Process3	130	0	230	70
Process4	120	0	300	50
Process5	120	0	240	80
Process6	140	0	250	55
Process7	200	0	200	70
Actuate1	4	21	50	10
Actuate2	5	20	60	15
Actuate3	4	19	50	10

We used a "dynamic" objective function representing power requirements of the service to better illustrate the capabilities of our method. The power requirements of the service depend on which fog node is used to host this service. This is achieved by dividing the power requirements into two parts: processing and transfer power. The processing power is constant, and it is always required to perform an operation (the values of power requirements were taken from the publication [19]). On the other hand, the transfer power presented in Table 3 is required if no security is used to transfer the data (i.e., a plain http protocol is used). The information on required power levels for a data transfer without any security is based on the experimental results presented in the paper [47]. When the service is placed in a fog node providing more security, then the corresponding requirement for a transfer power is increased. For example, if a service is placed in a fog node providing 86 bits of security (e.g., this node is using 1024-bit RSA for a key agreement), then the corresponding transfer power is multiplied by a coefficient of 1.5. The transfer power increase coefficients were based on the results presented in [45] and [48]. We decided after an analysis of the provided data to use these multipliers for modeling the increase in power due to increased security: 1.5 for 86 bits of security, 2.25 for 112 bits, 4 for 128 bits, and 7.5 for 256 bits of security.

5.2. Evaluation Results

We use a simplified scenario where only two objective functions are used to show how the IMOPSO algorithm works and how the Pareto set of solutions looks. A Pareto set may be displayed in this case using a two-dimensional chart. A judgment matrix used in this case consists only of 4 elements:

$$J = \begin{pmatrix} 1 & 3 \\ 1/3 & 1 \end{pmatrix} \tag{3}$$

If two objective functions, RAM and CPU, are used, then this judgment matrix means that an even RAM usage distribution among all the fog nodes is more important than an even CPU usage. A Pareto set produced by the IMOPSO algorithm is presented in Figure 8. Then, a Pareto solution set is used in the second stage, employing an AHP, to find the best placement of services. The best placement is summarized in Table 4.

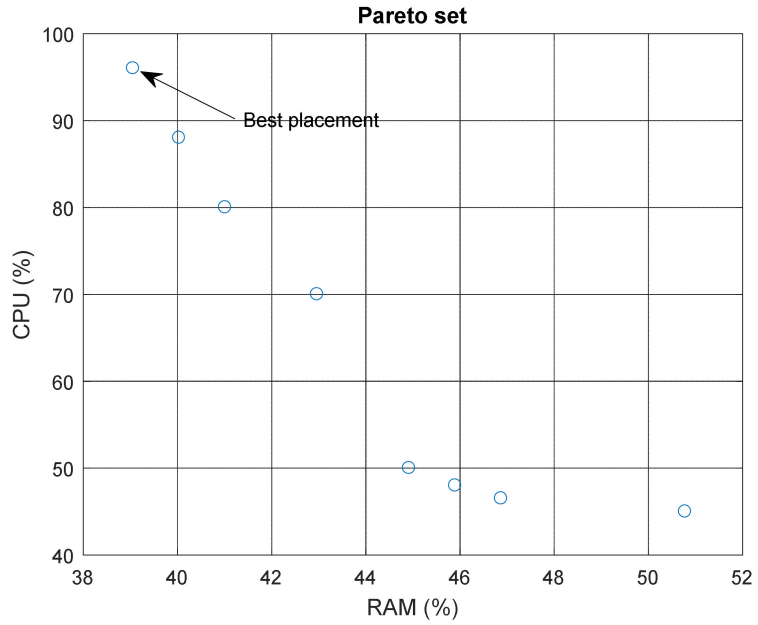

Figure 8. Pareto set of a simplified scenario.

Table 4. The best service placement in a simplified scenario.

	Fog1	Fog2	Fog3	Fog4
Services	Actuate3 Process4 Process5 Process6	Sense1 Sense2 Actuate2 Process1	Sense3 Process2	Actuate1 Process3 Process7

The best score (the best values of the objective functions) in this case is $(39,\ 96)^T$, meaning that this service placement ensures a maximal RAM usage of 39% among all four fog nodes. The maximal usage of CPU is 96% in this case.

The second scenario shows an influence of the judgment matrix on the optimal placement of services. Four objective functions are used in this case: power, CPU, security, and RAM. The first judgment matrix prioritizes security and energy over the CPU and RAM:

$$J_1 = \begin{pmatrix} 1 & 3 & 1/6 & 3 \\ 1/3 & 1 & 1/6 & 1 \\ 6 & 6 & 1 & 6 \\ 1/3 & 1 & 1/6 & 1 \end{pmatrix} \tag{4}$$

The second judgment matrix prioritizes an even power consumption:

$$J_2 = \begin{pmatrix} 1 & 7 & 3 & 6 \\ 1/7 & 1 & 1/2 & 1 \\ 1/3 & 2 & 1 & 2 \\ 1/6 & 1 & 1/2 & 1 \end{pmatrix} \tag{5}$$

A Pareto set of solutions using the judgment matrix J_1 is shown in Figure 9. Only some projections of the set are shown as the set members are four-dimensional vectors and they cannot be fully rendered in charts.

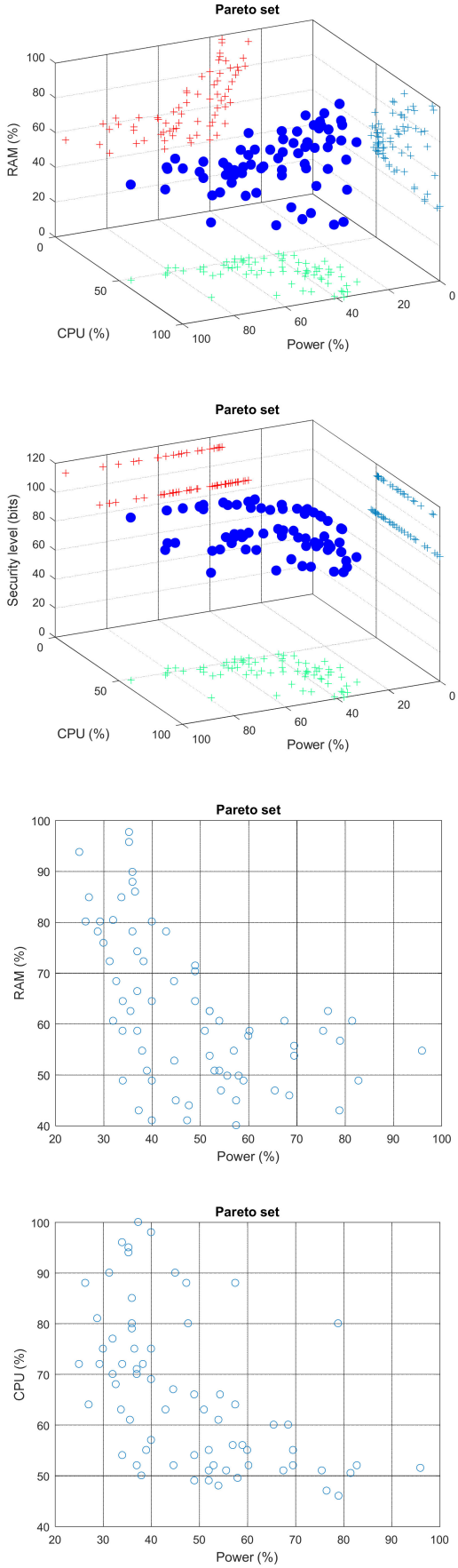

Figure 9. Pareto set of the second scenario, security and energy are prioritized.

The best placement of services is presented in Table 5. The best score in this case is $(35,\ 94,\ 112,\ 98)^T$. The overall security (determined by a security level of the least security-capable fog node hosting at least one service) is 112 bits in this case, and the fog node Fog4 is not hosting any services, as its security is only 86 bits.

Table 5. Best service placement in the second scenario, security is prioritized over other criteria.

	Fog1	Fog2	Fog3	Fog4
Services	Process2 Process7	Sense1, Sense2, Sense3, Process1, Process3, Process6 Activate1, Activate3	Process4 Process5 Activate2	-

If the judgment matrix J_2, which prioritizes even power consumption, is used in the same situation, then the best placement is different (see Table 6), and the best score is $(26,\ 88,\ 86,\ 84)^T$.

Table 6. Best service placement in the second scenario, power is prioritized over other criteria.

	Fog1	Fog2	Fog3	Fog4
Services	Process5 Process6	Process2, Process3, Process4, Activate1, Activate3	Process1	Sense1, Sense2, Sense3, Process7, Activate2

The maximal power consumption among all the fog nodes is 26% in this case, and it is significantly better than in the first variant (35%), but the overall security of the solution is degraded to 86 bits, as several services are placed in the fog node Fog4.

The third illustrative scenario is meant to illustrate how the proposed service placement method works in cases when some devices change their positions, and corresponding services must be reallocated. We use an objective function considering the range from a physical sensor device to the service monitoring device which is physically placed in one of the fog nodes to demonstrate this scenario. The range in this case is only important for services which are communicating with sensors or actuators. The range is considered 0 independently of the fog nodes they are hosted in with the services which are processing data. A judgment matrix prioritizing the range is used in this scenario, while the objective functions in this case are: range, CPU, security, RAM.

$$J_3 = \begin{pmatrix} 1 & 5 & 3 & 5 \\ 1/5 & 1 & 1/2 & 1 \\ 1/3 & 2 & 1 & 2 \\ 1/5 & 1 & 1/2 & 1 \end{pmatrix} \tag{6}$$

We used the data presented in the diagram (see Figure 10) to model the placement of the services. All coordinates here are presented in meters.

The best service placements in each case are summarized in Tables 7 and 8, and the corresponding scores are $(13,\ 67,\ 128,\ 73)^T$ and $(9,\ 54,\ 86,\ 55)^T$.

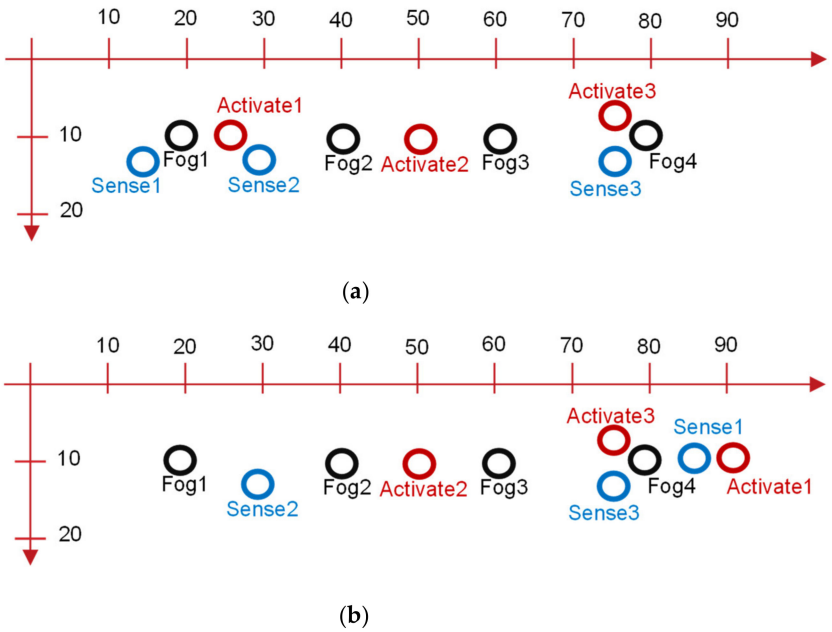

Figure 10. Service placement diagram. (**a**) Initial placement, (**b**) modified placement.

Table 7. Best service placement in the third scenario, the initial placement of sensors and actuators.

	Fog1	Fog2	Fog3	Fog4
Services	Sense1, Sense2, Activate1, Process1, Process2, Process3, Process5, Process6.	-	Sense3, Activate2, Activate3, Process4, Process7.	-

Table 8. Best service placement in the third scenario, the placement of sensors and actuators after changes in their location.

	Fog1	Fog2	Fog3	Fog4
Services	Process1, Process2, Process4, Process7	Sense2, Activate1, Process5	Process3, Process6	Sense1, Sense3, Activate1, Activate3

The evaluation results clearly show that if the range is the most important objective function, then the services are more likely to be placed in the adjacent fog nodes. On the other hand, if more sensors are located near a less secure fog node, then the overall security of the solutions may decrease (128 bits vs. 86 bits in the second scenario).

6. Discussion, Conclusions, and Future Work

An increase in IoT-based services has led to a need for more efficient means of handling resources in systems comprising heterogeneous devices. A fog computing paradigm brings computational resources closer to the edge of the cloud, but energy-, communication-, and computation resource-constrained devices dominate near the edge. Different application areas (healthcare, multimedia, home automation, etc.) require different characteristics of the IoT system. The usage of various heterogeneous devices leads to difficulties in predicting how much of the resources would be required within the fog nodes when all the services are going to allocate all the resources they need. Moreover, the need for roaming services which follow the actors (i.e., a person is moving inside a building, cars, etc.) arises due to the limitations of some hardware devices (i.e., a limited range of communication protocols), and therefore the resources in the fog nodes need to be reallocated in this

case every time the situation changes. The best way to deal with these dynamic service reallocations is to use service orchestrators, which decide the best way to allocate and move, start, and stop any corresponding services as needed. One of the main challenges while designing an effective service orchestrator is the need for a specialized method to obtain an optimized service placement inside the available fog nodes.

A new optimization method for an optimal distribution of services among available fog nodes was proposed in this paper. The two-stage method uses integer multi-objective particle swarm optimization to find a Pareto optimal set of solutions and the analytical hierarchy process using an application-specific judgment matrix for a decision on any optimal distribution of services. Such a processing distribution allows one to assess different heterogeneous criteria with different units of a measurement and different natures (qualitative or quantitative). The method, apart from providing one best solution, also ranks all the Pareto optimal solutions, enabling one to compare them with each other (answering the question "how much better is one solution than the other?") and, if needed, to choose the second best, the third best, etc. solution.

The proposed method effectively works with the whole range of objective functions (evaluation criteria), which could be easily expanded by new objective functions representing different criteria. Moreover, the objective functions may be dynamic, meaning that not only the value but also the algorithm of an objective function calculation may be different based on the service placement in particular fog nodes with particular software and hardware capabilities.

If the same end device, service, and fog device set is used in a different application area (i.e., healthcare vs. home automation) which requires different prioritization of criteria (i.e., security is more important in healthcare compared to home automation) then only the AHP judgment matrix must be changed. The method adapts to the situation and provides appropriate results.

A number of interesting aspects of the proposed method could be explored in the future. It would be interesting to use it in a real orchestrator of IoT infrastructure to practically evaluate how different placements of services inside fog nodes influence the performance of the whole IoT system. Another very interesting aspect to investigate is objective function construction according to the experimentally obtained real-life results involving all the interrelations among different criteria. An experiment using real hardware and software would help to estimate some additional aspects of the proposed algorithm, including the performance under different configurations of the infrastructure (number of fog nodes, number of end devices, etc.) and different architectures of the corresponding devices (supporting parallel processing, optimization using CPU or GPU, offloading an optimization task to the cloud services, etc.).

We believe that the results of this work will be useful in further research in the area of IoT fog computing service orchestration, and it will allow researchers to develop more efficient IoT systems.

Author Contributions: Conceptualization, A.V., N.M., N.Š.; investigation, N.M., A.V., J.T.; methodology, A.V., N.M., N.Š.; resources, A.V.; software, N.M., J.T.; supervision, A.V.; visualization, N.M., N.Š.; writing—original draft, A.V., N.M., N.Š.; writing—review and editing, A.V., N.M., N.Š., J.T.; funding acquisition, A.V. All authors contributed to the final version. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported in part by the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 830892, project "Strategic programs for advanced re-search and technology in Europe" (SPARTA).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Velasquez, K.; Abreu, D.P.; Goncalves, D.; Bittencourt, L.; Curado, M.; Monteiro, E.; Madeira, E. Service Orchestration in Fog Environments. In Proceedings of the 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, Czech Republic, 21–23 August 2017; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2017; pp. 329–336.
2. Javadzadeh, G.; Rahmani, A.M. Fog Computing Applications in Smart Cities: A Systematic Survey. *Wirel. Netw.* **2020**, *26*, 1433–1457. [CrossRef]
3. Imrith, V.N.; Ranaweera, P.; Jugurnauth, R.A.; Liyanage, M. Dynamic Orchestration of Security Services at Fog Nodes for 5G IoT. In Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2020; pp. 1–6.
4. Kayes, A.; Rahayu, W.; Watters, P.; Alazab, M.; Dillon, T.; Chang, E. Achieving security scalability and flexibility using Fog-Based Context-Aware Access Control. *Futur. Gener. Comput. Syst.* **2020**, *107*, 307–323. [CrossRef]
5. De Brito, M.S.; Hoque, S.; Magedanz, T.; Steinke, R.; Willner, A.; Nehls, D.; Keils, O.; Schreiner, F. A service orchestration architecture for Fog-enabled infrastructures. In Proceedings of the 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), Valencia, Spain, 8–11 May 2017; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2017; pp. 127–132.
6. Dsouza, C.; Ahn, G.-J.; Taguinod, M. Policy-driven security management for fog computing: Preliminary framework and a case study. In Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014), Redwood City, CA, USA, 13–15 August 2014; IEEE: New York, NY, USA, 2014; pp. 16–23.
7. Capossele, A.; Cervo, V.; De Cicco, G.; Petrioli, C. Security as a CoAP resource: An optimized DTLS implementation for the IoT. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015; IEEE: New York, NY, USA, 2015; pp. 549–554.
8. Jamil, B.; Shojafar, M.; Ahmed, I.; Ullah, A.; Munir, K.; Ijaz, H. A job scheduling algorithm for delay and performance optimization in fog computing. *Concurr. Comput. Pr. Exp.* **2019**, *32*, 5581. [CrossRef]
9. Aslanpour, M.S.; Gill, S.S.; Toosi, A.N. Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet Things* **2020**, *12*, 100273. [CrossRef]
10. Tsai, J.-S.; Chuang, I.-H.; Liu, J.-J.; Kuo, Y.-H.; Liao, W. QoS-Aware Fog Service Orchestration for Industrial Internet of Things. *IEEE Trans. Serv. Comput.* **2020**, 1. [CrossRef]
11. De Sousa, N.F.S.; Perez, D.A.L.; Rosa, R.V.; Santos, M.; Rothenberg, C.E. Network Service Orchestration: A survey. *Comput. Commun.* **2019**, *142–143*, 69–94. [CrossRef]
12. Šatkauskas, N.; Venčkauskas, A.; Morkevičius, N.; Liutkevičius, A. Orchestration Security Challenges in the Fog Compu-ting. In *Communications in Computer and Information Science, Proceedings of the International Conference on Information and Software Technologies, Kaunas, Lithuania, 15–17 October 2020*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 196–207.
13. Velasquez, K.; Abreu, D.P.; Assis, M.R.M.; Senna, C.; Aranha, D.; Bittencourt, L.F.; Laranjeiro, N.; Curado, M.; Vieira, M.; Monteiro, E.; et al. Fog orchestration for the Internet of Everything: State-of-the-art and research challenges. *J. Internet Serv. Appl.* **2018**, *9*, 14. [CrossRef]
14. Desikan, K.S.; Kotagi, V.J.; Murthy, C.S.R. Topology Control in Fog Computing Enabled IoT Networks for Smart Cities. *Comput. Netw.* **2020**, *176*, 107270. [CrossRef]
15. Zahmatkesh, H.; Al-Turjman, F. Fog computing for sustainable smart cities in the IoT era: Caching techniques and enabling technologies—An overview. *Sustain. Cities Soc.* **2020**, *59*, 102139. [CrossRef]
16. Zhang, C. Design and application of fog computing and Internet of Things service platform for smart city. *Futur. Gener. Comput. Syst.* **2020**, *112*, 630–640. [CrossRef]
17. Yang, R.; Wen, Z.; McKee, D.; Lin, T.; Xu, J.; Garraghan, P. Fog Orchestration and Simulation for IoT Services. In *Fog and Fogonomics*; Yang, Y., Huang, J., Zhang, T., Weinman, J., Eds.; Wiley: Hoboken, NJ, USA, 2020; pp. 179–212. ISBN 978-1-119-50109-1.
18. Wen, Z.; Yang, R.; Garraghan, P.; Lin, T.; Xu, J.; Rovatsos, M. Fog Orchestration for Internet of Things Services. *IEEE Internet Comput.* **2017**, *21*, 16–24. [CrossRef]
19. Skarlat, O.; Nardelli, M.; Schulte, S.; Borkowski, M.; Leitner, P. Optimized IoT service placement in the fog. *Serv. Oriented Comput. Appl.* **2017**, *11*, 427–443. [CrossRef]
20. Naha, R.K.; Garg, S.; Chan, A.; Battula, S.K. Deadline-based dynamic resource allocation and provisioning algorithms in Fog-Cloud environment. *Futur. Gener. Comput. Syst.* **2020**, *104*, 131–141. [CrossRef]
21. Khebbeb, K.; Hameurlain, N.; Belala, F. A Maude-Based rewriting approach to model and verify Cloud/Fog self-adaptation and orchestration. *J. Syst. Arch.* **2020**, *110*, 101821. [CrossRef]
22. Tuli, S.; Poojara, S.R.; Srirama, S.N.; Casale, G.; Jennings, N.R. COSCO: Container Orchestration using Co-Simulation and Gradient Based Optimization for Fog Computing Environments. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *33*, 1. [CrossRef]
23. Rahbari, D.; Nickray, M. Task offloading in mobile fog computing by classification and regression tree. *Peer Peer Netw. Appl.* **2019**, *13*, 104–122. [CrossRef]
24. Arkian, H.R.; Diyanat, A.; Pourkhalili, A. MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications. *J. Netw. Comput. Appl.* **2017**, *82*, 152–165. [CrossRef]

25. Velasquez, K.; Abreu, D.P.; Curado, M.; Monteiro, E. Service placement for latency reduction in the internet of things. *Ann. Telecommun.* **2016**, *72*, 105–115. [CrossRef]
26. Brogi, A.; Forti, S.; Ibrahim, A. How to Best Deploy Your Fog Applications, Probably. In Proceedings of the 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), Madrid, Spain, 14–15 May 2017; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2017; pp. 105–114.
27. Urgaonkar, R.; Wang, S.; He, T.; Zafer, M.; Chan, K.; Leung, K.K. Dynamic service migration and workload scheduling in edge-clouds. *Perform. Eval.* **2015**, *91*, 205–228. [CrossRef]
28. Bellendorf, J.; Mann, Z. Ádám Classification of optimization problems in fog computing. *Futur. Gener. Comput. Syst.* **2020**, *107*, 158–176. [CrossRef]
29. Huang, X.; Ganapathy, S.; Wolf, T. Evaluating Algorithms for Composable Service Placement in Computer Networks. In Proceedings of the 2009 IEEE International Conference on Communications, Dresden, Germany, 14–18 June 2009; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2009; pp. 1–6.
30. Mann, Z.Á. *Optimization in Computer Engineering–Theory and Applications*; Scientific Research Publishing: Wuhan, China, 2011; ISBN 978-1-935068-58-7.
31. Guerrero, C.; Lera, I.; Juiz, C. A lightweight decentralized service placement policy for performance optimization in fog computing. *J. Ambient. Intell. Humaniz. Comput.* **2018**, *10*, 2435–2452. [CrossRef]
32. Barker, E. *Recommendation for Key Management Part 1: General*; NIST Special Publication 800-57 Part 1 Revision 4; NIST: Gaithersburg, MD, USA, 2016.
33. Yu, B.; Wu, S.; Jiao, Z.; Shang, Y. Multi-Objective Optimization Design of an Electrohydrostatic Actuator Based on a Particle Swarm Optimization Algorithm and an Analytic Hierarchy Process. *Energies* **2018**, *11*, 2426. [CrossRef]
34. Yang, T.; Huang, Z.; Pen, H.; Zhang, Y. Optimal Planning of Communication System of CPS for Distribution Network. *J. Sens.* **2017**, *2017*. [CrossRef]
35. Pan, Q.-K.; Tasgetiren, M.F.; Liang, Y.-C. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Comput. Oper. Res.* **2008**, *35*, 2807–2839. [CrossRef]
36. Wang, B.Z.; Deng, X.; Ye, W.C.; Wei, H.F. Study on Discrete Particle Swarm Optimization Algorithm. *Appl. Mech. Mater.* **2012**, *220–223*, 1787–1794. [CrossRef]
37. Strasser, S.; Goodman, R.; Sheppard, J.; Butcher, S. A New Discrete Particle Swarm Optimization Algorithm. In Proceedings of the 2016 on SIGMOD'16 PhD Symposium, San Francisco, CA, USA, 26 June–1 July 2016; ACM: New York, NY, USA, 2016; pp. 53–60.
38. Wang, L.; Ye, W.; Fu, X.; Menhas, M.I. A Modified Multi-Objective Binary Particle Swarm Optimization Algorithm. In *Lecture Notes in Computer Science, Proceedings of the Advances in Swarm Intelligence, Chongqing, China, 12–15 June 2011*; Tan, Y., Shi, Y., Chai, Y., Wang, G., Eds.; Springer Berlin Heidelberg: Berlin/Heidelberg, Germany, 2011; pp. 41–48.
39. Coello, C.C.; Toscano-Pulido, G.; Lechuga, M. Handling multiple objectives with particle swarm optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 256–279. [CrossRef]
40. Khaira, A.; Dwivedi, R.K. A State of the Art Review of Analytical Hierarchy Process. *Mater. Today Proc.* **2018**, *5*, 4029–4035. [CrossRef]
41. Saaty, T.L.; Vargas, L.G. The Seven Pillars of the Analytic Hierarchy Process. In *Models, Methods, Concepts & Applications of the Analytic Hierarchy Process*; Springer: Boston, MA, USA, 2001; pp. 27–46. ISBN 978-1-4615-1665-1.
42. Higgins, M.; Benaroya, H. Utilizing the Analytical Hierarchy Process to determine the optimal lunar habitat configuration. *Acta Astronaut.* **2020**, *173*, 145–154. [CrossRef]
43. Cheikhrouhou, O.; Koubaa, A.; Zaard, A. Analytical Hierarchy Process based Multi-objective Multiple Traveling Salesman Problem. In Proceedings of the 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), Bragança, Portugal, 4–6 May 2016; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2016; pp. 130–136.
44. Chang, S.-J.; Li, T.-H.S. Design and Implementation of Fuzzy Parallel-Parking Control for a Car-Type Mobile Robot. *J. Intell. Robot. Syst.* **2002**, *34*, 175–194. [CrossRef]
45. Suárez-Albela, M.; Fernández-Caramés, T.M.; Fraga-Lamas, P.; Castedo, L. A Practical Evaluation of a High-Security Energy-Efficient Gateway for IoT Fog Computing Applications. *Sensors* **2017**, *17*, 1978. [CrossRef]
46. Aazam, M.; Huh, E.-N. Dynamic resource provisioning through Fog micro datacenter. In Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), St. Louis, MO, USA, 23–27 March 2015; IEEE: New York, NY, USA, 2015; pp. 105–110.
47. Venčkauskas, A.; Morkevicius, N.; Jukavičius, V.; Damaševičius, R.; Toldinas, J.; Grigaliūnas, Š. An Edge-Fog Secure Self-Authenticable Data Transfer Protocol. *Sensors* **2019**, *19*, 3612. [CrossRef] [PubMed]
48. Suárez-Albela, M.; Fraga-Lamas, P.; Fernández-Caramés, T.M. A Practical Evaluation on RSA and ECC-Based Cipher Suites for IoT High-Security Energy-Efficient Fog and Mist Computing Devices. *Sensors* **2018**, *18*, 3868. [CrossRef] [PubMed]